Proceedings of the
18th International Conference on
Digital Audio Effects

DAFx15

Trondheim, Norway
November 30 - December 3, 2015

Proceedings published by:
Department of Music and Department of Electronics and Telecommunications
Norwegian University of Science and Technology
November 2015
Editors: Peter Svensson and Ulf Kristiansen

Credits:
Logo design DAFx-15: Amund Ulvestad
Cover design: Sigurd Saue and NTNU Grafisk senter
Cover photo: Endre Forbord

Web sites:
www.ntnu.edu/dafx15
www.dafx.de

ISSN 2413-6700 (Print)
ISSN 2413-6689 (Online)

Foreword

Welcome to DAFx-15 in Trondheim

It is with great pleasure that we welcome you to the 18th International Conference on Digital Audio Effects in Trondheim, Norway from Monday November 30 to Thursday December 3, 2015. For the first time the DAFx conference series revisits a host city, 16 years after Trondheim hosted the 2nd DAFx conference in 1999 at the Norwegian University of Science and Technology (NTNU), organized primarily by the Department of Electronics and Telecommunications. Since then the Department of Music has established an ambitious Music Technology group and the two departments are jointly hosting this year's edition of the DAFx conference. We hope that you will sense an inspiring mix of music and audio technology in the program that we present.

DAFx-15 offers four days of scientific program intermingled with concerts and social events. The bulk of the conference consists of 61 papers selected for presentation Tuesday through Thursday: 31 oral presentations and 30 poster presentations. We have organized the oral presentations in 11 sessions, covering topics such as virtual analog, sound synthesis, audio analysis, physical modelling, spatial audio, audio coding, speech applications and audio effects. The posters are deliberately not organized into topics to favor exchanges between scientists of the same field of research. There are 10 poster presentations each day, including two short announcement sessions in the auditorium.

We wanted the conference to reflect how advances in audio technology finds application in music performance and specifically asked for contributions related to real-time applications of digital audio processing. This theme have not made a strong impact on the contributed papers, but we have strengthened it through keynotes and tutorials. We have invited three distinguished keynote speakers to kick off each conference day with an inspiring talk. On Tuesday, the merited artist and researcher Marije Baalman will discuss digital audio as open source hardware and software in a performance context, and she will return as performer at the evening concert. On Wednesday, the brilliant young researcher Kurt Wegner will present status quo on wave digital audio effects at CCRMA in a keynote prepared with Julius Smith. Finally, Dr Franz Zotter from IEM, Graz, and chair of a prize-winning initiative on virtual acoustics within the German acoustical society, will present a keynote on Ambisonics audio effects on Thursday.

Three tutorials launches the conference on Monday. Peter Svensson, professor of electroacoustics at NTNU, starts with a tutorial on "Sound field modeling for virtual acoustics". Professors Øyvind Brandtsegg and Trond Engum will follow with a tutorial/musical demonstration of "Cross-adaptive effects and real-time creative use" together with invited musicians. The third tutorial will be given by Xavier Serra who will present the AudioCommons initiative that "aims at bringing Creative Commons audio content to the creative industries".

There will be several musical contributions during the conference, but the main concert

event is held on Tuesday evening at Rockheim, the national museum of popular music. The social program also includes a welcome reception on Monday evening, right next to Nidaros cathedral, Norway's national sanctuary and it concludes with a conference dinner on Wednesday at the idyllic Ringve Museum, the national music museum.

We are proud to present to you the proceedings of DAFx-15. From this year on the publication is registered as an annual serial publication with an ISSN number. We hope this will make it even more attractive to contribute to future DAFx conferences. We have to thank all the great people that helped us make this year's conference happen, including the local organization team, all the reviewers from the DAFx-15 programme committee and the DAFx board members. Warm thanks go to the organizers of DAFx-13 and DAFx-14, who have been very helpful and responsive to our questions along the way. We would also like to extend our thanks to this years sponsors: Native Instruments GmbH, Audiokinetic Inc., Soundtoys Inc., Ableton AG and Aalberg Audio AS, and to our supportive institution, Norwegian University of Science and Technology (NTNU).

But most of all we would like to thank the DAFx community who makes these conferences possible by contributing great scientific work and enlightened discussions whenever we meet. We hope that DAFx-15 can play a similar role as earlier conferences in stimulating progress in the field of digital audio. When we presented our idea of DAFx-15 last year, we promised cold, darkness and snow. As we write these words, the first snow is falling heavily in the cold winter dark, so we will most likely keep our promise. Now it is all up to you, researchers, authors and participants at DAFx-15 to make this a memorable conference!
Welcome to Trondheim!

The DAFx-15 conference committee
Sigurd Saue (Music Technology): General Chair
Jan Tro (Electronics and Telecommunications): Vice Chair, Social program
Peter Svensson (Electronics and Telecommunications): Program Chair
Ulf Kristiansen (Electronics and Telecommunications): Program Co-chair
Øyvind Brandtsegg (Music Technology): Concerts Coordinator
Tim Cato Netland (Electronics and Telecommunications): Technical Coordinator

Conference Committees

DAFx Board

Organizing committee

Sigurd Saue
Jan Tro
Øyvind Brandtsegg
Ulf Kristiansen
Peter Svensson
Tim Cato Netland

Program Committee, Reviewers

Trevor Agus
Jens Ahrens
Roland Badeau
Søren Bech
Stefan Bilbao
Øyvind Brandtsegg
Jean Bresson
Michael Bürger
Andres Cabrera
Christophe D'Alessandro
Bertrand David
Kristjan Dempwolf
Philippe Depalle
Christian Dittmar
Ross Dunkel
Dan Ellis
Gianpaolo Evangelista
Sebastian Ewert
Bernhard Feiten
Federico Fontana
Volker Gnann
Pierre Hanna
Christian Hofmann
Craig Jin
Jari Kleimola
Malte Kob
Sebastian Kraft
Ulf Kristiansen
Wolfgang Kropp
Tapio Lokki
Simon Lui
Tom Lysaght
Jaromir Macak
Piotr Majdak
Sylvain Marchand
Rémi Mignot
Damian Murphy
Thibaud Necciari
Søren Nielsen
Jyri Pakarinen
Jouni Paulus
Geoffroy Peeters
Peter Pocta
Stephan Preihs
Ville Pulkki
Rudolf Rabenstein
Tor Ramstad
Josh Reiss
Mark Sandler
Augusto Sarti
Sigurd Saue
Gerald Schuller
Xavier Serra
Jan Skoglund
Julius Smith
Clifford So
Alex Southern
Nicolas Sturmel
Fabian-Robert Stöter
Peter Svensson
Sakari Tervo
Jan Tro
Tony Tew
Joseph Timoney
Toon van Waterschoot
Maarten van Walstijn
Christophe Vergez
Tuomas Virtanen
Vesa Välimäki
Carl Haakon Waadeland
Rory Walsh
Yi-Hsuan Yang
Udo Zoelzer
Franz Zotter

Table of Contents

Oral session 2: Physical Modeling

Oral session 3: Audio Effects

Posters

Oral session 4: Audio and Music Analysis

Keynote 2

Oral session 5: Audio Coding and Implementation

Posters

Oral session 6: Spatial Audio and Auralization

Oral session 7: Virtual Analog

Posters

Oral session 8: Speech Applications

Keynote 3

Oral session 9: Perceptually Based Applications

Posters

Oral session 10: Virtual Analog Approaches

Oral session 11: Physical Modelling

Posters

Tutorials

Peter Svensson: Sound field modeling for virtual acoustics

Abstract The terms virtual acoustics and auralization have been used for around 20 years for the generation of computer simulations of sound fields that can be listened to. This tutorial will give a brief overview over the components involved: the source modeling, the modeling of an environment via an impulse response, and the rendering stage. The focus will be on the modeling og environments, with the categories of physical modeling and perceptual modeling. Furthermore, the physical modeling can be done by accurately solving the wave equation, or by geometrical-acoustics based methods. Possibilities and limitations with these methods will be discussed, demonstrating the various reflection components of specular reflection, diffuse reflection, and diffraction. Examples will be shown using the authors Matlab Edge diffraction toolbox for generating animations of these phenomena.

Øyvind Brandtsegg and Trond Engum: Cross-adaptive effects and realtime creative use

Abstract Adaptive effects and modulations have been researched during the last two decades within the DAFx community, and cross-adaptive effects have been utilized for autonomous mixing and related applications. Current research into cross adaptive effects for creative use in realtime applications has led to the development of methods to incorporate these techniques into regular DAWs for audio production and performance. The tutorial will give insight into these methods, with practical examples on how to incorporate the tools in a DAW based workflow. Examples of use within live performance will also be presented.

Xavier Serra: The AudioCommons Initiative and the technologies for facilitating the reuse of open audio content

Abstract Significant amounts of user-generated audio content, such as sound effects, musical samples and music pieces, are uploaded to online repositories and made available under open licenses. Moreover, a constantly increasing amount of multimedia content, originally released with traditional licenses, is becoming public domain as its license expires. Nevertheless, this content is not much used in professional productions. There is still a lack of familiarity and understanding of the legal context of all this open content, but there are also problems related with its accessibility. A big percentage of this content remains unreachable either because is not published online or because it is not well organised and

annotated. With the Audio Commons Initiative we want to promote the use of open audio content and to develop technologies with which to support the ecosystem composed by content repositories, production tools and users. These technologies should enable the reuse of this audio material, facilitating its integration in the production workflows used by the creative industries. In this workshop we will go over the core ideas behind this initiative, then overview the existing audio repositories, technologies and production tools related to it, and finally outline the planned tasks to address the challenges posed by the initiative.

Keynote 1

Marije Baalman: Digital Audio Out of the Box - Digital Audio Effects in Art and Experimental Music

Abstract While since the late 1990's laptops have become a common element in electronic music on stage, in recent years there is a move away again from the laptop, towards dedicated devices that perform one particular task. With the advent of platforms such as the BeagleBone, Raspberry Pi, but also Arduino, efficient computing of digital audio has found a large interest amongst artists who create their own instruments or sounding objects, usually within the context of open source software and hardware. In this talk I will show various examples of these applications of digital audio in the field of art and experimental music; and discuss how their development and discourse is embedded in the open source movement.

MORPHING OF GRANULAR SOUNDS

Sadjad Siddiq
Advanced Technology Division,
Square Enix Co., Ltd.
Tokyo, Japan
siddsadj@square-enix.com

ABSTRACT

Granular sounds are commonly used in video games but the conventional approach of using recorded samples does not allow sound designers to modify these sounds. In this paper we present a technique to synthesize granular sound whose tone color lies at an arbitrary point between two given granular sound samples. We first extract grains and noise profiles from the recordings, morph between them and finally synthesize sound using the morphed data. During sound synthesis a number of parameters, such as the number of grains per second or the loudness distribution of the grains, can be altered to vary the sound. The proposed method does not only allow to create new sounds in real-time, it also drastically reduces the memory footprint of granular sounds by reducing a long recording to a few hundred grains of a few milliseconds length each.

1. INTRODUCTION

1.1. Granular synthesis in video games

In previous work we described the morphing between simple impact sounds [1]. In this paper we focus on morphing between complex sounds that consist of a large amount of similar auditory events. Examples of such sounds are the sound of rain, consisting of the sound of thousands of rain drops hitting the ground; the sound of running water, which is essentially the sound of thousands of resonating air bubbles in water [2]; the sound of popping fireworks or the sound of breaking rock [3]. Since we use granular synthesis to synthesise such sounds in real-time we call these sounds "granular sounds" and the auditory events they consist of "grains".

Although such sounds are used extensively in video games, they are rarely produced by granular synthesis. The most common approach is to rely on recorded samples, but this does not give sound designers much control over the sound. Additionally such samples are usually heavy on memory, because they must be quite long to avoid repetitiveness. Using granular synthesis to create such sounds from prerecorded or synthesized grains is a method that is much lighter on memory and gives sound designers a lot of freedom to modify the sound in multiple ways by modifying parameters of the synthesis [3].

Figure 1 summarizes the implementation of a simple granular synthesizer. In granular synthesis signals are produced by mixing grains. These grains are usually very short, in the range of 2 to 20 ms. The number of grains per second can be used as a parameter to control the granularity of the sound. A low number would allow the listener to perceive individual grains while a high number would result in a large amount of grains overlapping each other and create a signal close to white (or colored) noise. Before mixing, grains are usually modified in a number of ways. In the simplest case their amplitude is attenuated randomly. Depending on the probability distribution of the attenuation factors, this can have a very drastic effect on the granularity and tone color of the produced sound. For a more detailed introduction to granular synthesis refer to [4] or [5].

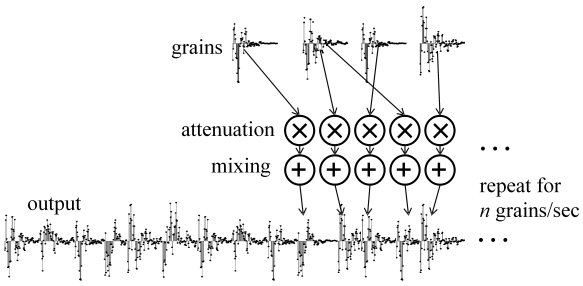


Figure 1: *Summary of granular synthesis. Grains are attenuated and randomly added to the final mix.*

When using granular synthesis in video games, the synthesis algorithm can be coupled to the physics engine to create realistic sounds. To implement the sound of breaking rock, as introduced in [3], we calculate the number of grains per second in the final mix based on the number of collisions between rock fragments as calculated by the physics engine. Also the distribution of the attenuation factors applied to the grains in the final mix is calculated based on the magnitude of impacts in the physics engine. The sounds of rain or running water can also be synthesized and controlled in a very flexible way when using granular synthesis.

1.2. About the proposed system

One reason why granular synthesis does not find widespread use in video games might be the necessity to record or synthesize appropriate grains. Recording can be problematic if the underlying single sound events are very quiet, like small rain droplets hitting the ground, or if they are hard to isolate, like resonating air bubbles in water. Synthesis can solve this problem in some cases, but deriving synthesis models to create appropriate grain sounds can be a time consuming process. In addition such models are often very hard to parametrize and control due to their level of abstraction.

In the system described in this paper grains are extracted automatically from short recorded samples of granular sounds to avoid the aforementioned issues. Using these grains, sounds that are very close to the original samples can be reproduced. To furnish sound designers with a straight-forward tool enabling them to design a variety of granular sounds, we combine automatic grain extrac-

tion with sound morphing. By morphing between the grains that have been extracted from two (or more) different recorded samples, sounds that lie perceptually at an arbitrary position between these samples can be created. Although this approach has the limitation that sound samples are needed to create sound - as opposed to a system where grains are synthesized - it has the big advantage that sound designers do not have to record grain sounds or deal with complicated grain synthesis models. Instead they can use existing recordings of granular sounds as a starting point for sound synthesis.

1.3. Related work

Several papers discuss automatic extraction of grains or other features from granular sounds.

Bascou and Pottier [6] automatically detect the distribution and pitch variation of grains in granular sounds. However, the grains they are working with are extracted manually. Their algorithm works by detecting pitch-shifted versions of these grains in the spectrogram of a signal.

Lee et al. [7] extract grains from audio signals based on sudden changes in the energy content of the spectrum, expressed as spectral flux between neighbouring frames. These grains are then used to resynthesize a signal in which the distribution of the extracted grains over time can be modified. To fill gaps between grains that are caused by a different distribution, grains are made longer by using linear prediction. Gaps can also be filled by concatenating similar grains. In their study grains do not overlap.

Schwarz et al. [8] extract grains from audio and arrange them in multidimensional space according to several features describing their tone color. Grains are extracted by splitting the input signal at parts that are silent or based on high-frequency content of the signal. The software tool CataRT [9], developed by the authors, allows users to generate sound by navigation through the tone color space, moving between groups of similar grains. This comes close to morphing between sounds, but requires users to provide appropriate grains for all needed tone colors.

Lu et al. [10] implement sound texture synthesis using granular synthesis. They extract grains from an input signal based on MFCC-similarity measurement between short frames. Grains are varied in various ways during resynthesis to create variations of the sound.

The extracted grains used by Lu et al. [10] are between 0.3 s and 1.2 s long and thus much longer than the grains used in our method. Also Fröjd and Horner [11] cut the input signal into grains ("blocks") with a comparatively large optimal length of 2 s. The advantage of long grains as used in these studies is that micro structures of the sound can be captured in the grains. However, such microstructures can not be changed during resynthesis.

1.4. Structure of this paper

The next section describes the technique used for automatic grain extraction. Section 3 gives an overview on how sound can be synthesized from the extracted grains. The method used for morphing between granular sounds is described in section 4. Results of synthesis and morphing are reported in the last section. A link to sound samples is provided.

2. AUTOMATIC GRAIN EXTRACTION

2.1. Noise subtraction

In granular sounds that consist of a very dense pattern of grains, most of these converge to noise and only the louder ones can be perceived separately. To improve the quality of the extracted grains, we first remove this noise by spectral subtraction (see [12] for a detailed introduction).

To determine the spectrum of the noise to be removed, the sample is first cut into overlapping frames. We use frames of 1024 samples that are extracted from the signal at increments of 32 samples. The reason for choosing a small increment is to maximize the number of extracted frames, since the loudest 85 % of all frames are rejected later. This was found to be a simple measure to reduce the number of frames containing loud and short bursts, which are typically found in granular sounds. A small overlap is also needed to ensure quality in the resynthesized signal after noise subtraction. After applying a Hamming window each frame is transformed to the frequency domain using a FFT of size N_{fft}, which is equal to the frame length. Then the amplitude spectrum of each frame is calculated. The spectra are smoothed by applying a FIR filter whose impulse response is a Gaussian calculated by the equation

$$h[n] = a \cdot e^{-\frac{(n-b)^2}{2c^2}}, \tag{1}$$

where $n = [0, 1, ..., 33]$, a is a normalization constant so that $\sum h[n] = 1$, $b = 16$ and $c = 3$.

The energy $e[m]$ of each frame m is calculated as

$$e[m] = \sum_{n=0}^{N_{\text{fft}}/2+1} S_a[m, n]^2, \tag{2}$$

where $S_a[m, n]$ is the amplitude of the frequency bin n in the smoothed spectrum of frame m.

To avoid loud sound events that distort the extracted noise spectrum, only the quietest 15 % of all frames are used to calculate the noise spectrum S_n, which is calculated as the average amplitude spectrum of these frames.

To reconstruct the signal with reduced noise, the noise spectrum S_n is first subtracted from the amplitude spectra S_a of each frame, setting all bins to zero where the amplitude of the noise spectrum is higher:

$$S'_a[m, n] = \begin{cases} S_a[m, n] - S_n[n], & \text{if } S_a[m, n] > S_n[n] \\ 0, & \text{if } S_a[m, n] \leq S_n[n] \end{cases} \tag{3}$$

Then the amplitude spectra with reduced noise S'_a are applied to the complex FFT coefficients $S[m, n]$ of each frame after normalizing them.

$$S_c[m, n] = S'_a[m, n] \frac{S[m, n]}{S_a[m, n]}, \text{where } n = 0, 1, ..., N_{\text{fft}}/2 \tag{4}$$

The second half of the FFT coefficients, i.e. the frequency bins $n = N_{\text{fft}}/2 + 1, ..., N - 1$ of all frames are obtained by mirroring the complex conjugate:

$$S_c[m, n] = S_c^*[m, N_{\text{fft}} - n], \text{where } n = \frac{N_{\text{fft}}}{2} + 1, ..., N_{\text{fft}} - 1 \tag{5}$$

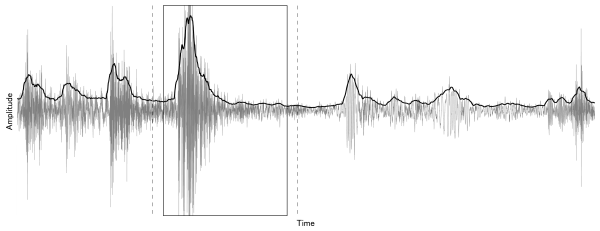


Figure 2: *Extraction of a grain from granular sound. The extracted grain (solid rectangle) is found between the boundaries of the window (dotted lines) placed around the maximum of the envelope (solid curve).*

The resulting complex spectra are then transformed to the time domain using the inverse FFT and overlapped according to the way the frames where extracted initially, which yields the signal with reduced noise $s_c[t]$.

The noise spectrum S_n is also used when resynthesizing the sound and when morphing between different sounds.

2.2. Grain extraction

The grains are extracted from the loudest parts of the signal $s_c[t]$. The loudest parts are found by looking at its smoothed envelope $g[t]$. The envelope $g'[t]$ is obtained using $\hat{s}_c[t]$, the Hilbert transform of the signal and its complex conjugate $\hat{s}_c^*[t]$:

$$g'[t] = \sqrt{\hat{s}_c[t] \cdot \hat{s}_c^*[t]} \tag{6}$$

The smoothed envelope $g[t]$ is obtained by applying a moving average of length 100.

As is shown in figure 2, the first grain is extracted from a window around the index τ, which is the position of the global maximum of the envelope. The index t_s of the first sample to be extracted and the index t_e of the last sample are determined by finding the minima of the envelope that lie within a certain window around τ. The window is defined by the variables w_s and w_e, which denote the maximum number of samples between τ and the start or the end of the window respectively. The grain is thus extracted between

$$t_s = \min(g[t] : \tau - w_s \leq t < \tau) \text{ and} \tag{7}$$

$$t_e = \min(g[t] : \tau < t \leq \tau + w_e). \tag{8}$$

The grain is stored in the grain waveform table but deleted from the signal $s_c[t]$ and the envelope $env[t]$ by setting both in the range $[t_s, t_e]$ to zero. After this deletion the same algorithm is reiterated to find the next grain. This is repeated until the user specified number of grains have been extracted or until the envelope is all zero.

To reduce unwanted noise during later synthesis, the start and end of all extracted grains are attenuated gradually so that jumps in the final mix are avoided. This is done by attenuating with the envelope a, which is defined as

$$a[t'] = \begin{cases} \sqrt[4]{\frac{t'}{\tau - t_s}}, & \text{for } t_s \leq t < \tau \\ \sqrt[4]{1 - \frac{t-\tau}{t_e - \tau}}, & \text{for } \tau \leq t \leq t_e \end{cases} \tag{9}$$

where $t' = t - t_s$.

3. SOUND SYNTHESIS USING EXTRACTED GRAINS

To synthesize sound of arbitrary length that is similar to the original sound from which noise and grains were extracted, we first synthesize noise according to the noise spectrum S_n, which was extracted as described in section 2.1. Then we add the extracted grains to the noise signal.

The noise is generated by shaping white noise according to the extracted noise spectrum S_n using multiplication of the spectra in the frequency domain. After generating a block of white noise of length $N_{\text{fft}}/2$, i.e. half the length of the FFT used when creating the noise spectrum to avoid aliasing, it is padded with zeros to form a block of N_{fft} samples and transformed to the frequency domain using the FFT, yielding the frequency domain white noise signal $R'[n]$. Since the time domain signal was real valued, $R'[n]$ is symmetric and the multiplication only has to be applied to the first half:

$$R[n] = S_n[n]R'[n], \text{where } 0 \leq n \leq N_{\text{fft}}/2 \tag{10}$$

The product of this multiplication is mirrored to form the second half of the frequency domain representation

$$R[n] = R^*[N_{\text{fft}} - n], \text{where } N_{\text{fft}}/2 < n < N_{\text{fft}} \tag{11}$$

The shaped noise is obtained by transforming $R[n]$ back to the time domain. To create longer noise signals, multiple white noise buffers of length $N_{\text{fft}}/2$ have to be processed in this way and overlapped with length $N_{\text{fft}}/2$ after transformation to the time domain.

After creating noise, the extracted grains are added. Adding the grains at the same positions from which they were extracted with their original amplitude will create a signal that is very close to the original - not only acoustically, but also with regard to the actual waveform. However, to synthesize a signal that sounds similar to the original the grains do not need to be distributed in the exact same way as they were in the original. Instead, grains are placed randomly in the synthesized sound.

Especially when working with few extracted grains, repetitiveness can be avoided when the amplitude of the grains in the synthesized signal is varied randomly. To create a sound that resembles the original this variation should follow the same loudness distribution as the grains in the original sound. This can be achieved by the following method: Before synthesis all grains are normalized, so that their amplitude is one, but their original amplitude is stored in a separate array. During synthesis each grain is attenuated with an amplitude that is randomly drawn from the array of amplitudes. Alternatively random numbers can be drawn from a probability distribution that matches the original grain amplitude loudness distribution. Depending on the nature of the sound big variations in the loudness of the grains might not be desirable, because the tone color of quiet and loud grains are different. In such cases small variation of the original amplitude can help to prevent repetitiveness in the synthesized sound.

However, since not all grains can be extracted during grain extraction, synthesis that is only based on the number of grains extracted per second and their loudness distribution can yield very unsatisfactory results. Modifying these parameters can increase the quality of the synthesized sound. This also gives sound designers more control over the produced sound. They can modify the sound by varying the loudness of the noise, the loudness distribution of

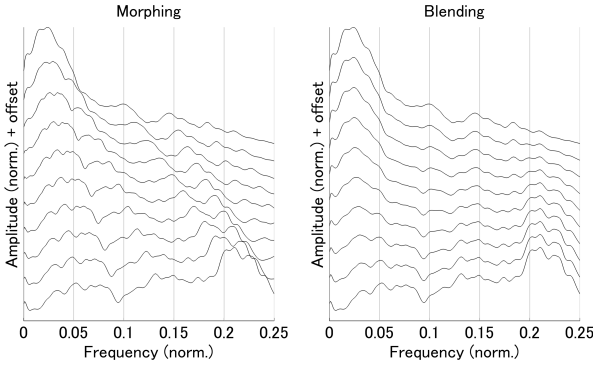


Figure 3: *Gradual morphing (left) and blending (right) between the two spectra at the top and bottom of the graphs. Note how formants move along the frequency axis when morphing.*

the grains and the number of grains per second. Finding appropriate values for these parameters is an important step in creating realistic sound.

Some granular sounds are the result of thousands of grains sounding at the same time. When synthesizing such sounds the noise component plays a significant role in recreating a realistic signal, but also grains should be layered on top of each other. The number of grains extracted per second only reflects the number of a few loud grains that were detected, but the number of grains actually sounding per second can be much higher. In such cases realism can be greatly increased by adding a higher number of grains to the final mix, attenuating grains with a loudness distribution that favours small amplitudes.

Section 5 presents some synthesis results along with the configurations used to create them.

4. MORPHING OF GRANULAR SOUNDS

4.1. Morphing vs. Blending

Noise and grains that are extracted from two different granular sounds can be combined in different ways. The straight-forward way of taking the (weighted) average of the noise spectra and using grains from both sounds will yield a sound that corresponds more to mixing both original sounds than to creating a sound whose tone color lies perceptually between the original sounds. This can be desirable when blending between one sound to the other sound.

However, to morph between the original sounds as described in section 1.2, i.e. to create a sound whose tone color lies between the original sounds, a different method must be used. One important manifestation of tone color is the shape of the spectrum of a sound because the distribution of energy over the frequency components of a sound plays an important part in tone color perception [13]. In the noise spectra in figure 3 the different energy distributions can clearly be seen. To gradually morph the tone color of one sound to the other sound it is necessary to move the formant(s) of one spectrum to the positions of the formant(s) in the other spectrum, also changing their shape gradually. Taking averages of both spectra with gradually changing weights, as shown in the right side of the figure, does not have this effect. Morphing between two spectra is shown in the left side of the figure.

The next sections describe the technique used for morphing and its application to granular sounds. This technique is applied to the noise and the grains extracted from the input sounds as described earlier.

4.2. Morphing of spectra

Shifting the formants of some spectrum A to the location of the formants of another spectrum B is essentially the same as redistributing the energy in spectrum A so that it resembles the energy distribution in spectrum B. The method used to achieve this was already introduced in an earlier publication [1], but for completeness we reproduce its explanation here.

We can express the energy distribution over the samples of an power spectrum by its integral. The integral of the power spectrum $s(\omega)$ where $\omega = [0; \Omega]$,

$$S(\omega) = \int_0^{\omega} s\left(\theta\right) d\theta, \tag{12}$$

expresses the energy between frequency ω and zero frequency. In other words, $S(\omega)$ is the cumulative energy of the spectrum.

We use the normalized cumulative energy of two spectra to match frequencies of the same cumulative energy and to find a spectrum with an intermediary energy distribution. To normalize we first remove any zero offset of the power spectrum

$$s_0(\omega) = s(\omega) - \min(s) \tag{13}$$

and then divide by the cumulative energy of s_0 at $\omega = \Omega$, which corresponds to the integral:

$$s_{\mathrm{norm}}\left(\omega\right) = s_0(\omega) / \int_0^{\Omega} s_0\left(\theta\right) d\theta \tag{14}$$

Then we calculate the integral of s_{norm}

$$S_{\mathrm{norm}}(\omega) = \int_0^{\omega} s_{\mathrm{norm}}\left(\theta\right) d\theta \tag{15}$$

whose maximum value $S_{\mathrm{norm}}(\Omega) = 1$ due to the normalization.

We do this for the two spectra $a(\omega)$ and $b(\omega)$ to get the normalized cumulative power spectra $A(\omega)$ and $B(\omega)$, keeping the normalization parameters

$$p_a = \min(a), \tag{16}$$

$$p_b = \min(b), \tag{17}$$

$$q_a = \int_0^{\Omega} a_0\left(\theta\right) d\theta \text{ and} \tag{18}$$

$$q_b = \int_0^{\Omega} b_0\left(\theta\right) d\theta. \tag{19}$$

We interpolate by first finding frequencies ω_a and ω_b in spectra A and B where the cumulative energy is equal to an arbitrary level y. In the interpolated spectrum, the frequency ω_{ab} where the cumulative energy reaches y should lie between ω_a and ω_b. We calculate this frequency as $\omega_{ab} = v \cdot \omega_a + (1 - v) \cdot \omega_b$ with v in

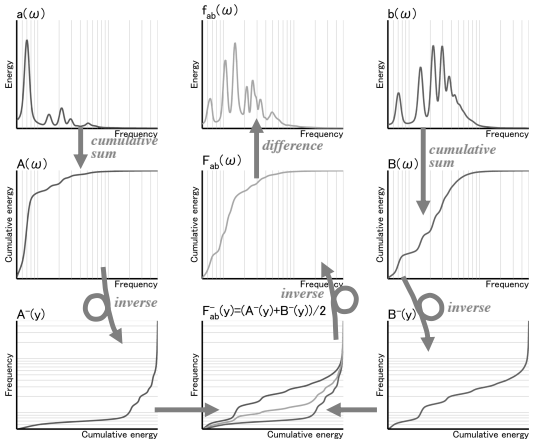


Figure 4: *Summary of the spectra morphing algorithm. Follow the arrows to read the figure. Given the two power spectra $a(\omega)$ and $b(\omega)$, their cumulative power spectra $A(\omega)$ and $B(\omega)$ are calculated. These are inverted to functions of cumulative energy, $A^-(y)$ and $B^-(y)$. Their average $X^-_{ab}(y)$ is inverted back to a function of frequency yielding the cumulative sum $X_{ab}(\omega)$, which is differentiated to get the morphed spectrum $x_{ab}(\omega)$.*

the range $[0; 1]$ expressing the desired similarity of the interpolated spectrum to A. This is the same as calculating the weighted average along the frequency axis between the cumulative energy curves of spectrum A and B. We therefore need to invert $A(\omega)$ and $B(\omega)$, which are functions of the frequency ω, to $A^-(y)$ and $B^-(y)$ which are functions of the cumulative energy y. We then calculate the weighted average of these inverses and get

$$F^-_{ab}(y) = v \cdot A^-(y) + (1 - v) \cdot B^-(y) \tag{20}$$

which is the inverse of the cumulative energy. Inverting and differentiating this function gives us the normalized interpolated power spectrum $f_{ab_{\mathrm{norm}}}$. To denormalize the spectrum we use the normalization parameters mentioned above and get

$$f_{ab} = (vq_a + (1 - v)q_b) \cdot X_{ab_{\mathrm{norm}}} + vp_a + (1 - v)p_b. \tag{21}$$

For implementation we need to consider discrete spectra. Supposing $f_{\mathrm{norm}}[n]$ is a discrete normalized power spectrum, the cumulative energy is calculated as the cumulative sum of its samples:

$$F_{\mathrm{norm}}[n] = \sum_{i=0}^{n} f_{\mathrm{norm}}[i] \tag{22}$$

To calculate its inverse we interpolate frequency values at arbitrary energy intervals. By varying the size of the interval we can control the quality of the output. After calculating the weighted average of the interpolated inversions we need to invert (interpolate) this average again to get the cumulative power spectrum as a discrete function of frequency. Calculating the difference between succeeding elements ($f[n] = F[n] - F[n-1]$) allows us to get the normalized interpolated power spectrum. After denormalizing in the same fashion as for continuous spectra, we get the interpolated power spectrum. The implementation is summarized in figure 4.

4.3. Morphing of noise spectra

The noise spectra of both sounds are morphed according to the algorithm described above. The resulting spectrum is used to shape the white noise as described in section 3.

4.4. Morphing of grains

4.4.1. Overview

The morphing of grains is not as straight forward as the morphing of the noise spectra: There are several grains for each sound, so before morphing it must be decided between which grains of sound A and which grains of sound B morphing should be conducted. Additionally grains are waveforms and cannot be represented by a single power spectrum, because temporal information of the signal would be lost.

The next paragraph describes how grains are first paired and then cut into frames and morphed.

4.4.2. Pairing grains

We want to morph between similar grains, so we need to find a way to measure the distance between two grains based on their similarity. This measurement is based on the spectral shape, which is one feature of the tone color. The distance is measured by the difference in the energy distribution between the frequency spectra of two grains. To calculate the difference between grains A and B, the frequency spectra are calculated over the whole length of both grains by first dividing the grains into overlapping frames of length $L = 256$ samples, extracted at increments of $d = 16$ samples to ensure a high time resolution, before transforming them to the frequency domain. The frames are padded with zeros to form blocks of length $N_{\mathrm{fft}} = 2L$ before applying the FFT. This extra space is needed to avoid aliasing caused by later multiplication in the frequency domain. Applying the FFT of length N_{fft} to frame m_{A} of grain A yields the complex FFT coefficients $X_{\mathrm{A}}[m_{\mathrm{A}}, n]$. From these coefficients power spectra are calculated for each frame using the formula

$$E_{\mathrm{A}}[m_{\mathrm{A}}, n] = |X_{\mathrm{A}}[m_{\mathrm{A}}, n]|^2, \text{ for } 0 \leq n \leq N_{\mathrm{fft}}/2, \tag{23}$$

where $E_{\mathrm{A}}[m_{\mathrm{A}}, n]$ are the resulting power spectra. The power spectrum $\overline{E_{\mathrm{A}}[n]}$ representing the whole grain is calculated by averaging all power spectra of the grain. The same calculations are executed for grain B.

The distance between both grains is calculated as the difference between the energy distribution of the power spectra $\overline{E_{\mathrm{A}}[n]}$ and $\overline{E_{\mathrm{B}}[n]}$. As in section 4.2, the energy distribution is expressed by the normalized cumulative sum of the power spectra

$$S_{\mathrm{A}}[n] = \sum_{i=0}^{n} \overline{E_{\mathrm{A}}}[i] / \sum \overline{E_{\mathrm{A}}}[i] \tag{24}$$

and $S_{\mathrm{B}}[n]$ which is calculated similarly. The difference in the energy distribution is measured by calculating the size of the area between the graphs of $S_{\mathrm{A}}[n]$ and $S_{\mathrm{B}}[n]$ as shown in figure 5.

Once the distances between all grains of sound A to all grains of sound B have been calculated, grains are paired using a variation of the stable marriage problem. Since the number of grains of both sounds is not necessarily equal, grains of the sound with more grains can be paired with more than one grain of the other

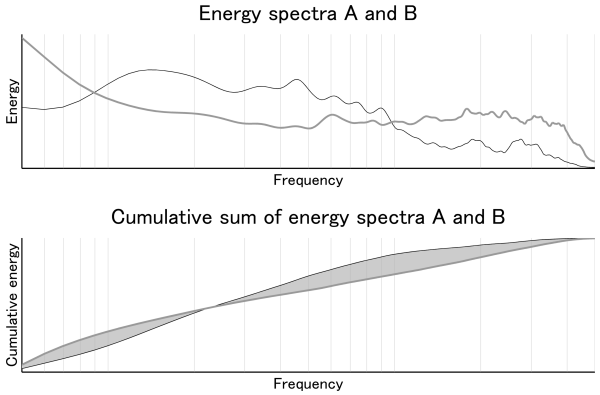


Figure 5: *The distance between the two spectra in the upper plot is measured by calculating the size of the area between the lines representing the cumulative sum of the spectra, as shown in the lower plot.*

sound. The following explanation uses the metaphor of companies (grains of the sound with more grains) and applicants (grains of the sound with less grains) to avoid venturing into the domain of polygamy. Two grains are considered to be a good match if their distance is small. Since the number of companies N_{co} and the number of applicants N_{ap} is not equal, every company has to hire $N = N_{\mathrm{co}}/N_{\mathrm{ap}}$ candidates on average so that all applicants are hired. The maximum number $N_{\max}$ of candidates a company can hire is fixed at the integer above N: $N_{\max} = \lceil N \rceil$.

Each iteration every applicant without a job offer applies at his most preferred company among the companies he has not yet applied at. After all applicants have filed applications, companies which have more than $N_{\max}$ applicants reject the worst matching applicants, keeping only $N_{\max}$ applications. These matches are tentative and can be rejected in the next iteration.

The globally worst matching applications of all companies are also rejected until the average number of applicants per company is equal to or lower than N. The algorithm is reiterated until all applicants are employed by a company.

Every grain in the applicant group thus has one match in the company group. Every grain in the company group has one or more matches in the applicant group, but is only paired with the best matching applicant.

4.4.3. Implementation of grain morphing

Once each grain of sound A is linked to a similar grain in sound B and vice-versa, morphing between two paired grains A and B can be considered.

As mentioned before, frames are extracted from the grains. Since the number of frames in grain A (M_{A}) and the number of frames in grain B (M_{B}) are not necessarily equal due to differing length of grains A and B, frames need to be aligned in a certain way. For the morphing of grains linear alignment yielded good results. The number of frames M_{AB} in the morphed sound is determined by $M_{\mathrm{AB}} = \mathrm{round}((1-v)M_{\mathrm{A}} + vM_{\mathrm{B}})$, where v is again the morphing factor (see section 4.2). The calculation of frame m of the morphed sound is based on frame m_{A} and m_{B} of sounds A and B respectively, which are chosen using the following formula:

$$m_{\mathrm{A}} = \mathrm{round}(M_{\mathrm{A}} \cdot m/M_{\mathrm{AB}}) \tag{25}$$

$$m_{\mathrm{B}} = \mathrm{round}(M_{\mathrm{B}} \cdot m/M_{\mathrm{AB}}) \tag{26}$$

Morphing is conducted between the power spectra $E_{\mathrm{A}}[m_{\mathrm{A}}, n]$ and $E_{\mathrm{B}}[m_{\mathrm{B}}, n]$ of the aligned frames m_{A} and m_{B}, which results in the morphed power spectrum E_{AB}. Since power spectra do not contain information about the phase of the signal, this information has to be extracted from the FFT coefficients $X_{\mathrm{A}}[m_{\mathrm{A}}, n]$ and $X_{\mathrm{B}}[m_{\mathrm{B}}, n]$ which were calculated earlier for frames m_{A} and m_{B} respectively. The phase is retained in the normalized complex spectra C_{A} and C_{B} which are calculated from the complex FFT coefficients. The formula to calculate the normalized complex spectrum C_{A} is

$$C_{\mathrm{A}}[m_{\mathrm{A}}, n] = \frac{S_{\mathrm{A}}[m_{\mathrm{A}}, n]}{|S_{\mathrm{A}}[m_{\mathrm{A}}, n]|}, \text{ where } 0 \le n \le N_{\mathrm{fft}}/2. \tag{27}$$

The spectrum C_{B} is calculated in the same way.

The morphed power spectrum E_{AB} is then applied to the complex spectra C_{A} and C_{B} of the aligned frames of both sounds to calculate the morphed spectrum S_{AB} of frame m, which is a weighted average of both sounds:

$$\begin{aligned} S_{\mathrm{AB}}[m, n] = (1-v) \cdot C_{\mathrm{A}}[m_{\mathrm{A}}, n] \cdot \sqrt{E_{\mathrm{AB}}[m, n]} \\ + v \cdot C_{\mathrm{B}}[m_{\mathrm{B}}, n] \cdot \sqrt{E_{\mathrm{AB}}[m, n]}, \end{aligned} \tag{28}$$

where $0 \le n \le N_{\mathrm{fft}}/2$.

The morphed spectrum is mirrored to obtain a complete set of FFT coefficients:

$$S_{\mathrm{AB}}[m, n] = S^*_{\mathrm{AB}}[m, N_{\mathrm{fft}} - n], \text{where } N_{\mathrm{fft}}/2 < n < N_{\mathrm{fft}} \tag{29}$$

Then it is transformed to the time domain with the inverse FFT to obtain a time domain signal representation of the morphed frame. Finally, to form the final output of the morph, the frames are overlapped with spacing their start points at intervals of d samples, which is the same spacing used during frame extraction. When overlapping the frames a window function can be applied. A Hann window of length $N_{\mathrm{fft}}/2$ followed by $N_{\mathrm{fft}}/2$ zeros - by which the second half of the signal is discarded - yielded good results.

4.5. Real-time implementation

Although sound morphing can easily be implemented in real-time using the method described above (see [1] for further details), the high number of grains (around 200-500 per sound) make real-time implementation very difficult for synthesized granular sounds when grain morphing is conducted at run-time. This is why morphing between grains is executed before run-time instead. To enable users to create a sound with a tone color that lies at an arbitrary position between sound A and B, or - in other words - to create a sound corresponding to an arbitrary value of $v \in [0; 1]$ (as introduced in section 4.2), several sets of grain morphs for several values of v are prepared. The higher the number of grain morph sets, the higher is the smoothness of the morph between the two granular sounds. To implement a system having a resolution of ten sets, eight grain morphs with $v = 0.1; 0.2...0.9$ can be created and stored in memory in addition to the original grains of sounds A and B corresponding to $v = 0$ and $v = 1$ respectively. At runtime grains are chosen randomly from the sets closest to a given v value. The morphing of a single spectrum, however, is very cheap, so morphing of the noise can be done in real-time.

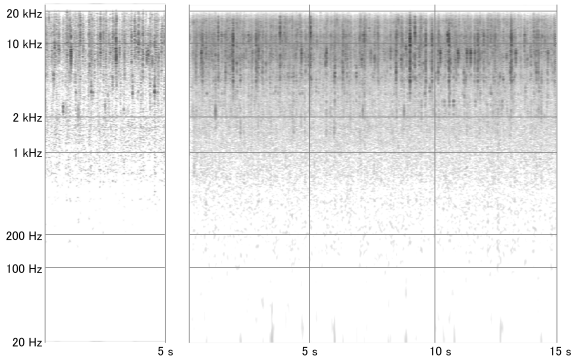


Figure 6: *Spectrogram of recording of rain sound (left) and synthesized rain sound (right).*

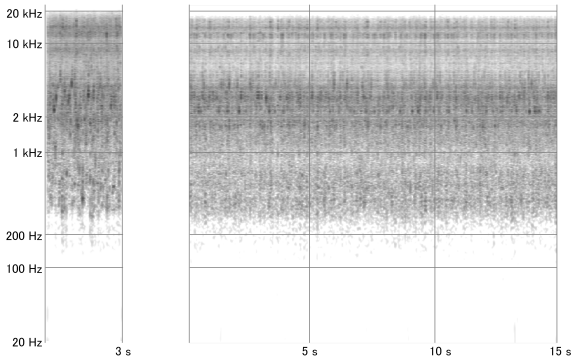


Figure 7: *Spectrogram of recording of water sound (left) and synthesized water sound (right).*

5. RESULTS

5.1. Synthesis

Realistic granular sounds that bear all the characteristics of the originals can be synthesized by extracting grains and noise from short samples of only a few seconds length. The sounds shown in figures 6 and 7 were created by extracting grains and noise from sample recordings of 3 seconds (water) or 5 seconds (rain).

For both sounds grains were normalized after extraction and attenuated by a random factor during synthesis. The random attenuation factors were drawn from a normal distribution with $\mu = 0$ and $\sigma = 3$, which also yielded negative attenuation factors. This increased the variation of the signal, since the amplitude of some grains was inverted.

To recreate a sound close to the original, the number of grains per second was set to 100, which corresponded approximately to the number of grains that was extracted from one second of the original sounds.

5.2. Morphing

The proposed algorithm works well for morphing between different sounds of rain or running water and yields realistic sounding results even when grains are extracted from only short sounds of only a few seconds length.

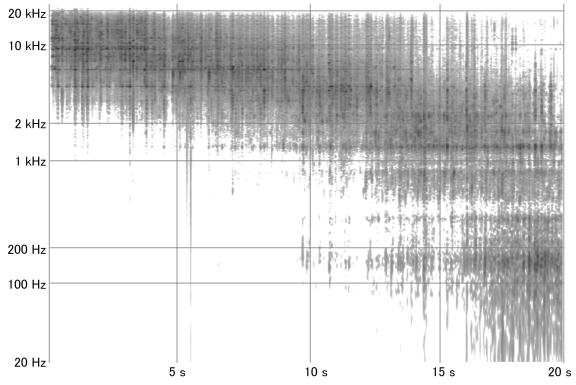


Figure 8: *Spectrogram of two sounds being morphed. Note how the formant slides between the positions in both sounds.*

Figure 8 shows the morph between two sounds synthesized from a recording of glass shards falling on a hard surface and a bubbling thick liquid. The spectral energy distribution is gradually changing between both sounds.

Morphing between very different sounds, like rain and fireworks, does not give very realistic results. However, this does not necessarily highlight a flaw in the proposed method, since such sounds do not exist in nature either.

5.3. Sound samples

Sound samples for synthesized sounds and morphed sounds can be found online[1].

6. FUTURE WORK

Apart from some necessary quality improvements in the grain extraction algorithm, there is much scope for enhancing the extraction of other features of the granular source sounds. These include the actual number of grains per second, the amplitude distribution of the grains or the temporal distribution of grains.

To consider granular sounds in which features change with time, like water splashes or breaking objects, temporal changes in the extracted parameters also need to be extracted.

7. REFERENCES

[1] Sadjad Siddiq, ``Morphing of impact sounds," in *Proceedings of the 139th Audio Engineering Society Convention*. Audio Engineering Society, 2015, to be published.

[2] William Moss, Hengchin Yeh, Jeong-Mo Hong, Ming C Lin, and Dinesh Manocha, ``Sounding liquids: Automatic sound synthesis from fluid simulation," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 3, pp. 21, 2010.

[1] See http://www.jp.square-enix.com/info/library/private/granularMorphing.zip. The password of the zip-file is "gmorph2015".

[3] Sadjad Siddiq, Taniyama Hikaru, and Hirose Yuki, ``当たって砕けろッ！プロシージャルオディ制作 (Go for broke! Creation of procedural audio content),'' Presentation at the Computer Entertainment Developers Conference, 2014, Slides and sound samples are available at `http://connect.jp.square-enix.com/?p=2639` (visited on 25.9.2015).

[4] Curtis Roads, *Microsound*, MIT press, 2004.

[5] Øyvind Brandtsegg, Sigurd Saue, and Thom JOHANSEN, ``Particle synthesis--a unified model for granular synthesis,'' in *Proceedings of the 2011 Linux Audio Conference. (LAC'11)*, 2011.

[6] Charles Bascou and Laurent Pottier, ``New sound decomposition method applied to granular synthesis,'' *in ICMC Proceedings*, 2005.

[7] Jung-Suk Lee, François Thibault, Philippe Depalle, and Gary P Scavone, ``Granular analysis/synthesis for simple and robust transformations of complex sounds,'' in *Audio Engineering Society Conference: 49th International Conference: Audio for Games*. Audio Engineering Society, 2013.

[8] Diemo Schwarz, Roland Cahen, and Sam Britton, ``Principles and applications of interactive corpus-based concatenative synthesis,'' *Journées d'Informatique Musicale (JIM), GMEA, Albi, France*, 2008.

[9] Diemo Schwarz, Grégory Beller, Bruno Verbrugghe, Sam Britton, et al., ``Real-time corpus-based concatenative synthesis with catart,'' in *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx), Montreal, Canada*. Citeseer, 2006, pp. 279--282.

[10] Lie Lu, Liu Wenyin, and Hong-Jiang Zhang, ``Audio textures: Theory and applications,'' *Speech and Audio Processing, IEEE Transactions on*, vol. 12, no. 2, pp. 156--167, 2004.

[11] Martin Fröjd and Andrew Horner, ``Sound texture synthesis using an overlap-add/granular synthesis approach,'' *J. Audio Eng. Soc*, vol. 57, no. 1/2, pp. 29--37, 2009.

[12] Saeed V Vaseghi, *Advanced digital signal processing and noise reduction*, John Wiley & Sons, 2008.

[13] Hermann Ludwig Ferdinand von Helmholtz, *Die Lehre von den Tonempfindungen als physiologische Grundlage für die Musik*, Vieweg, 1863.

REVERBERATION STILL IN BUSINESS: THICKENING AND PROPAGATING MICRO-TEXTURES IN PHYSICS-BASED SOUND MODELING

Davide Rocchesso roc@iuav.it

Stefano Baldan stefanobaldan@iuav.it

Stefano Delle Monache sdellemonache@iuav.it

Iuav University of Venice, Venice, Italy

ABSTRACT

Artificial reverberation is usually introduced, as a digital audio effect, to give a sense of enclosing architectural space. In this paper we argue about the effectiveness and usefulness of diffusive reverberators in physically-inspired sound synthesis. Examples are given for the synthesis of textural sounds, as they emerge from solid mechanical interactions, as well as from aerodynamic and liquid phenomena.

1. INTRODUCTION

Artificial reverberation has always been part of the core business of digital audio effects [1, 2]. Its main purpose is that of giving ambience to dry sounds, mimicking propagation, absorption, and diffusion phenomena, as they are found in three-dimensional enclosures, at the architectural scale.

Ideally, artificial reverberators are linear time-invariant systems whose impulse response looks and sounds like a decaying noise. In the response of a real room, the early pulses correspond to the early reflections coming from the walls, and the density of pulses rapidly increases in time as a result of multiple reflections and scattering processes. It is often assumed that late reverberation is ideally represented as an exponentially-decaying Gaussian process [3, 4]. Essentially, a good reverb creates a kaleidoscopic and seemingly random multiplication of incoming pulses.

Feedback delay networks [4, 5] (FDN) are often used as the central constituent of reverberators, because they are efficient and their stability can be accurately controlled. FDNs can be parameterized according to reference room geometries [6, 7] or to recorded impulse responses [8], but they are also quite usable as instrument resonators or time-varying modulation effects [9].

In the Ball-within-the-Box model [6] the normal modes of a rectangular room are related to geometrical directions of standing waves, and diffusion is treated as a continuous transfer of energy between harmonic modal series, by means of a single scattering object represented by the feedback matrix of a FDN.

In this paper we propose the use of reverberation units, namely FDNs, as constituents of physics-based sound synthesis models, whenever the goal is that of thickening the distribution of elementary events occurring in mechanical interactions, or to give account of scattering and propagation phenomena.

In fact, reverberation phenomena do not occur only in air at the architectural scale. As it is obviously deduced from the historical success of spring and plate reverb units, vibration in solids can have a clear reverberation character [10].

The textural character of many everyday sounds is indeed determined by dense repetitions of basic acoustic events, which can be assimilated to reverberation in a wide sense. The key for simulating reverberation phenomena is to achieve a high event density, or echo density in reverberation terms. Abel and Huang [11] proposed a robust measure, called normalized echo density (NED), that can be used to characterize reverberant responses. Such measure can reveal the buildup of echoes at various levels of diffusion for a reverberation system. They also showed that NED is a good predictor of texture perception, regardless of the bandwidth of each single echo (or event) [12].

Section 2 recalls the structure of a FDN and illustrates the realization considered in this paper. Section 3 explains how reverberation is used in the context of solid interaction synthesis, namely to differentiate between scraping and rubbing. Section 4 shows how reverberation is used for the simulation of some aerodynamic phenomena. Section 5 points to uses of diffuse reverb for the synthesis of massive liquid sounds.

2. THE CORE COMPONENT

A FDN is described by the following equations:

$$\begin{aligned} y(n) &= \sum_{i=1}^{N} c_i s_i(n) + dx(n) \\ s_i(n+m_i) &= \sum_{j=1}^{N} a_{i,j} s_j(n) + b_i x(n) \end{aligned} \tag{1}$$

where $s_i(n)$, $1 \leq i \leq N$, are the outputs of a set of N delay lines at discrete time n, and x and y are respectively the input and output signal. $a_{i,j}$, b_i, c_i and d are real numbers, acting as weighting and recombining coefficients.

The diffusive behavior of FDN reverberators is determined by the feedback matrix $\mathbf{A} = [a_{i,j}]_{N \times N}$. To ensure that the diffusion process preserves energy, such matrix should be lossless [5]. To speedup convergence towards a Gaussian distribution of echoes, all coefficients of $\mathbf{A}$ should have the same magnitude [4]. A third

requirement, especially for large matrices, is efficiency, i.e., the possibility to have sub-quadratic complexity for matrix-vector multiplies. Some lossless, maximally-diffusive, and efficient matrices, such as Hadamard matrices, have been proposed in the literature [4, 5].

In this paper we consider a realization having the three properties of energy preservation, equal-magnitude coefficients, and efficiency, which is based on a circulant feedback matrix defined from a Galois sequence [13]. Circulant matrices afford matrix-vector multiplies in $O(N \log N)$ time by means of FFT or, alternatively, they admit a particularly simple implementation of such multiplies, whose parallelization is straightforward.

The sample-by-sample computation of the reverberator based on a 15×15 circulant matrix [13] can be organized as follows:

```
double SDTReverb_dsp(SDTReverb *x, double in) {
  double a, b, c, d, *s, out;
  int i;

  out = 0.0;

  for (i = 0; i < 15; i++) {
    s = &x->v[i];
    b = s[1] + s[2] + s[3] + s[5] +
        s[6] + s[9] + s[11];
    c = s[0] + s[4] + s[7] + s[8] + s[10] +
        s[12] + s[13] + s[14];
    a = 0.25 * (b - c);
    d = SDTDelay_dsp(x->delays[i], in + a);
    x->v[i] = x->g[i] *
              SDTOnePole_dsp(x->filters[i], d);
    out += x->v[i];
  }
  memcpy(&x->v[15], x->v, 14 * sizeof(double));
  return out / 15.0;
}
```

The main artifice for such a compact code is the juxtaposition of two copies of the outputs of the delay lines (`memcpy` operation), which allows the exploitation of the circulant structure as 15 independent iterations of a `for` loop, which could be efficiently parallelized. The proposed implementation is actually included as a Cycling'74 Max external in the Sound Design Toolkit (SDT) [14], a collection of physics-based sound models for the aural rendering of basic acoustic phenomena, such as contacts between solids, liquid and aerodynamic interactions[1]. `SDTReverb_dsp()` uses the SDT implementation of delay lines and one-pole IIR filters for frequency-dependent damping.

This maximally diffusive yet efficient FDN [13] takes six arguments as input parameters: the size of the room along the x, y and z axes (l_x, l_y, and l_z), a geometric randomness coefficient (between 0 and 1), the global reverberation time and the reverberation time at 1 kHz.

Virtual room dimensions are used to compute the lengths of the delay lines, which play a key role in the system response as they represent the fundamental periods at which the virtual environment resonates. In our implementation, delay times are computed to simulate the bouncing period of stationary plane waves in a rectangular room [6].

Each delay time is the reciprocal of

$$f = \frac{c}{2}\left[\left(\frac{n_x}{l_x}\right)^2 + \left(\frac{n_y}{l_y}\right)^2 + \left(\frac{n_z}{l_z}\right)^2\right]^{\frac{1}{2}}, \qquad (2)$$

where c is the speed of sound in the medium, and the triplets $[n_x, n_y, n_z]$ belong to the set

1,0,0	2,1,0	1,1,0	1,2,0	0,1,0
0,2,1	0,1,1	0,1,2	0,0,1	1,0,2
1,0,1	1,1,1	1,2,1	2,1,1	2,0,1.

Perfect rectangular enclosures provide a good reference model to distribute echoes in time. However, a subtle artifact called sweeping echo can arise from rigid geometrical specifications [15]. That is why, in our realization, a randomness coefficient is used to add some irregularity to the delay times, thus simulating slight deviations from a strict rectangular reverberation box.

Delay lines are implemented as follows [16]: A single circular buffer is swept by two allpass interpolated readers, updated and crossfaded at a 16 sample alternation rate. This arrangement allows to handle fractional as well as continuously varying delay times, preserving intonation accuracy and avoiding audible glitches in the feedback loop.

As the FDN matrix is lossless, to achieve a finite reverberation time the recirculating signal is multiplied by an attenuation coefficient g_i before entering the delay lines, yielding an exponential decay. To achieve a -60 dB attenuation ($\frac{1}{1000}$ of the initial amplitude) on a delay line of period τ at a given reverberation time T, the attenuation coefficient is computed as follows:

$$g_i = 10^{\frac{-3\tau}{T}}. \qquad (3)$$

Frequency-dependent attenuation is achieved by applying a simple one-pole lowpass filter at the output of each delay line, implemented by the difference equation

$$y(n) = (1 + a)x(n) - ay(n - 1). \qquad (4)$$

The cutoff frequencies are computed in order to obtain a filter attenuation of 60 dB at 1 kHz after a given time T'. Remembering to take into account the frequency independent attenuation coefficient, using a sampling period t_s the filter response at 1 kHz must be

$$g_\omega = \frac{10^{\frac{-3\tau}{T'}}}{g_i} = \frac{|1 + a|}{\sqrt{a^2 + 2a\cos(2000\pi t_s) + 1}}. \qquad (5)$$

Solving this quadratic equation gives two possible values of a, but only one solution preserves the stability of the lowpass filter ($|a|$ <= 1). That value is therefore the desired feedback coefficient.

2.1. Echo buildup time

The FDN multiplies elementary acoustic events by recirculating them through a set of delay lines. The density of "echoes" increases in time, with a speed that depends on delay line lengths which, in turn, depend on the size of the virtual resonator that the FDN is modeling. To quantify and represent the rapidity of echo buildup we use the NED measure [11], which is based on a sliding window and counts the number of impulse response taps which lie outside the standard deviation of the windowed samples, normalized to give 1 for a Gaussian distribution of values. In our

[1] The Sound Design Toolkit and its source code are available at `https://github.com/skat-vg/sdt`

implementation of NED we used a 20 ms Hanning window, gradually shrinking to 10 ms at the beginning of the impulse response. Figure 1 shows the buildup of the normalized echo density for a small (0.1 m), a medium (1.0 m), and a large (10.0 m) box. While for small and medium size boxes the buildup of a Gaussian process is practically instantaneous, it is clear how that takes over a hundred milliseconds in the case of boxes at architectural scale. This kind of diffuse reverb inevitably adds spaciousness and depth to any texture it will be applied to. Conversely, a resonator about 1 m in size starts promptly with a high echo density, and stabilizes around a NED value of 1 after about 10 ms. A small box also starts with a high echo density, but the tail of its response tends to ring, and that explains the dip in the dotted line of figure 1.

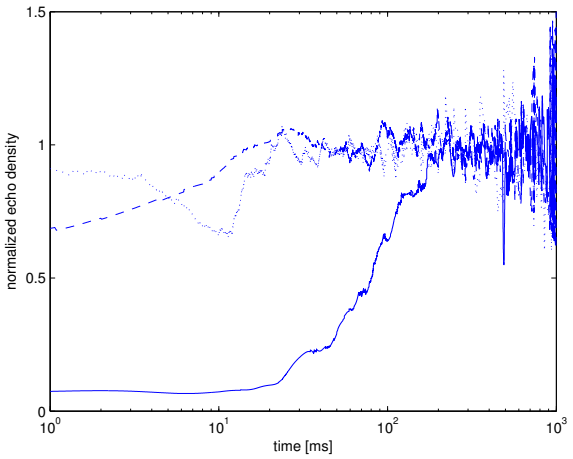


Figure 1: *Normalized echo density buildup for three maximally diffusive FDNs, modeling cubes of size* 0.1 m *(dotted),* 1.0 m *(dashed), and* 10.0 m *(solid), with maximal value for the randomness parameter.*

3. USE 1: SCRAPING/RUBBING

To produce sliding noises, Van Den Doel [17] proposed to bandpass filter a fractal noise, at a central frequency proportional to the velocity between the two surfaces in contact. The rugosity of the surface could be controlled by means of the fractal dimension. Similarly, Conan et al. proposed lowpass filtering a noise, with a biquad cutoff frequency proportional to velocity [18]. Since they showed that the density of impacts is the main discriminant for the perceptual distinction between rubbing and scraping, they proposed using a pulse-based noise generator: Pulses are generated by a Bernoulli process, and their temporal density can be controlled. The amplitude of pulses has a uniform distribution.

To realize a scraping/rubbing sound generator we take a different approach: We consider the rugosity of a surface, imported as an audio file, and we explicitly model an object sliding on it. The thickness of such object determines the number of impact points, the larger the object the higher the number of contacts. However, to increase the temporal density of micro-events further it is necessary to thicken the pattern of impacts, and a maximally-diffusive reverberator is the key component to increase the impact density.

The scraping model belongs to the set of physics-based algorithms available in the SDT. In the scraping model an audio signal is interpreted as surface profile in order to modulate the collisions between a point-mass exciter and a resonating object [19]. The sound model is implemented as Cycling'74 Max patch, shown in Figure 2. An arbitrary audio signal is acquired in order to provide a roughness profile to drive the sound synthesis (e.g., a sawtooth waveform at 50 Hz is displayed in the figure). The signal buffer length is 1000 ms, ideally corresponding to a 1000 mm-long surface. The "sliding parameters" layer is used to interpret the stored surface profile and drive the impact model accordingly. The vertical penetration of the probe sets the threshold level of the roughness profile above which the signal is detected, while the probe width parameter sets the size of the sliding window on the roughness profile (in mm, large = rubber, small = sharp object). The virtual probe is advanced every Δt ms by a distance $\Delta x = v\Delta t$, where v is the sliding velocity in m/s. The "velocity profile" box allows to draw velocity trajectories, that is describing the temporal unfolding of specific gestures, as in sawing, filing, scratching, and so forth. Additional parameters are Δt in ms and the diameter of a single contact area in cm. The sound quality of the single impact is described in the two boxes at the bottom of the GUI displayed in figure 2 (i.e., stiffness, contact shape, energy dissipation affecting the occurrence of bouncing phenomena, and modes of resonance), in order to characterize the scraping/rubbing on the surface profile with auditory impressions of different materials (e.g., metal, wood, plastic, glass, etc.)[2].

Figure 3 shows the spectrograms of the sounds produced by 1-second sliding gestures at constant speed, with a sharp (left, top) or with a wide (right, top) probe. The supporting surface is a sawtooth wave similar to the one shown in figure 2. The wide probe, as compared to the sharp probe, hits the surface asperities at many more points, thus producing a denser distribution of elementary impact noises. This multiplication of acoustic events is similar to early echoes in room reverberation. To increase the event density further we introduce the FDN reverberator, whose effects as a small or large square box (with maximum randomness), and as a damped or reflective enclosure, are also illustrated in figure 3. It is clear that both the enlargement of the probe and the introduction of diffusive reverberation increase the density of micro-impacts, and this adds a degree of freedom for the sound designer. Similarly, in classic reverb design one must decide how to split memory and operations between early reflections (simulated by a FIR structure) and diffuse reverberation (FDN) [20].

Beside increasing the temporal thickening of the sound texture, the introduction of a diffusive reverberator affects other timbral aspects. For this purpose, we conditioned the reverb parameters in order to reduce the occurrence of evident timbral effects, trying to maximize the multiplication of micro-impacts. Based on some informal listening tests, we associated the probe width $(1.0 - 100.0\,\mathrm{mm})$ with the global reverb time in the range of $0.2 - 3.0$ s, while keeping the reverb time at 1 kHz shorter (a 0.4 factor of the global reverb time). Different values of FDN damping give different impressions of surface material, being it metallic for low damping, or wood-like for high damping. The velocity $(0.0 - 1.0\,\mathrm{m/s})$ is inversely associated to the room size, in the range of $1.3 - 0.3$ m, thus producing the compression of the room size for high velocity. While poorly justified by the physics, we

[2]Audio examples of various gestures and materials are available at `https://soundcloud.com/skat-vg/sets/dafx2015-audio-examples`

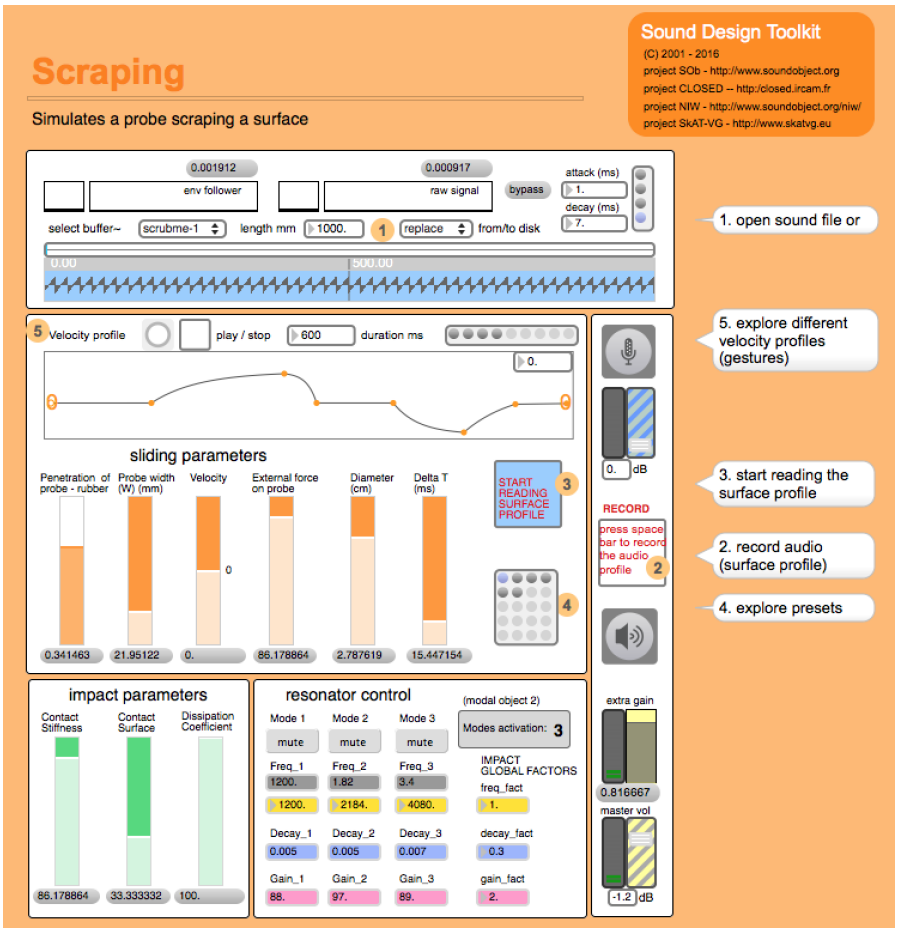


Figure 2: *Cycling'74 Max GUI of the scraping model.*

found this control strategy suitable to emphasize the typical glissando effect which occurs in quick sliding gestures and in back and forth motions.

Playing with the three size parameters of the FDN produces timbral effects, which can give the impression of extension and distance of the surface. For instance a texture of sparse, sharp micro-impacts with a room size of 14 m, a global time reverb of 14 s, and a damping at 5.6 s can easily convey the impression of a heavy rain on roofing iron sheets (see footnote 2).

4. USE 2: EXPLOSIONS

Powerful explosions, as well as objects traveling at supersonic speed such as rifle bullets or cracking whip tails, create shock waves, namely a sudden peak in pressure followed by a negative expansion tail. The algorithms in the Sound Design Toolkit simulate this event with a Friedlander waveform [21], which approximates the pressure change caused by an exploding point source emitting a spherical shock wave. The air then flows back to restore atmospheric pressure, generating a blast wind. This air flow is rendered by bandpass filtered white noise, modulated in amplitude by the Friedlander waveform as the wind intensity follows more or less the profile of the initial shock wave.

Real world explosions, however, are almost never perfectly impulsive. When happening in air, the initial shockwave is likely to generate some chaotic turbulence as it propagates. The blast can also get reflected by the ground or other obstacles, generating Mach stems and other kinds of interference. If the explosion transfers part of its energy to a solid object, such as the ground or the body of a rifle, vibrations propagate also through the solid. If the material is of non-uniform density, (e.g. rocks, gravel or soil), the wave is subject to different propagation speeds, reflections and refractions. Some explosions, then, cannot be approximated by a point source emitting spherical waves: Lightning bolts, for example, generate a simultaneous cylindrical shockwave across their length, and different wave sections interact as they meet because of the bolt tortuosity. These interactions create complex temporal patterns and have a direct effect on the resulting sound [22, 23].

The explosion model implemented in the Sound Design Toolkit exploits our maximally diffusive FDN to simulate scattering, diffusion, interferences and other kinds of interaction caused by the phenomena described above, adding complexity to the initial blast wave and improving the realism of the acoustic result. Figure 4 displays the block diagram of the whole explosion synthesis model.

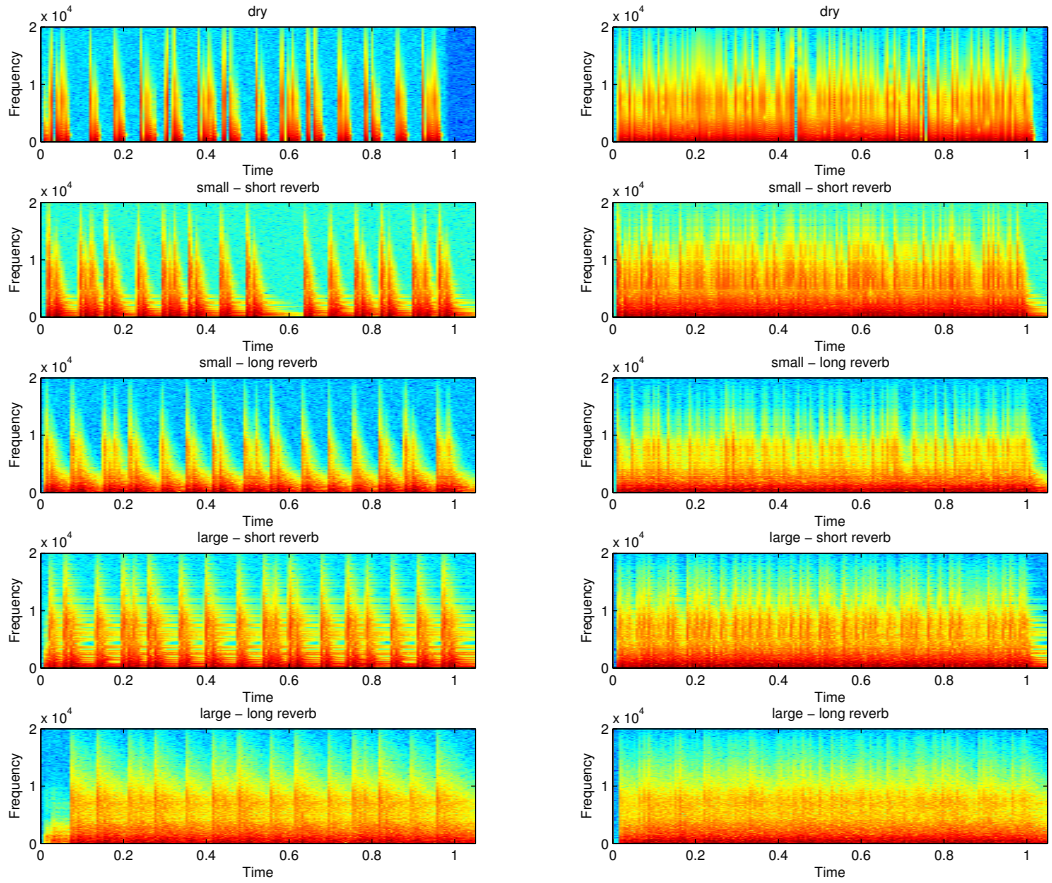


Figure 3: *Spectrograms resulting from the exploration at constant speed of a surface sawtooth texture, with a sharp (*$0.002\,\mathrm{m}$*, left) or wide (*$0.020\,\mathrm{m}$*, right) probe. Multiplication of impacts is obtained by means of a maximally-diffusive reverberator corresponding to a small (*$0.1\,\mathrm{m}$*) or large (*$1.0\,\mathrm{m}$*) resonator, with short (*$0.2\,\mathrm{s}$*, highly damped) or long (*$3.0\,\mathrm{s}$*, slightly damped) reverberation tail.*

The Friedlander waveform is initially scattered by the artificial reverb, and subsequently processed by a lowpass filter. The cutoff frequency of the filter is inversely proportional to the square root of the listening point distance, to simulate frequency dependent attenuation caused by the impedance of the medium in which acoustic waves propagate. The resulting waveform as it is represents the explosion shock wave, but it is also applied as an amplitude envelope to a band limited noise generator to simulate the blast wind. These two outputs are finally sent into two independent delay lines, to model a different propagation velocity for each component.

The reverberation algorithm is applied just after the Friedlander wave generator, with all of its parameters bound to a single free variable, namely the duration of the audible diffusive tail. By empirical trial and error, we have found that acoustically convincing results can be achieved adjusting the length of the delay lines so that an incoming shockwave is recirculated about a hundred times on average before becoming inaudible. For example, a diffusive tail of 1 second would require delay lengths to vary around an average value of 10 milliseconds. Moreover, higher frequencies are slightly damped by setting the reverberation time at $1\,\mathrm{kHz}$ to 90% of the whole reverberation time.

5. USE 3: SPRAYS AND SPLASHES

Liquids are involved in a great number of different and very complex sound events, such as dripping, splashing, burbling, pouring, fizzling and so on. However, the main source of these sounds is not the liquid in itself but rather its population of bubbles, trapped by its movement or generated by cavitation phenomena. The pressure of the surrounding liquid mass applies energy on the gas, transforming the bubble in a pulsating oscillator and quickly converting it to a spherical form. Spherical bubbles have a very well known

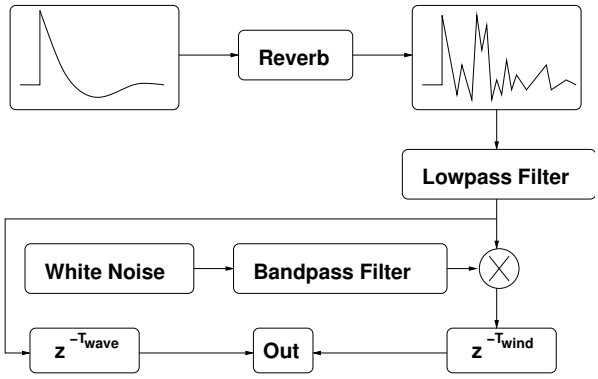


Figure 4: *Block diagram of the explosion sound synthesis model as implemented in the Sound Design Toolkit. The reverberator is cascaded to the Friedlander waveform generator to simulate scattering and diffusive phenomena in the wave propagation.*

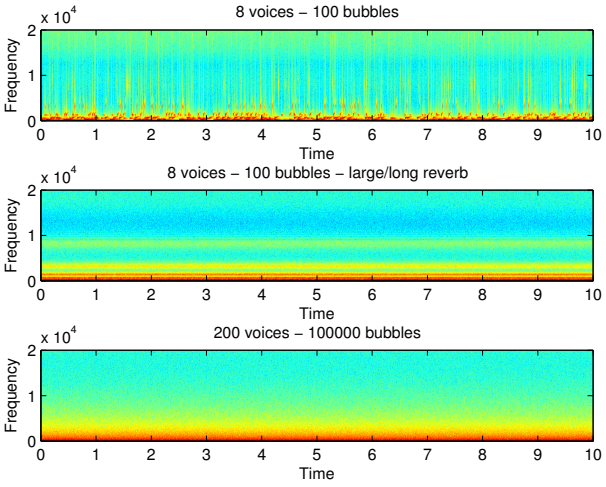


Figure 5: *Spectrograms of the sounds produced by a relatively small number of bubbles, without (top) and with (middle) reverb. The spectrogram at the bottom is obtained by a massive generation of bubbles.*

acoustic behavior [24], allowing to devise a physically informed approach for the synthesis of liquid sounds. A single bubble can be modeled by a simple exponentially decaying sinusoidal oscillator, whose amplitude and frequency envelopes are strictly dependent on the bubble radius and depth. The Sound Design Toolkit renders liquid sounds through additive synthesis, by means of a polyphonic sinusoidal oscillator bank driven by a stochastic model for bubble generation [25].

One of the main drawbacks of this method is that it requires a high degree of polyphony to produce convincing simulations, especially for very dense textures like rainstorms or roaring waterfalls, or bursting events like water splash crowns. If too many bubbles are generated compared to the available number of voices in the oscillator bank, amplitude envelopes get updated so often that the modulated sinusoids become almost stationary, resulting in a ringing rather than bubbling sound. On the other side, increasing the number of voices leads to a greater computational load. Artificial reverberation can be used as a cheaper alternative to generate dense sound events, replicating and diffusing the sound of each single bubble instead of generating more bubbles through actual polyphony.

To demonstrate the use of our FDN reverberation algorithm for this particular purpose, a very dense waterfall sound simulation was produced. The fluid flow synthesis model of the Sound Design Toolkit was configured with the highest level of polyphony achievable in real time on the target machine: 100000 bubbles per second distributed across 200 simultaneous voices (figure 5, bottom). Polyphony and event density was then reduced to 1000 bubbles per second on 32 voices and 100 bubbles per second on 8 voices (figure 5, top), generating sparser textures more similar to a small stream or to a gentle dripping (see footnote 2). Finally, artificial reverberation was applied to the latter sounds, using long decay times (10 seconds or more) and large room sizes (30 meters or more) (figure 5, middle). The use of reverb on the sparse sound events allowed to recover the dense character of the waterfall, although with a noticeable difference in timbre. This is visible in figure 5, where it is clear that reverberation preserves the characteristic formants present in the parsimonious bubble generation.

6. CONCLUSION

Earth, water, air and fire: The sonic textures found in everyday soundscapes can still be largely attributed to the four classic elements. Aristotle related the elements to the two sensible axes "hot – cold" (attributes extensively used by media theorist Marshall McLuhan) and "dry – wet". It may be a coincidence, but the adjectives dry and wet have a very precise meaning in audio technology and practice. They refer to the presence or addition, in a sound signal, of reverberation.

Reverberation is multiplication of audible events and processes, thickening of textures, or texturization of acoustic elements. Such multiplication occurs in rooms at the architectural scale, but it could also occur in physical interactions between solids, in gasses, or in liquids.

In this article we argue about the importance of a good, efficient, and versatile diffusive reverberation model in the toolkit of a sound designer. We found this component to be important to convert scraping to rubbing, to generate convincing thunders and explosions, and to get showers out of single drops. These are all applications that stretch the use and interpretation of artificial reverberation beyond a purely spatial boundary, to embrace sound textures at large.

7. ACKNOWLEDGMENT

The authors are pursuing this research as part of the project SkAT-VG and acknowledge the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 618067.

8. REFERENCES

[1] James A Moorer, "About this reverberation business," *Computer music journal*, vol. 3, no. 2, pp. 13–28, june 1979.

[2] Ville Pulkki, Tapio Lokki, and Davide Rocchesso, *Spatial Effects*, pp. 139–183, John Wiley & Sons, Ltd, 2011, in Digital Audio Effects, Udo Zölzer, ed.

[3] Matti Karjalainen and Hanna Jarvelainen, "More about this reverberation science: Perceptually good late reverberation," in *Audio Engineering Society Convention 111*, Nov 2001.

[4] Jean-Marc Jot, "Efficient models for reverberation and distance rendering in computer music and virtual audio reality," in *Proc. 1997 International Computer Music Conference*, Thessaloniki, Greece, 1997, pp. 236–243.

[5] Davide Rocchesso and Julius O. Smith, "Circulant and elliptic feedback delay networks for artificial reverberation," *IEEE Transactions on Speech and Audio Processing*, vol. 5, no. 1, pp. 51–63, Jan 1997.

[6] Davide Rocchesso, "The Ball within the Box: a sound-processing metaphor," *Computer Music Journal*, vol. 19, no. 4, pp. 47–57, 1995.

[7] Enzo De Sena, Hüseyin Haciihabiboglu, Zoran Cvetkovic, and Julius O. Smith III, "Efficient synthesis of room acoustics via scattering delay networks," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 23, no. 9, pp. 1478–1492, Sept 2015.

[8] Jean-Marc Jot, "An analysis/synthesis approach to real-time artificial reverberation," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Mar 1992, vol. 2, pp. 221–224 vol.2.

[9] Vesa Norilo, "Exploring the vectored time variant comb filter," in *Proc. Int. Conf. on Digital Audio Effects*, Erlangen, Germany, 2014.

[10] Cynthia Bruyns Maxwell, "Real-time reverb simulation for arbitrary object shapes," *Proc. Int. Conf. on Digital Audio Effects, Bordeaux, France*, 2007.

[11] Jonathan S Abel and Patty Huang, "A simple, robust measure of reverberation echo density," in *Audio Engineering Society Convention 121*, 2006.

[12] Patty Huang, Jonathan S Abel, Hiroko Terasawa, and Jonathan Berger, "Reverberation echo density psychoacoustics," in *Audio Engineering Society Convention 125*, 2008.

[13] Davide Rocchesso, "Maximally diffusive yet efficient feedback delay networks for artificial reverberation," *IEEE Signal Processing Letters*, vol. 4, no. 9, pp. 252–255, Sept 1997.

[14] Stefano Delle Monache, Pietro Polotti, and Davide Rocchesso, "A toolkit for explorations in sonic interaction design," in *Proceedings of the 5th Audio Mostly Conference: A Conference on Interaction with Sound*, Piteå, Sweden, 2010, pp. 1:1–1:7.

[15] E. De Sena, N. Antonello, M. Moonen, and T. van Waterschoot, "On the modeling of rectangular geometries in room acoustic simulations," *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, vol. 23, no. 4, pp. 774–786, April 2015.

[16] Scott A van Duyne, David A Jaffe, Gregory Pat Scandalis, and Timothy S Stilson, "A lossless, click-free, pitchbendable delay line loop interpolation scheme," in *Proc. Int. Computer Music Conference*, Thessaloniki, Greece, 1997, pp. 252–255.

[17] Kees van den Doel, Paul G. Kry, and Dinesh K. Pai, "Foleyautomatic: Physically-based sound effects for interactive simulation and animation," in *Proc. Conf. on Computer Graphics and Interactive Techniques*, New York, NY, USA, 2001, SIGGRAPH, pp. 537–544, ACM.

[18] Simon Conan, Etienne Thoret, Mitsuko Aramaki, Olivier Derrien, Charles Gondre, Richard Kronland-Martinet, and Solvi Ystad, "Navigating in a space of synthesized interaction-sounds: Rubbing, scratching and rolling sounds," in *Proc. Int. Conference on Digital Audio Effects*, Maynooth, Ireland, 2013, pp. 202–209.

[19] Stefano Papetti, Federico Avanzini, and Davide Rocchesso, "Numerical methods for a nonlinear impact model: a comparative study with closed-form corrections," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2146–2158, 2011.

[20] Davide Rocchesso and William Putnam, "A Numerical Investigation of the Representation of Room Transfer Functions for Artificial Reverberation," in *Proc. XI Colloquium Mus. Inform.*, Bologna, Italy, Nov. 1995, AIMI, pp. 149–152.

[21] Oleg Mazarak, Claude Martins, and John Amanatides, "Animating exploding objects," in *Proc. Graphics Interface*, 1999, pp. 211–218.

[22] Andy Farnell, *Designing sound*, Mit Press, Cambridge, MA, 2010.

[23] R.D. Hill, "Channel heating in return-stroke lightning," *Journal of Geophysical Research*, vol. 76, no. 3, pp. 637–645, 1971.

[24] M Minnaert, "On musical air-bubbles and the sounds of running water," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 16, no. 104, pp. 235–248, 1933.

[25] Kees van den Doel, "Physically based models for liquid sounds," *ACM Trans. on Applied Perception*, vol. 2, no. 4, pp. 534–546, 2005.

GRANULAR ANALYSIS/SYNTHESIS OF PERCUSSIVE DRILLING SOUNDS

Rémi Mignot
IRCAM & CNRS, UMR 9912,
Analysis & Synthesis Team,
Paris, France
remi.mignot@ircam.fr

Ville Mäntyniemi, & Vesa Välimäki
Aalto University,
Dept. Signal Processing and Acoustics,
Espoo, Finland
v.mantyniemi@gmail.com
vesa.valimkai@aalto.fi

ABSTRACT

This paper deals with the automatic and robust analysis, and the realistic and low-cost synthesis of percussive drilling like sounds. The two contributions are: a non-supervised removal of quasi-stationary background noise based on the Non-negative Matrix Factorization, and a granular method for analysis/synthesis of this drilling sounds. These two points are appropriate to the acoustical properties of percussive drilling sounds, and can be extended to other sounds with similar characteristics. The context of this work is the training of operators of working machines using simulators. Additionally, an implementation is explained.

1. INTRODUCTION

This work is a part of a project[1] which aimed at improving the sound synthesis of working machines, in term of realism and computer resource consumption. The context is the safety at work, by the improvement of the operator training using realistic machine simulators. Among the numerous kinds of sounds, one focused our attention, the percussive drilling sounds of rig machines, and is the subject of this paper.

Two complementary algorithms are described here: a quasi-stationary noise removal using a method based on the Non-negative Matrix Factorization, and a granular analysis/synthesis method for an efficient extraction of individual clicks (shock sounds). This work is not especially dedicated to percussive drilling sounds, and it can be extended to all other sounds which have the same acoustical properties. But, because it took part in a given project, all the tests and results have been done with drilling sounds only.

The synthesis process used in this paper has similarities to other physically insprired percussive sound synthesis methods based on individual event generation [1, 2, 3, 4], but the way we create the event sounds is novel.

The drilling of the machines under interest, roughly consists in the repetitive percussions of a hammer against the rock to perforate. This hammer is at the extremity of a drill string composed by several connected rods, allowing to dig deep into the ground. The movement of this assembly of bars is fed by a hydraulic system. See Fig. 1 for an illustration of a drilling machine under interest.

Because different drilling situations may occur, different kinds of sounds can be heard. Here is a list of possible situations:

- *Normal drilling*: the typical drilling sound which should be heard when drilling.
- *Underfeed*: feeding is the method of pushing the drilling tool against the rock. In an underfeed situation, the drill is not pushed hard enough against the rock and the sound is altered compared to a normal drilling situation.
- *Overfeed*: overfeeding is caused by pushing the drill too hard against the rock, and this causes a slight variation in sound compared to normal drilling.
- *Closed Rattling*: rattling is the sound of the rods being removed from the rod string. The rods are shook heavily to open the threads holding the rods together. In this situation the threads are still closed causing a clearly distinguishable sound.
- *Open Rattling*: this situation occurs when the threads finally open is a very short but extremely loud and high frequency sound event, which is clearly different compared to rattling with the threads still closed.

Moreover, the heard sound may also depend on the length of the rod string, and the nature of the drilled rock.

Therefore, since the sounds vary with the situation, the operator may use this additional information to improve the perception of what is happening. Consequently, in the context of training and safety, the realism of the drilling sound synthesis of the rig simulator is significant. For example, the overfeeding which may cause a break of the rods, should be recognizable by the operator, nevertheless we know this situation is not easy to detect, cf. [5].

In this work, because of the numerous different possible sounds, and the numerous recorded sounds (almost one thousand), we have chosen to develop an automatic method, as robust as possible for all possible situations.

This paper is organized as follows: in sec. 2 the background noise removal is detailed. The different steps of the analysis/synthesis method are explained in sec. 3. Then, section 4 gives a brief overview of the software developed during the project. Finally, this paper is concluded in sec 5.

Figure 1: Drilling machine of Sandvik. Picture retrieved from the web site: http://construction.sandvik.com

[1]REMES project, funded by the Finnish Work Environment Fund TSR: project no. 113252.

2. BACKGROUND NOISE REMOVAL

All the available recordings of drilling sounds were corrupted by an inherent background noise coming from the diesel engine or the hydraulic system. Because of the different natures of this background noise and the drilling sound of interest, it is not possible to simultaneously analyze and synthesize them as a unique entity.

To analyze a noisy recorded sound and to resynthesize the noiseless sound, a first approach may be possible when the sound of interest can be relevantly modeled, and if the analysis is robust to noise, cf. e.g. [6]. But in our case, it seems really difficult to define a fine modeling of drilling sounds. Otherwise, an adaptive spectral subtraction may be possible to remove the background noise, cf. e.g. [7]. Nevertheless, it requires the knowledge of the noise spectrum which should be constant in time, stationary. Unfortunately, in our case it slowly changes because for instance of the moving engine speed, RPM (Revolutions Per Minutes).

Here, we propose an approach based on the Non-negative Matrix Factorization, which only assumes a quasi-stationary background noise, which may slowly change in time, and an approximatively known Signal-to-Noise Ratio (SNR).

2.1. Non-negative Matrix Factorization

For audio applications, the Non-negative Matrix Factorization (NMF) is usually used for polophonic music transcription, cf. e.g. [8, 9], denoising and source separation, cf. e.g. [10, 11]. It allows to approximate the matrix of the magnitude spectrogram, cf. e.g. [12, 8]. Using a Short-Time Fourier Transform (STFT), cf. e.g. [13, p.35], the $(M \times N)$ spectrogram matrix $V = |X|$ represents the N successive "instantaneous" magnitude spectra of the sound, which are given by its columns with M frequencies. Using this time-frequency representation, we know the variation in time of the frequency components.

In this case, the Non-negative Matrix Factorization consists in the approximation of the $(M \times N)$ matrix V, which has only non-negative elements, into the product WH as follows:

$$V \approx WH \quad \Longleftrightarrow \quad V_{m,n} \approx \sum_{k=1}^{K} W_{m,k} H_{k,n}, \tag{1}$$

where W is a $(M \times K)$ matrix and H is a $(K \times N)$ matrix. The dimension K of the factorization is chosen much smaller than M and N, i.e. $K \ll M$ and N.

Consequently, the Non-negative Matrix Factorization models the columns of V as a weighted sum of the columns of W. As a first conclusion, W is considered as the frequency dictionary giving the basis with size K on which the spectrogram V is decomposed. And since the kth row of the matrix H gives the time-varying weight of the kth word of W, column k, the matrix H is considered as the time-activation matrix.

Basically, the algorithm of this factorization consists in the minimization of a "distance" between the original spectrogram V and its approximation $\widetilde{V} = WH$. Two standard choices are the well-known Euclidean distance, and the generalized Kullbach-Leibler divergence more common in NMF, cf. e.g. [14].

The solving of this minimization problem is usually based on the iterative Newton algorithm, cf. e.g. [15, ch.4.3]. Starting from initial non-negative matrices W and H, usually randomly chosen, the matrices W and H are successively and iteratively updated. This algorithm proves to converge to the closest local minimum of the initial values. Note that we have used for this work the generalized Kullbach-Leibler divergence.

Note that additionally, the columns of W are normalized, without affecting the modeling. With $\lambda_k = \sum_m W_{m,k}^2$, the norms of the columns of W, and $\Lambda = \mathrm{diag}([\lambda_1, \lambda_2, \dots \lambda_K])$, we apply the assignments: $W \leftarrow W\Lambda$ and $H \leftarrow \Lambda^{-1}H$, at every iteration.

2.2. Quasi-stationary noise removal using NMF

As said previously, the value $H_{k,n}$ of the activation matrix gives the contribution of the kth column of W at the time index n. Then, if the kth row of H slowly varies in time, the contribution of the kth word of W also varies slowly. To remove the background noise assuming that it is quasi-stationary, the basic idea of our approach is to constrain some rows of H to vary slowly. Note that, this time smoothing is a common used constraint with NMF, cf. e.g. [10, 9, 16]. The proposed method of this paper is noticeably simpler, but proves to be satisfying in our case.

Let's define K_n the size of the noise basis, the modified NMF algorithm for the background noise removal consists in adding a "smoothing" operation of the K_n first rows of H after its updates. As a result, this new iterative algorithm will implicitly learn the noise basis, which is stored in the corresponding K_n first columns of W. At the same time, since the properties of the drilling sound are opposite, this process also learns the sound basis, which is stored in the remaining $K - K_\mathrm{n}$ other columns of W.

The smoothing operation of the first rows of H only consists in a linear filtering with a very low cutoff frequency f_c. For example, if $f_c = 2$ Hz, the obtained noise is authorized to vary only twice a second. In the same time, if $K_\mathrm{n} = 2$, in principle this method tries to extract a slowly time-varying noise, by modeling it as a moving mix of two different noises.

Nevertheless, as such, this approach does not succeed to remove the background noise, because at convergence, the gain of the corresponding rows of H are too low, and the whole spectrogram V is actually modeled by the other $K - K_\mathrm{n}$ words. To solve this problem we add a second constraint: choosing the value σ of the Signal-to-Noise Ratio (SNR), which is the ratio between the energy of the drilling sound and the energy of the background noise, we artificially modify the gains such that this ratio is checked, and without modifying the total energy.

Let's define $E\{X\} = \sum_{m,n} X_{m,n}^2$ the energy operator for all $(M \times N)$ matrices, V_n and V_s the spectrogram of the noise and the signal respectively, and the decomposition $H^T = [H_\mathrm{n}^T, H_\mathrm{s}^T]$ with $.^T$ the matrix transpose. The additional constraints are then:

$$E\{V_\mathrm{s}\} + E\{V_\mathrm{n}\} = E\{V\}, \tag{2}$$

$$E\{V_\mathrm{s}\} \;/\; E\{V_\mathrm{n}\} = \sigma. \tag{3}$$

With $\alpha_\mathrm{n} = E\{V\}/(E\{V_\mathrm{n}\}(\sigma+1))$ and $\alpha_\mathrm{s} = \alpha_\mathrm{n}\sigma E\{V_\mathrm{n}\}/E\{V_\mathrm{s}\}$, these constraints are verified by assigning after every normalization $H_\mathrm{n} \leftarrow \alpha_\mathrm{n} H_\mathrm{n}$ and $H_\mathrm{s} \leftarrow \alpha_\mathrm{s} H_\mathrm{s}$.

Because it may be difficult to know the "real" SNR of the input noisy sound, it seems to be interesting to release the SNR adjustment. For that, first the iterative algorithm is computed with Q_1 iterations where both constraints are applied, smoothing and gain adjustment, then Q_2 additional iterations are computed with only the smoothing. Consequently, during the Q_1 first iterations, the noise dictionary is learned by forcing the SNR, then the algorithm continues from this solution and converges to a close solution, with a possible different SNR.

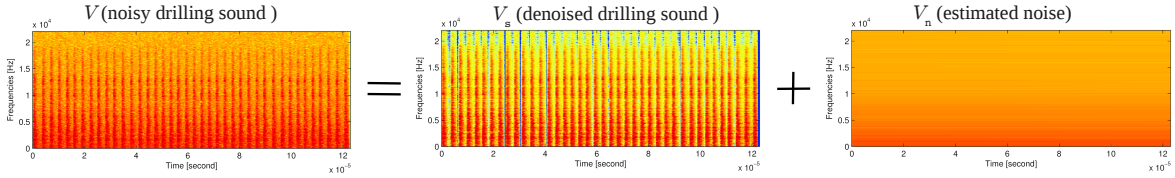


Figure 2: Illustration of the NMF based denoising. Spectrograms of the input noisy sound, denoised drilling sound, and reconstructed noise.

2.3. Noiseless drilling sound reconstruction

With the obtained approximation $\widetilde{V} = WH$, and with the original phase matrix $\Phi = \angle X$, the time signal is reconstructed with the inverse Short-Term Fourier Transform of $\widetilde{X} = \widetilde{V} \otimes \mathrm{e}^{j\Phi}$, with $\otimes$ the element-wize product. Then, the derivation of the estimated noiseless drilling sound y, follows the same principle: the inverse Short-Term Fourier Transform of $Y = V_n \otimes \mathrm{e}^{j\Phi} = (W_\mathrm{n} H_\mathrm{n}) \otimes \mathrm{e}^{j\Phi}$ is computed. This corresponds to a non-supervised source separation where the drilling sound of interest and the background noise are considered as two distinct sources.

For the project we used the following parameters: first the sampling rate F_s is forced by the available recordings, around 48 kHz. The Short-Term Fourier Transform is computed using a Hann sliding window with size 1024 samples and a fine hop size of 128 samples. The bilateral spectra with size 2048 are reduced to only consider the frequency range $[0, F_s/2]$, then $M = 1025$ bins. The NMF dimension is $K = 82$, and the noise dimension is $K_\mathrm{n} = 2$, using the cutoff frequency $f_c = 2$ Hz. During the $Q_1 = 100$ first iterations, the SNR has been constrained to 1 (0dB), and the algorithm continues with $Q_2 = 100$ other iterations. See for example the illustration of Fig. 2.

3. DRILLING SOUND ANALYSIS/SYNTHESIS

Having the denoised drilling sounds, the new challenging task is to analyze them in order to synthesize them using a realistic and very low-cost method, but also with the possibility to change the mean frequency and other parameters.

Remark that the sound is physically produced by the shock of the hammer against the rock, with possible resonances along the rod. Because of the irregular repetition of the shocks, in term of time position, amplitude, and spectrum, we choose a granular approach, which seems well suited for this case, cf. e.g. [17]. In the following, single strike sounds from a rock drill will henceforth be referred to as clicks.

To summarize the analysis process, the shocks instant and the amplitude envelope of the clicks in time are alternatively estimated twice. The first stage uses a robust but inaccurate technique, and the second one is based on a refined optimization procedure which requires a relatively good initialization. Then, some clicks are individually extracted from the input denoised sound, and the synthesis only consists in repetitively playing these extracted clicks. To get a natural synthesis, the irregularity is reproduced by randomly choosing the clicks among the stored click collection, for the spectral irregularity, and by randomly choosing the time positions and the amplitudes. These two last random processes are done according to simple rules which are learned with the input drilling sound, mean and variance of the frequency and the amplitude.

In this section, we first give some tools for the time-envelope estimation of the drilling sounds, and then all stages of the analysis are successively described.

3.1. Time-envelope estimation

Integrated energy: To detect the click positions in time and their amplitudes, we define here a smooth time function based on energy integration. With x_n the denoised drilling sound at sample n, w_n a finite weighting window with size $2N+1$, we define the smooth energy function e_n as follow:

$$e_n = \sum_{j=-N}^{N} x_{n+j}^2 w_j^2 \tag{4}$$

This energy function has the property to efficiently smooth the signal and to raise the clicks as obvious maxima. Then it will be easier to detect the click occurrences by analyzing the peaks of e_n than analyzing the peaks of x_n. Remark that as seen below, the window shape w_n is an important feature in the click detection, and it will be refined later.

Whitening: At first look, as such the energy function e_n has obvious peaks which directly correspond to the click occurrences, but has also many local maxima which do not correspond to a click.

For this reason, the original denoised drilling sound is first *whitened* by filtering it by the inverse filter obtained by a linear prediction, cf. [18]. This operation provides a deconvoluted signal y_n which has a flat spectral envelope.

Instead of computing the energy function e_n with the signal x_n, we analyze its white version y_n. This inverse filtering has the property to remove the minimal phase part, and in some cases, it concentrates the energy of a single click at its beginning. Moreover, when the high frequency components are low, it raises them, then the *contours* are enhanced for a better precision.

Here the chosen LPC order is 1024, which is quite high and provides an expensive filter, but this process is computed off-line during the analysis only.

Modeling the time-envelope of clicks: We assume here that all single clicks are the product of an unknown flat signal, possibly stochastic and different for every click, and of an envelope, identical for all clicks. This envelope is now modeled using an Attack/Decay model, separated into two parts:

- The increasing attack with length N_a and parameter α_a. Its formula $\forall n \in [0, N_a]$ is

$$a_n = \begin{cases} n/N_a, & \text{if } \alpha_a = 0, \\ \frac{1-\exp(-\alpha_a n)}{1-\exp(-\alpha_a N_a)}, & \text{otherwise.} \end{cases} \tag{5}$$

  Note that this envelope increases from 0 at $n = 0$ to 1 at $n = N_a$. An example of this behavior is shown in Fig. 3.

- The decreasing decay part has an exponential behavior with α_d the positive damping factor. Its formula $\forall n > N_a$ is

$$a_n = \exp\left(-(n - N_a)\alpha_d\right). \tag{6}$$

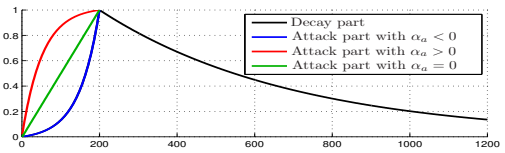


Figure 3: Illustration of an individual click envelope with different values of α_a. Here the attack time is $N_a = 200$ samples.

In the following sections, the full procedure for the click detection and the envelope estimation are described.

3.2. First estimation

First click detection: Based on the estimated energy function computed with the white signal y_n, and with a Hann weighting window with length 1023 samples, a first click detection is done by picking up the local maxima of e_n. Also, in order to know the limits of the clicks, the estimated click position are associated to the pair of local minima on the left and on the right. Then, for all detected clicks, we have: the positions P_p of the peak, P_l and P_r of the closest minimum on the left and on the right respectively.

First estimation of the envelope parameters: The previous window size is chosen to eliminate enough false alarms, but the obtained energy function is too smooth. Then, a new energy function e'_n is computed with a smaller window with size 511 samples and the envelope parameters α_a, N_a and α_d are estimated by matching the modeled envelope a_n to the square root of e'_n: $\nu(n) = \sqrt{e'_n}$, on all time ranges $[P_l, P_r]$, for all detected clicks, cf. Fig. 4.

- N_a: the attack time is computed as the median value of all $P_p - P_l$, for all detected clicks.
- α_d: the estimation of the decay factor is straightforward. Using all the values $\nu(P_p)$ and $\nu(P_r)$, for all clicks, all α_a are obtained solving

$$\begin{aligned} \nu(P_r) &= \mathrm{e}^{-\alpha_d(P_r - P_p)}\,\nu(P_p) \\ \Leftrightarrow \quad \alpha_d &= \frac{-1}{(P_r - P_p)} \log \frac{\nu(P_r)}{\nu(P_p)}, \end{aligned}$$

  and the used value is the mean of all computed values.
- α_a: the estimation of the attack factor is not easy because it is not possible to isolate it from eq. (5); we use an iterative procedure to solve the problem. With $P_m = (P_p + P_l)/2$ the middle point between P_p and P_l, the obtained attack must join the minimum point in P_l and the maximum point in P_p passing through the middle point in P_m, which means that the following equation must be solved:

$$\nu(P_m) = a(P_m - P_l)\,\nu(P_p).$$

  This solving iterative procedure is based on the simplex method, cf. [19].

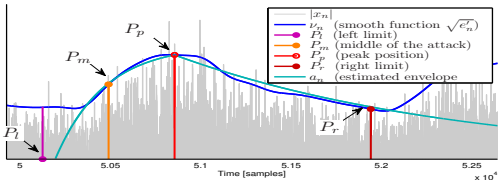


Figure 4: Illustration of the first envelope estimation. Here N_a, α_a and α_d are estimated such that the envelope of all individual clicks passes close to the 4 drawn points.

3.3. Second click detection

In the previous section, the click detection and the envelope modeling were based on the calculus of the energy functions e_n and e'_n. Unfortunately, this click detection is highly biased, because of the used window w_n which is a Hann window.

But, having a coherent estimation of the click envelope a_n we can significantly reduce this bias by replacing the Hann window by the click envelope itself. Then a new refined energy function e_n is computed using $w_n = a_n$, and again, the click positions in time are detected using the local maxima of e_n.

Indeed, if the (denoised) white signal y_n is the product of a stationary signal u_n with variance σ^2, and an envelope signal g_n which consists in a succession of click envelopes a_n, i.e. $g_n = \sum_i \beta_i a_{n-P_i}$ with P_i the time position of the ith click and β_i its amplitude, the energy function is written

$$e_n = \sum_{j=-N}^{N} y_{n+j}^2 a_n^2 = \sum_{j=-N}^{N} u_{n+j}^2 \Big(\sum_i \beta_i a_j a_{n+j-P_i}\Big)^2.$$

Since $a_n \geq 0$, the function $\left(\sum_i \beta_i a_j a_{n+j-P_i}\right)^2$ has local maxima at $n = P_i$, the true click positions, and also e_n because here u_n is assumed to be stationary.

Nevertheless, as such, this operation does not provide a good solution because of some reasons, and we need to add some limitations. First, as explained below, the first parameter estimation of the previous section is also biased, and usually the decay factor is over-estimated, then w_n is defined using the envelope a_n with a decay factor α_d half than the estimated value. Second, the size of the window $w_n = a_n$ is chosen to be half of the mean distance of the detected clicks in the previous section. This limitation reduces the influence of neighbor clicks, and improves the resolution. Third, the new energy function e_n has some "false" local minima. Then the search of the "true" local maxima is done by searching the absolute maxima of the new e_n over the time ranges: $[P_l - N_a, P_p - N_a]$, where P_l, P_r and N_a are the minima positions and the attack time estimated previously. Here the ranges are delayed by N_a, because now the maxima does not give the center of the click, but its beginning.

If no maximum is found in a selected range, then we assume there is no corresponding click, and the range is deleted. Among the obtained maxima, which are the estimated click positions P_c, beginning, we also delete some of them such that the minimal distance between two neighboring detected clicks is smaller than half of the median. The choice of the detections to delete depends on the positions themselves and also on the amplitudes e_{P_c}.

3.4. Second estimation of the envelope parameters

The estimation of section 3.2, based on a fitting of some points of the energy function e'_n, was also biased. Then, having the refined click positions P_c, with less false detections, now we can also refine the estimation of the envelope parameters N_a, α_a, and α_d.

We here propose a more accurate approach, based on an iterative algorithm for the solving of a non-linear optimization problem and with few parameters, three in our case. The chosen algorithm is based on the simplex method, cf. [19] and is implemented in Matlab as the standard function *fminseach()*.

To define the criterion to minimize, let's remark that: with a good estimation of the envelope a_n, dividing the signal y_n by the combined envelope $g_n = \sum_i \beta_i a_{n-P_i}$, with P_i the new refined position and $\beta_i = \sqrt{e(P_i)}$ its amplitude, the obtained signal z_n should have an envelope as flat as possible. Then the criterion C is based on the flatness of the energy function f_n computed with z_n. It is computed as follow:

- The envelope a_n of an individual click is computed using eqs. (5) and (6), for all $a_n \geq 0$.
- The "combined" envelope is given by: $g_n = \sum_i \beta_i a_{n-P_i}$.
- The signal z_n is computed as follows: $z_n = y_n / g_n$.
- The energy function f_n is computed with z_n and a Hann window w_n with size 1024: $f_n = \sum_{j=-N}^{N} z_{n+j}^2 w_j^2$.
- The flatness is now tested by computing the square sum of the difference of f_n and its mean. If z_n is flat, f_n is close to a constant function and the following criterion is small:

$$C = \sum_{n=0}^{N} \left(f_n - \frac{1}{N} \sum_{i=0}^{N-1} f_i \right)^2$$

With this definition, the simplex algorithm [19] provides the parameter values (N_a, α_a, and α_d) of the closest local minimal of C, from the initial values.

3.5. Last click detection

Again, with the refined envelope parameters we can obtain a refined detection of click positions, and their corresponding amplitudes as in sec. 3.3. But now, because the estimation of α_d is not biased, its value is not divided by 2.

Remark that, since the new click position P_c are refined, we could imagine to make an iterative process to refine again the envelope parameters, and so on. But in the most favorable cases, the improvements are insignificant, and in the worst cases, the process may be unstable, and may provide worse results. Then, the estimation of the click position P_c, their corresponding amplitude β_c, and the envelope parameters N_a, α_a and α_d stops here.

3.6. Click extraction

With all detected values P_c and β_c for all clicks, and the envelope parameters N_a, α_a and α_d, this section explains how a collection of some click is extracted from the denoised signal x_n. The total number of detected clicks is denoted C_d, and the number of extracted clicks is denoted C_e. Basically, $C_d \approx 70$ for 2 seconds of signals with a frequency of 35 clicks per second, and the chosen number of extracted clicks is $C_e = 10$.

The chosen extracted clicks must check the following constraints: first, to reduce the overlap of the next neighbor clicks, the distance between P_c and P_{c+1} must be as high as possible. Second, to reduce the contribution of neighbor clicks, on the left and on the right, its amplitude β_c must be higher than β_{c-1} and β_{c+1}. Then, all the C_d clicks are sorted according to these criteria, and the C_e preferred clicks are selected for extraction.

In a first step, the signal y_n is flattened, as in sec. 3.4. The flat signal z_n is obtained by the division of y_n by the new "combined" envelope

$$g'_n = \max_{i \in [1, C_d]} \Big(\beta_i a_{n-P_i} \Big).$$

Note that here we use the "max" operator instead of the sum operator. The use of the max avoids the "cumulation" of the tails of the envelopes, whereas the use of the sum favors this cumulation. In section 3.4, this effect is used to limit the value of α_d during the optimization, and now it is preferable to limit this effect.

As a result, the signal z_n is flat, which means that the effect of the click envelope is canceled, as shown in the middle sub-figure of Fig. 5. Now to extract an individual click with a damping tail which overlaps with the following clicks, we just have to multiply the flat signal z_n by the envelope a_n of an individual click, cf. the bottom sub-figure of Fig. 5.

In this way, neither the tail of the previous click is totally removed, and nor the signal of the following one, in a strict sense. But the contribution of neighbor clicks is efficiently removed for the following reasons: first, thanks to the simultaneous masking, the extracted click of interest efficiently masks the tail of the previous one because its envelope is smaller. Second, the amplitude of the following one is efficiently reduced and thanks to the temporal masking, its contribution is not audible in most of the cases.

Finally, since the clicks are associated with the white signal y_n, then we cancel this whitening by filtering the extracted click samples by the LPC filter $H(z) = 1/A(z)$, cf. sec. 3.1. Moreover, since the extracted clicks have different amplitudes, they are normalized to a unique norm.

3.7. Statistical parameters

The real-time granular synthesis only consists in successively playing the C_e extracted clicks with a random selection at each time and with a quasi-constant frequency. However, to provide a realistic synthesis, it is needed to add a fluctuation in frequency and in amplitude, as it is the case in real drilling sounds.

First the mean period T_0, in samples, of the click occurrences is computed as the mean of the distances of detected clicks:

$$T_0 = \frac{1}{C_d - 1} \sum_{i=1}^{C_d - 1} P_{i+1} - P_i,$$

and the mean frequency is then $F_0 = F_s / T_0$. Also, to take account of the variance of the click position, the fluctuation in term of frequencies, the standard deviation is computed as follows:

$$\sigma_T = \Big(\frac{1}{C_d - 1} \sum_{i=1}^{C_d - 1} \big(P_{i+1} - P_i - T_0 \big)^2 \Big)^{\frac{1}{2}}$$

Second, in the same way the mean amplitude B and its standard deviation are computed:

$$B = \frac{1}{C_d - 1} \sum_{i=1}^{C_d - 1} \beta_i$$

$$\sigma_B = \Big(\frac{1}{C_d - 1} \sum_{i=1}^{C_d - 1} \big(\beta_i - B \big)^2 \Big)^{\frac{1}{2}}$$

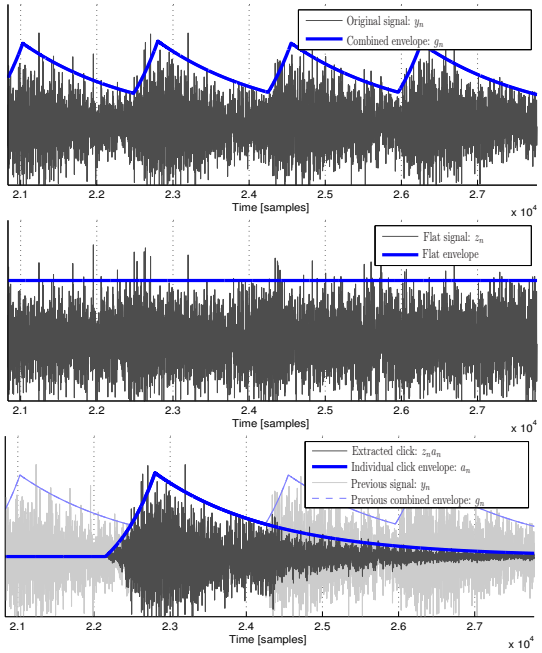


Figure 5: Illustration of the click extraction, based on the flattening of the termporal envelope of a drilling sound recording.

3.8. Real-time synthesis

Then, when a new individual click has just started to be played back, in real-time, the synthesizer chooses the instant of the next played click with a delay time randomly chosen according to a distribution with mean $T_0 = 1/F_0$ and variance σ_T^2. Also its amplitude is randomly chosen according to a distribution with mean B and variance σ_B^2. This procedure provides some fluctuations in the amplitudes and the positions which makes the synthesis more realistic. Moreover, the new played click is randomly chosen among the collection of the C_e extracted clicks. Figure 6 summarizes the global sound analysis/synthesis process.

In this work, the background noise of the engine is not resynthesized from the NMF analysis. The reasons are that its resynthesis based on the NMF does not yield a satisfying sound quality and that another good technique for background noise synthesis was already available in the project.

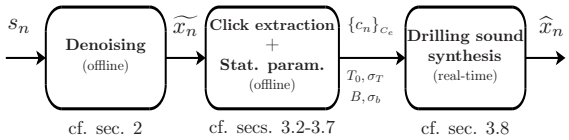


Figure 6: Summary of the global sound synthesis process: first the drilling sound of interest x_n is extracted from the noisy recorded sound s_n, cf. sec. 2, then the analysis extracts C_e clicks noted here c_n, cf. 3.2-3.6, and the statistical parameters (T_0, σ_T, B and σ_B) are estimated, cf. sec. 3.7. Finally the sound $\widehat{x_n}$ is synthesized, cf. 3.8.

3.9. Informal listening test

No formal and rigorous listening test has been done to evaluate the denoising and the synthesis. Nevertheless, the simple comparison of a former drilling sound synthesis, made by Creanex, and this new synthesis reveals an outstanding improvement. Moreover, an informal listening test has been made with four experts who have an excellent experience in drilling machines. As a result, they judged the denoised sounds as sufficiently correct, and they found the drilling synthesis realistic and accurate. Consequently, it proves the ability of this method to accurately synthesize drilling sounds. More details of this test are given in [20].

4. SOFTWARE IMPLEMENTATION

This section summarizes the software developed during the REMES project. The main goal is to provide some tools and examples for: the manual annotation of drilling sounds, the automatic analysis, the test of results, and the efficient real-time synthesis.

These applications have been used with the drilling sounds provided by Sandvik, but they have been designed to work with other sounds having similar signal properties. For more details about this software package, we refer the reader to [20].

4.1. Annotation

Because most of the recordings contain some different successive drilling situations (normal drilling, overfeed, rattling, ...), it is necessary to annotate the limits of each part for each file. The automatic annotation is difficult, and it was not the subject of this work. Moreover, a bad annotation may provide a worse analysis. Then a simple Graphical User Interface has been developed in Matlab to annotate easily and quickly all the sounds, cf. Fig. 7.

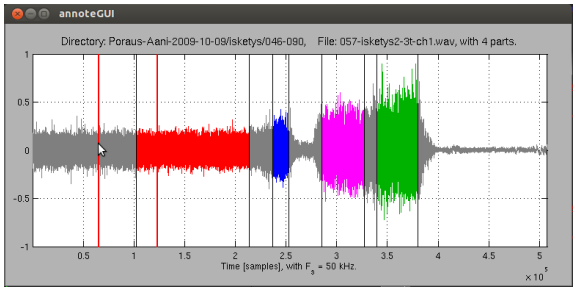


Figure 7: Screenshot of the annotator software. In this example, 4 parts are annotated with different colors.

4.2. Analyzer

The analyzer is a piece of software which automatically computes the analysis of all annotated parts, without manual intervention.

At the moment of the end of the project, 1296 parts are annotated, and the complete analysis lasts approximately 7 hours, using a 4 core CPU at 3.20 GHz. All steps of sec. 3 are computed successively for all annotated parts. Note that to make this process faster, the length of the parts are limited by 2 seconds. Finally, all analyzed parameters, such as the extracted clicks, the envelope

parameters N_a, α_a and α_d, and the statistical parameters, F_0, σ_T, B and σ_B, are stored into some data files.

4.3. Test and parameter refinement

To check and eventually modify the analyzed parameters, another GUI has been developed using Matlab, cf. Fig. 8.

This software loads the analyzed drilling data, computed by the analyzer, and proposes to compare the original noisy sound, the denoised drilling sound and the synthesis. Also, some sliders and editor controls allow to modify the synthesis parameters, and eventually to save the results in the data file.

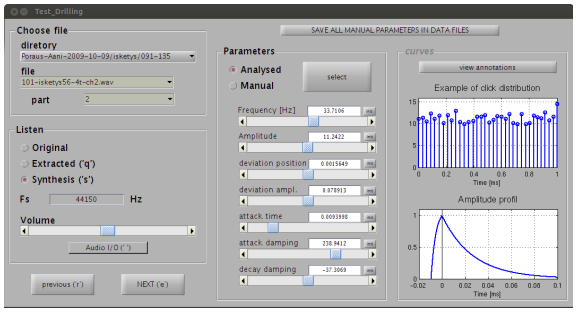


Figure 8: Screenshot of the Graphical User Interface for testing and for the parameter modification.

4.4. Real-time synthesis library

For the efficient real-time synthesis, a C++ library and a demonstration application have been developed. The main benefit of this drilling sound synthesis tool, compared to a simple playback of the denoised sound, is the ability to modify in real-time some parameters and to reproduce a continuous modification of the sound.

The available parameters are: the mean frequency, the mean amplitude, the deviation in time, the deviation in amplitude, a modified attack factor and a modified decay factor. Fig. 9 shows a screenshot of this demonstration application, with six sliders for the parameter control.

5. CONCLUSION

These paper presents two complementary methods useful for the purpose of the mentioned project: first, a non-supervised removal of quasi-stationary noise has been proposed, which is based on the Non-negative Matrix Factorization. This algorithm is relevant because of the opposite natures of the background noise and the drilling sound of interest. Second, a granular analysis/synthesis approach has been presented. It roughly consists in: the estimation of the click positions in time and of the time envelope modeling of the single clicks; the extraction of some individual clicks; and finally a successive play of the stored clicks, with some random rules learned on the original sound. Not only the synthesis method used a small amount of memory, less than 500 kBytes per sound, but also the CPU consumption is negligible on current personal computers.

Note that, at the end of this work, 1296 annotated drilling sounds have been analyzed for tests. These recorded sounds have

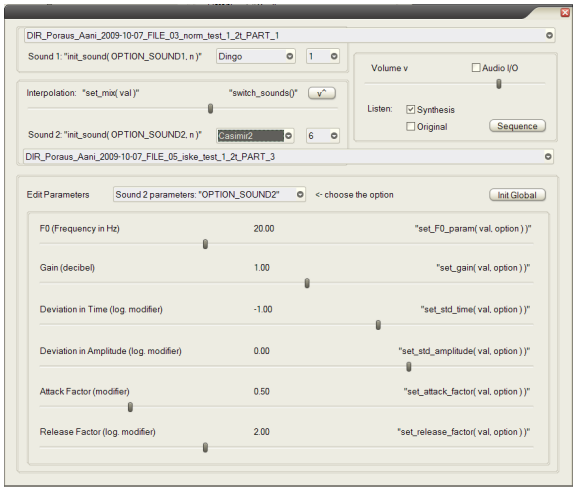


Figure 9: Screenshot of the Graphical User Interface. Example of implementation of the C++ real-time library.

been produced by rig machines of Sandvik during previous projects, cf. e.g. [5]. The developed methods and software package can be used for synthesizing realistic drilling sounds in a working machine simulator. As demonstration, the companion webpage [21] proposes the listening of some excerpts of sounds: original, denoised and synthesized.

6. ACKNOWLEDGMENTS

This work was funded by the Finnish Work Environment Fund TSR (project no. 113252, REMES – Realistic Machine and Environmental Sounds for a Training Simulator to Improve Safety at Work).

7. REFERENCES

[1] P.R. Cook, "Modeling Bill's gait: Analysis and parametric synthesis of walking sounds," in *Proc. Audio Eng. Soc. 22nd Conf. Virtual, Synthetic, and Entertainment Audio*, 2002, pp. 73–78.

[2] L. Peltola, C. Erkut, P.R. Cook, and V. Välimäki, "Synthesis of hand clapping sounds," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 3, pp. 1021–1029, 2007.

[3] R. Nordahl, L. Turchet, and S. Serafin, "Sound synthesis and evaluation of interactive footsteps and environmental sounds rendering for virtual reality applications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 9, pp. 1234–1244, 2011.

[4] S. Oksanen, J. Parker, and V. Välimäki, "Physically informed synthesis of jackhammer tool impact sounds," in *Proc. Int. Conf. Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, Sept. 2013, pp. 168–171.

[5] S. Oksanen, T. Pirinen, and V. Välimäki, "Subjective assessment of percussive drilling sounds," in *Proc. Internoise 2009*, Ottawa, Canada, Aug. 2009, 2009.

[6] C. Févotte, B. Torrésani, L. Daudet, and S.J. Godsill, "Sparse linear regression with structured priors and application to denoising of musical audio," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 174–285, 2008.

[7] S.F. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-27, no. 2, pp. 113–120, Apr. 1979.

[8] P. Smaragdis and J.C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, USA, Oct. 2003.

[9] N. Bertin, R. Badeau, and E. Vincent, "Enforcing harmonicity and smoothness in bayesian non-negative matrix factorization applied to polyphonic music transcription," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 638–549, 2010.

[10] T. Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Trans, Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1066–1074, Mar. 2007.

[11] A. Ozerov and C. Févotte, "Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 550–563, 2010.

[12] D.D. Lee and H.S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[13] U. Zölzer (Ed.), *DAFX - Digital Audio Effects*, John Wiley & Sons, Ltd., 2002.

[14] D.D. Lee and H.S. Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Process. Systems*, pp. 556–562, 2000, MIT Press.

[15] E. Walter and L. Pronzato, *Identification of Parametric Models*, Communications and Control Engineering, 1997, 413 pages.

[16] N. Mohammadiha, P. Smaragdis, and A. Leijon, "Supervised and unsupervised speech enhancement using nonnegative matrix factorization," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2140–2151, 2013.

[17] B. Truax, "Real-time granular synthesis with a digital signal processor," *Computer Music Journal*, vol. 12, no. 2, pp. 14–26, 1988.

[18] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, 1975.

[19] J.C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder-Mead simplex method in low dimensions," *SIAM Journal of Optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[20] V. Mäntyniemi, R. Mignot, and V. Välimäki, "REMES Final Report - The Finnish Work Environment Fund TSR Project no. 113252," Series: Aalto university publication series science + technology, 16/2014, Aallto University, 2014.

[21] R. Mignot, "Granular analysis/synthesis of precussive drilling sounds: Companion web page," `http://research.spa.aalto.fi/publications/papers/dafx15-drilling/`.

[22] M. Heiniö, "Rock excavation handbook," Sandvik Tamrock Corp, 1999.

[23] V. Mäntyniemi, "Sound synthesis in working machine simulators," M.S. thesis, Aalto University, 2014.

FEATURE DESIGN FOR THE CLASSIFICATION OF AUDIO EFFECT UNITS BY INPUT / OUTPUT MEASUREMENTS

Felix Eichas, Marco Fink, Udo Zölzer

Department of Signal Processing and Communications,
Helmut-Schmidt-Universität
Hamburg, Germany
felix.eichas@hsu-hh.de

ABSTRACT

Virtual analog modeling is an important field of digital audio signal processing. It allows to recreate the tonal characteristics of real-world sound sources or to impress the specific sound of a certain analog device upon a digital signal on a software basis. Automatic virtual analog modeling using black-box system identification based on input/output (I/O) measurements is an emerging approach, which can be greatly enhanced by specific pre-processing methods suggesting the best-fitting model to be optimized in the actual identification process. In this work, several features based on specific test signals are presented allowing to categorize instrument effect units into classes of effects, like distortion, compression, modulations and similar categories. The categorization of analog effect units is especially challenging due to the wide variety of these effects. For each device, I/O measurements are performed and a set of features is calculated to allow the classification. The features are computed for several effect units to evaluate their applicability using a basic classifier based on pattern matching.

1. INTRODUCTION

Audio effect units are used by musicians or sound engineers to transform the signal of an (electrical) instrument in order to modify it in a certain way. Many of these effect units already have been emulated and digital models have been derived to apply these effects while, for example, mixing a recording [1]. Many musicians value *vintage* music equipment because of its specific tonal characteristic and want to recreate this characteristic without spending a large amount of money on rare vintage equipment. With the aid of system identification, the unique properties of an effects unit can be captured and emulated as a *VST plugin* or with a DSP-based effects unit in a live-setup.

A lot of effect units have been emulated in very different ways. Modeling via circuit analysis, as done by [2–5], is realized by transforming the schematic of the device into a digital model. This procedure is very precise and can describe effect units accurately. But it also has several drawbacks. The computational load is very high, because one or more nonlinear equations have to be solved iteratively for every nonlinear element in the circuit and every sample of the input data. In addition, the characteristic of every nonlinear circuit element has to be known or assumed to solve the nonlinear equations.

An alternate way of emulating analog audio units is system identification with input/output (I/O) measurements. By sending specifically designed input signals through the device under test (DUT) and measuring the output, the influence of the system on the input signal can be determined and recreated with a digital model to achieve the same characteristics [6].

Black-box identification is still an important topic and there are countless contributions to this domain from neural networks to wavelet-transform based models for identification. Sjöberg et al. published a summary of different nonlinear modeling techniques [7]. A method, that has been successfully used to model nonlinear audio systems, was introduced by Novak et al. [8]. They used a block-oriented Hammerstein model for the identification of a distortion guitar effect pedal.

Feature extraction from audio data is an important topic for music information retrieval (MIR) [9]. Nevertheless, typical MIR features like Zero Crossings or Spectral Flux can not be utilized to classify the subtle tonal characteristic of specific audio effect devices.

The majority of commercial digital audio effects, emulating a specific device, are parametric digital models which are usually tweaked by a professional sound engineer to approach the sound of the analog unit. This identification procedure can be automated using black-box modeling. The proposed feature set can facilitate the decision which digital model is best-suited to reproduce the characteristics of the analog effects unit under test. For this purpose specifically designed input signals are generated, replayed through the device, and, by recording the output, the influence of the system on these signals is measured and different characteristics are extracted.

In future work, the extracted features can be used to classify the DUT and choose an appropriate model from a model set. The classification can be done in several ways, e.g. with a neural network or a weighted decision tree. Once a model is chosen, an identification algorithm can be used to fit the model's characteristics to resemble the DUT.

This paper describes the signal model for several typical effects in section 2, the input signals designed for the feature computation are shown in section 3, and the computation of the features is explained in section 4. In section 5 the measurement-setup is shown and the features are evaluated by measuring some typical effect pedals and comparing the resemblance of the pedal characteristic and the computed feature. The usability of the features is demonstrated using a simple classifier in section 6. Section 7 concludes this paper.

2. SIGNAL MODEL

This section describes the influence of typical effects, as categorized in Fig. 1, on an input signal $x(t)$. This analysis was done to

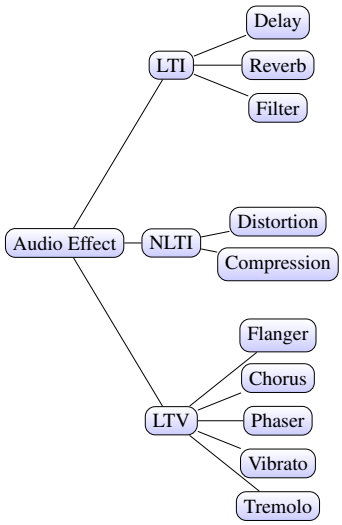


Figure 1: *Categorization of typical digital audio effects into linear time-invariant (LTI), non-linear time-invariant (NLTI), and linear time-variant (LTV).*

design proper input signals and select adequate features, calculated from the recorded (output) signals.

2.1. Linear Time-Invariant Effects

2.1.1. Filter

In the field of digital audio effects, a filter is a linear time-invariant (LTI) system, which is able to amplify or attenuate certain frequency regions of the input signal $x(t)$. The impulse response $h(t)$ defines the characteristic of the filter and is convolved with the input signal to produce the filtered output,

$$y(t) = x(t) * h(t). \tag{1}$$

2.1.2. Reverberation

Although, the mathematical representation of the reverberation effect

$$y(t) = x(t) * h_{\text{rev}}(t), \tag{2}$$

namely the convolution of the input signal with an impulse response h_{rev}, is very similar to the filter effect, those effects are easy to distinguish from each other. The impulse response of filters and reverberations differ significantly in length and noisiness and hence can be differentiated with measurements.

2.1.3. Delay

Repetitions of an input signal can be achieved using delay pedals. The output signal

$$y(t) = x(t) + g \cdot y(t - t_d) \tag{3}$$

can be modeled with a direct path and a delay line in the feedback path. The delay time t_d defines the temporal distance between two repetitions while the amplitude of the repetitions is controlled with the feedback gain g. It should be noted that this is a simple model which does not take into account that many delay effects employ some modulation or filtering in the feedback path to create a certain kind of tonal characteristic.

2.2. Non-Linear Time-Invariant Effects

2.2.1. Compression

Compression is a non-linear effect reducing the dynamic range of the input signal. Therefore, the input signal $x(t)$ is fed to time-variant variable-gain amplification stage, weighting $x(t)$ with a gain factor $g(t)$ to produce the output

$$y(t) = g(t) \cdot x(t). \tag{4}$$

The variable-gain amplifier can be modeled as an envelope signal $x_+(t)$ smoothed with a signal-dependent lowpass filter $\text{LP}_{\text{AT/RT}}$ like

$$g(t) = \text{LP}_{\text{AT/RT}}\{x_+(t)\}, \tag{5}$$

where $\text{LP}_{\text{AT/RT}}$ defines a lowpass filter for the attack (AT) and release (RT) case. Typical choices for the envelope signal $x_+(t)$ are the peak signal $|x(t)|$ or the *root-mean-square* signal $x_{\text{RMS}}(t)$, depending on the type of compressor.

2.2.2. Distortion

Distortion effects modify the input signal $x(t)$ with a nonlinear function $f(x)$ mapping the level of the input signal $x(t)$ to the level of the output signal $y(t)$, as shown in

$$y(t) = f(x(t)). \tag{6}$$

The shape of the nonlinear function defines the tonal quality of the effect. Musicians tend to sub-categorize distortion effects in overdrive, distortion, and fuzz in ascending order of nonlinearity. For an accurate signal model, representing an analog distortion effect unit, there should be additional input and an output filters. Here they are omitted for the sake of readability.

2.3. Linear Time-Variant Effects

Linear time-variant effects, often called *modulation* effects, modulate the input signal in terms of volume, frequency or phase using a low frequency oscillator (LFO) controlling the rate of the modulation.

2.3.1. Tremolo

The tremolo effect was introduced in the 1950s by companies like Fender or Vox. Initially it was an electronic circuit integrated in the guitar amplifier which modulated the volume of the output signal periodically. The modulating signal $x_{\text{LFO}}(t)$ is multiplied with the input signal $x(t)$

$$y(t) = x_{\text{LFO}}(t) \cdot x(t). \tag{7}$$

$x_{\text{LFO}}(t)$ is a periodic signal having, for example, sinusoidal, triangular, sawtooth or square-wave characteristic.

2.3.2. Vibrato

Vibrato is, like the tremolo, one of the earliest common guitar effects. The effect originated from the so called *whammy bar*, a mechanical lever typically located at the bridge of the guitar, which modulated the tension of the strings and therefore the frequency of the guitar tone. Another popular effect unit was called the *Leslie speaker*, where a rotating speaker created frequency modulation due to the Doppler effect. This behavior is recreated in effect units by producing an output signal

$$y(t) = x(t - x_{\text{LFO}}(t)), \tag{8}$$

where $y(t)$ is a delayed version of the input signal $x(t)$. The periodic variation of the time delay by the LFO signal $\text{x}_{\text{LFO}}(t)$ produces a recurrent pitch variation in the output signal.

2.3.3. Phaser

A phaser produces an output signal which is a mix of the unprocessed, *dry* input signal $d \cdot x(t)$ and an allpass-filtered, *wet* version of the same signal, $w \cdot \text{AP}^M\{(x(t), x_{\text{LFO}}(t))\}$, where d and w are gain factors weighting the unprocessed or processed parts of the input signal. The allpass filter cascade introduces a frequency dependent phase shift in respect to the dry signal. When both signals are added together to yield the output signal

$$y(t) = d \cdot x(t) + w \cdot \text{AP}^M\{(x(t), x_{\text{LFO}}(t))\}. \tag{9}$$

Phase cancellation and elevation for certain frequencies occur. This results in notches in the spectrum of the output signal. These notches are moving along the frequency axis over time, because $x_{\text{LFO}}(t)$ modulates the center frequency of the allpass filters periodically. The amount of notches is controlled by the order M of the allpass filter cascade. For first order filters there are $\frac{M}{2}$ notches because a first-order allpass filter introduces a maximum phase shift of π [10].

2.3.4. Chorus

The chorus effect originated from the idea of emulating several musicians playing in unison (the same melody). Because they are not playing in perfect sync, there are small deviations in loudness and timing. To achieve this effect one or more copies of the input signal are delayed by 10 ms to 25 ms with small and random variations in the delay times and added to the original signal. It is a mix between the original (*dry*) signal $x(t)$ and a vibrato effect,

$$y(t) = x(t) + w \cdot x(t - x_{\text{LFO}}(t)). \tag{10}$$

The delayed signal is realized with a variable delay line, controlled by the LFO-signal $x_{\text{LFO}}(t)$ [11, 12].

2.3.5. Flanger

The flanger produces a similar sound to that of a chorus effect but differs in some aspects. The main difference is the delay time of the modulation path. It is between 0 ms and 15 ms. Secondly, for most flangers, the modulation is done with a feedback path while the chorus has a feed-forward architecture [11]. The output signal

$$y(t) = x(t) + w \cdot x_{\text{SP}}(t), \tag{11}$$

consists of a direct path $x(t)$ and a weighted second path $w \cdot x_{\text{SP}}(t)$ as shown in Fig. 2. The second path is the feedback path with

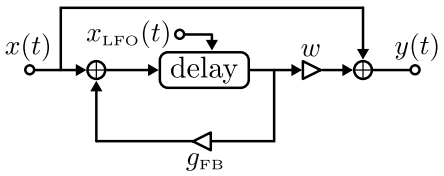


Figure 2: *Block diagram of a flanger.*

feedback gain g_{FB},

$$x_{\text{SP}}(t) = x(t - x_{\text{LFO}}(t)) + g_{\text{FB}} \cdot x_{\text{SP}}(t - x_{\text{LFO}}(t)). \tag{12}$$

The oscillator signal $x_{\text{LFO}}(t)$ modulates the delay of the second path with a sinusoidal signal [12].

3. INPUT SIGNALS

In this section the input signals which are used for the feature computation, are introduced. Because the input signals were digitally synthesized, the authors decided to change the notation from continuous time signals $x(t)$ to discrete time signals $x(n)$ where $x(t)$ is sampled with sampling rate f_s.

3.1. Sinusoidal Signals

3.1.1. Sine Wave with Fixed Amplitude

The first basic input signal is a sinusoidal wave with frequency f_0, duration of d seconds, and fixed amplitude A,

$$x_{\text{sin fix}}(n) = A \cdot \sin\left(2\pi \frac{f_0}{f_s} n\right), \tag{13}$$

with $n = [0, \ldots, d \cdot f_s]$ and sampling rate f_s.

3.1.2. Amplitude-Modulated Sine Wave

The sine signal can be modified to feature varying envelopes

$$x_{\text{sin var}}(n) = a(n) \cdot \sin\left(2\pi \frac{f_0}{f_s} n\right), \tag{14}$$

where $a(n)$ is a time-variant amplitude. In this work, the amplitude switched abruptly between a_{high} and a_{low} with a low frequency, e.g. $f_{\text{mod}} = 2\,\text{Hz}$.

3.2. Impulse

To identify a system's linear behaviour for a whole frequency range, a broadband stimulus has to be fed to the system. The discrete unit impulse

$$x_{\text{pulse}}(n) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{else} \end{cases}, \tag{15}$$

achieves the excitation of a system for the whole frequency range uniformly and is used to measure the impulse response of the DUT.

3.3. Sweep

Another way of exciting a system within a certain frequency range is to feed an exponentially swept sine wave, as described in [13]. The sweep

$$x_{\text{sweep}}(n) = \sin\left(\omega_1 \cdot \frac{N}{\ln(\frac{\omega_2}{\omega_1})} \cdot \left[\mathrm{e}^{\frac{n}{N}\cdot\ln(\frac{\omega_2}{\omega_1})} - 1\right]\right), \quad (16)$$

has a duration of N samples and covers frequencies from $\omega_1 = \frac{2\pi f_{\text{start}}}{f_s}$ to $\omega_2 = \frac{2\pi f_{\text{stop}}}{f_s}$.

4. FEATURES

As previously mentioned, this study is groundwork for a classification procedure that shall present the features on which a following classification can be based. In this section, the features are derived and adapted to create a comparable quantitative representation.

4.1. Distortion Feature

A typical description of distortion for audio devices is the so-called total harmonic distortion (THD)

$$\text{THD} = 20 \cdot \log_{10}\left(\frac{\sqrt{V_1^2 + V_3^2 + \ldots + V_6^2}}{V_0}\right), \quad (17)$$

where V_i is the RMS amplitude of the i_{th} harmonic. To calculate the THD, signal $x_{\text{sin fix}}$ (see Eq. 13) is sent through the DUT and recorded. Afterwards the recorded signal is segmented to contain k times the period length $k \cdot \frac{f_s}{f_0}$ samples of the sine wave. This has the advantage that there is no leakage effect for the frequency f_0 and all of its harmonics, i.e. all the energy of f_0 falls into one Discrete Fourier Transform (DFT) bin.

The THD is calculated by extracting the RMS amplitudes of the fundamental frequency and the RMS amplitude of the harmonics. The highest harmonic considered is V_6. The related metric s_{THD} is defined as

$$s_{\text{THD}} = \frac{\text{THD} + 70}{80} \quad (18)$$

and maps the values from the initial range of $[-70, \ldots, 10]$ to $[0, \ldots, 1]$. Outlying values are clipped to the maximum and minimum value, respectively.

4.2. Compression Feature

To detect a compressor, the signal levels of the varying amplitude sine wave $x_{\text{sin var}}$ from (Eq. 14) are analyzed. If the signal-level-difference of the output would be smaller than the signal-level-difference of the input, dynamic range compression is occurring. But reduction of the signal-level-difference is also occurring in a highly nonlinear distortion effect. For this reason the compression feature is based on the attack and release regions of a compressor and not on the signal-level-difference. Hence, it is assumed that attack and release times differ significantly.

The computation of the compression feature is depicted in Fig. 3 and described in the following. After recording the replayed test signal $x_{\text{sin var}}$ (see Eq. 14), with two different amplitude levels (see Fig. 4 (a)), the compression feature is calculated. At first, the envelope of the signal is computed by linearly interpolating between all its local maxima. The result of this processing step is shown in Fig. 4 (b).

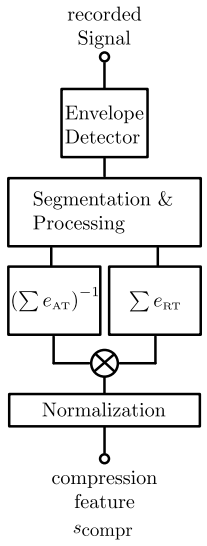


Figure 3: *Block-diagram of the compression feature calculation.*

The resulting envelope is segmented into attack and release part by cutting each part according to the time-domain positions of the input signal. The beginning of the attack-part corresponds to the region of the input signal $x_{\text{sin var}}$ where it changes from high signal level a_{low} to a_{high}. And the start point of the release-part is the region where the input signal changes from low signal level a_{low} to a_{high}. The segmented envelope parts of attack and release regions are depicted in Fig. 4 (c) and (d).

After attack and release part have been segmented, the release envelope is mirrored horizontally. Then, the minimum of both curves is subtracted from the signal. The result of this processing step yields only the overshoot of attack and release filter and is shown in Fig. 4 (e) and (f). The colored areas in said figure represent the integral over the curves.

The ratio of the integrals is computed and normalized to form the compression feature

$$s_{\text{compr}} = |F_C - 1|, \quad (19)$$

where $F_C = \frac{\sum e_{RT}(n)}{\sum e_{AT}(n)}$ is the ratio of the release and attack integrals. The initial assumption that a compressor has different settings for attack and release time does not always have to be true, but most commercial compressor units tend to have a much longer release time than attack time, whereas release and attack time of a distortion effect unit should be roughly the same. s_{compr} becomes zero if attack and release time are the same and approaches a higher value if attack and release time differ. s_{compr} is clipped to one if it should exceed this value.

4.3. Time Variance Feature

To detect the time-variance of a system, the exponential sine sweep, input signal x_{sweep}, is sent three times through the DUT and recorded. To avoid sending the signal with the same periodicity as a potential LFO of the time-variant effect could feature, the pauses between each sweep are non-uniform. The recorded signal is segmented into the three recorded sweeps $y_1(n)$, $y_2(n)$, and

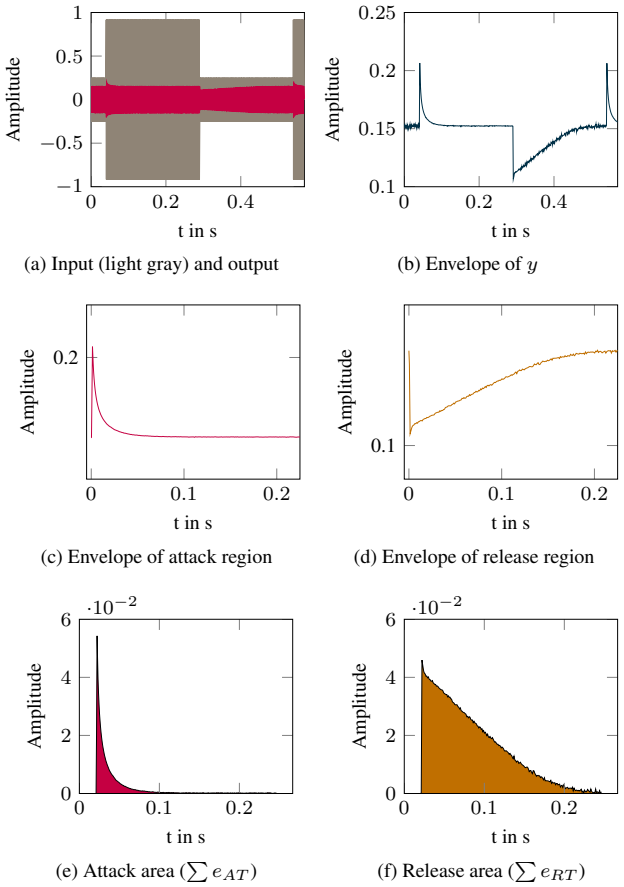


Figure 4: *Compression Feature - Measured waveforms of an MXR DynaComp.*

$y_3(n)$ which are then subtracted from each other to yield three error signals $e_1(n) = y_1(n) - y_2(n)$, $e_2(n) = y_1(n) - y_3(n)$, and $e_3 = y_2(n) - y_3(n)$. The *root mean square* (RMS) value of $e_1(n)$, $e_2(n)$, and $e_3(n)$ is calculated and the maximum value e_{rms} is selected to form the time-variance metric

$$s_{\text{tvar}} = 1 - e^{-10 \cdot e_{rms}}. \tag{20}$$

Since a system is either time-invariant or time-variant the time-variance metric is distorted with a nonlinear function (see Eq. 20), to emphasize values of s_{tvar} which are non negligible.

Please note that the time-variance metric is sensitive to the level of the error signals $e_1(n)$, $e_2(n)$ and $e_3(n)$. Therefore the recorded signal $y_{\text{sweep}}(n)$ is normalized before segmentation to have a maximum amplitude of one before calculating the time-domain error.

4.4. Impulse Response Length Feature

As previously mentioned, the class of linear time-invariant effects is quite versatile. To distinguish simple filters from effects which massively influence the temporal structure of a source signal, like

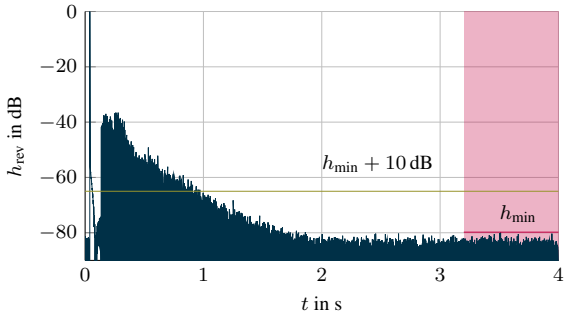


Figure 5: *Exemplary impulse response of reverb effect.*

reverb and delay effects, the length of the impulse response is a valuable information. The impulse response length feature is determined by the duration of the impulse response to decrease to a certain value. For this purpose, the DUT is excited with x_{pulse} to record the impulse response h_{rev}, which is normalized to an amplitude of 1 in the following. The maximum value h_{min} of the last section of the normalized impulse response, shown as red area in Fig. 5, is then weighted to define the threshold $h_{\text{thr}} = h_{\text{min}} + 10\,\text{dB}$. Afterwards the impulse response is scanned in time-inverse direction to find the point of time s_{len} where the first sample exceeds h_{thr}. In Fig. 5, this corresponds to $s_{\text{len}} = 0.94$ s. The feature yields the time it takes the impulse response to decrease below the threshold in seconds. So the metric only takes values greater than zero, $s_{\text{len}} \geq 0$, and is clipped to one if it exceeds one second.

5. MEASUREMENTS

5.1. Setup

The measurement setup is depicted in Fig. 6. A computer is connected to a high quality external *USB audio interface*. The output of the audio interface is connected to the input of the DUT and the output of the DUT is connected to the input of the audio interface. True amplitude scaling is guaranteed by sending a sine

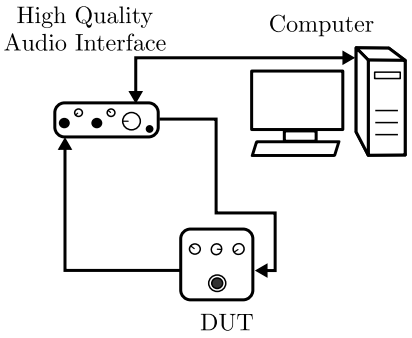


Figure 6: *Measurement Setup.*

with 0 dBFS (full scale), measuring the maximum amplitude of the recorded signal and computing the gain factor. Every recorded signal is weighted with this gain factor.

Time synchronization is achieved by computing the cross correlation $r_{xy}(m)$ between recorded signal $y(n)$ and input signal to the DUT $x(n)$. The shift of the main peak of $r_{xy}(m)$ from the center position indicates the amount of samples the recorded signal is delayed in respect to the input signal.

The different input signals, described in section 3, are merged into one long signal before they are sent through the DUT. After the signal has been recorded, it is cut into its original parts and each part is given to the feature extraction.

For extreme settings of the effect units, e.g. very long decay times for reverb or delay, problems arose while recording the output of the DUT. To assure that the effects do not alter the signal beyond recognition, all potentiometers of the effect units were set to a neutral (12 o'clock) position during the measurements. This also assured that the effect would not be too light to alter the signal enough.

5.2. Results

To get an impression of the applicability of the proposed features, they are computed for several real effect pedals. All analyzed pedals and the resulting features are listed in Tab. 1. First, the Harley Benton 7-band EQ was utilized. The distortion feature of $s_{\text{THD}} = 0.15$ and the time variance feature $s_{\text{tvar}} = 0.001$ clearly show the linear, time-invariant behavior of this system. The TubeScreamer, ZenDrive, and the FuzzFace DIY clones from Musikding also show negligible time variance values. In contrast, the distortion feature values show a rising trend from 0.54 to 0.78. The strong non-linearity of those pedals is reflected in those values.

The MXR Dyna Comp compressor also shows a high distortion value of $s_{\text{THD}} = 0.64$. Nevertheless, it can be clearly distinguished from the distortion pedals due to the s_{compr} value of 0.87, which doesn't exceed 0.06 for the distortion pedals. The s_{tvar} measurements for the already mentioned pedals are as expected consistently very low. When the measurements are repeated with modulation pedals (MXR Flanger, MXR Phase90, and the BOSS CEB-3 Bass Chorus) a significant difference is detected. The smallest value is $s_{\text{tvar}} = 0.44$ for the Flanger. Unexpectedly, the modulation pedals also show much distortion. The verification of this finding by spectral analysis and actual listening affirmed the nonlinear characteristics of these devices. Nevertheless, the categorization into the group of time-variant effects is still possible with proposed time variance feature.

The s_{compr} value yields no noteworthy results for all time-variant effect units. Due to time variant phase-shifts and signal superpositions, the envelope of the recorded signal behaves in an, in this context, unpredictable way. This leads to s_{compr} values varying between 0.07 to 1, for repeated measurements. Even though the delay is technically not a time-variant effect, the results for the compression metric do not yield an accurate or even reproducible result. The computation of the compression metric is based on the envelope of the recorded signal, which is corrupted by the copies of the input signal which are added to the output signal. The remaining effect pedals of the analysis are the TC Electronics *Hall of Fame* reverb and the *Flashback X4* delay. Apparently, these devices tend to perform quite linear ($s_{\text{THD}} = 0.14, 0.06$) but differ strongly from the EQ pedal in terms of their impulse response length s_{len} which are 0.94 and over 1 seconds respectively. Considering the insignificant s_{len} values for other pedals the authors assume that the identification of reverb and delay pedals using the proposed feature is doubtlessly realizable.

Table 2: *Template vectors for different effect categories.*

Filter	$\hat{v}_F = [0,\ 0,\ 0,\ 0]$
Distortion	$\hat{v}_D = [1,\ 0,\ 0,\ 0]$
Compression	$\hat{v}_C = [0,\ 0,\ 1,\ 0]$
Linear Time-variant	$\hat{v}_{TV} = [0,\ 1,\ 0,\ 0]$
Reverb o. Delay	$\hat{v}_{RV} = [0,\ 0,\ 0,\ 1]$

6. EXEMPLARY CLASSIFICATION

To demonstrate the usability of the proposed features a very simple classifier was implemented. It is based on pattern matching as suggested by [14] for the purpose of chord identification based on chroma vectors. Therefore, the features from section 4 for a specific device are combined in a feature vector

$$v = [s_{\text{THD}},\ s_{\text{tvar}},\ s_{\text{compr}},\ s_{\text{len}}]. \tag{21}$$

In the next step, the squared euclidean distance Δv_i

$$\Delta v_i = \sum (v - \hat{v}_i)^2 \tag{22}$$

of the current feature vector v to all template vectors $\hat{v}_i$, listed in Tab. 2 is computed. The output of the classifier is the effect corresponding to the template vector resulting in the smallest Δv_i. The results of these calculations can be seen in Fig. 7. The euclidean distance from the current feature vector to each Δv_i (Tab. 2) is shown for each effect unit under test. The lowest distance yields the result of the classification. Already this very simple approach delivers a flawless classification for the values of Tab. 1.

7. CONCLUSION

This goal of this study was to derive features that allow the classification of audio effect units via I/O measurements. Based on simple signal models of typical effects and specifically designed input sequences, the features are derived. The features describe the characteristic of a DUT in terms of distortion, compression, time variance, and impulse response length.

The experimental application of the features on 10 different guitar effect units (*stompboxes*) proved the usability of the features to differentiate between effect classes. A very simple classifier based on pattern matching was already capable of flawlessly distinguishing between the tested devices. Advanced classifiers based on neural networks or support vector machines are expected to perform even better and be able to handle more effect categories using potentially more features. This process flow allows the autonomous modeling of various audio processing equipment.

Further improvements could be implemented to differentiate even more effects. Currently it is for example not possible to distinguish a reverb effect unit from a delay effect unit or a chorus from a flanger. Further features can be designed for enhanced classification offering detailed information about the device under test.

8. REFERENCES

[1] U. Zölzer, *DAFX: Digital Audio Effects*, vol. 2, Wiley Online Library, 2011.

[2] M. Holters and U. Zölzer, "Physical modelling of a wah-wah effect pedal as a case study for application of the nodal

Table 1: *Overview of results for various effect pedals.*

Effect Category	**Effect Unit**	s_{THD}	s_{tvar}	s_{compr}	s_{len}
Filter	Harley Benton EQ	0.15 (-57.72 dB)	0.005	0.04	0.11
Distortion	Musikding TubeScreamer	0.54 (-27.2 dB)	0.005	0.03	0.002
Distortion	Musikding Zendrive	0.64 (-18.89 dB)	0	0.06	0.06
Distortion	Musikding FuzzFace	0.78 (-7.57 dB)	0.0198	0.03	0.002
Compression	MXR Dyna Comp	0.64 (-18.9 dB)	0.1813	0.87	0
Flanger	MXR Flanger	0.45 (-33.94 dB)	0.8892	(0.1 - 0.8) time variant	0.03
Phaser	MXR Phase90	0.93 (4.61 dB)	0.970	(0.07 - 0.8) time variant	0.004
Chorus	BOSS CEB-3 Bass Chorus	0.4 (-37.65 dB)	0.993	(0.3 - 1) time variant	0.006
Reverb	TC Hall of Fame	0.14 (-58.6 dB)	0.30	0.138	0.94
Delay	TC Flashback X4 (2290)	0.06 (-65.07 dB)	0.020	(0 - 1) superposition	1

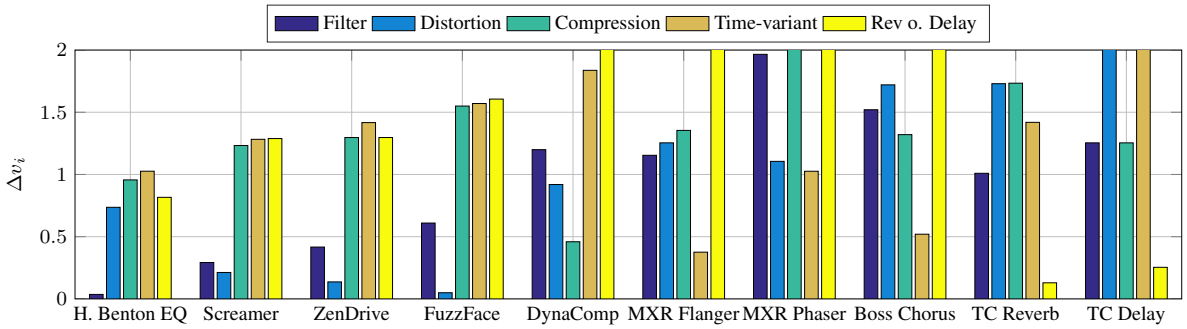


Figure 7: *Squared euclidean distance of feature vector and template vector for all measured devices.*

dk method to circuits with variable parts," in *Proceedings of Digital Audio Effects (DAFx-11)*, Paris, France, Sept. 19-23, 2011, pp. 31–35.

[3] D. Yeh and J.O. Smith, "Simulating guitar distortion circuits using wave digital and nonlinear state-space formulations," in *Proceedings of Digital Audio Effects (DAFx-08)*, Espoo, Finland, Sept. 1-4, 2008, pp. 19–26.

[4] J. Macak, *Real-time Digital Simulation of Guitar Amplifiers as Audio Effects*, Ph.D. thesis, Brno University of Technology, 2011.

[5] M. Karjalainen and J. Pakarinen, "Wave digital simulation of a vacuum-tube amplifier.," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006)*, Toulouse, France, May 2006, pp. 153–156.

[6] L. Ljung, *System identification*, Springer, 1998.

[7] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.Y. Glorennec, H. Hjalmarsson, and A. Juditsky, "Nonlinear black-box modeling in system identification: a unified overview," *Automatica*, vol. 31, no. 12, pp. 1691–1724, 1995.

[8] A. Novak, L. Simon, P. Lotton, and J. Gilbert, "Chebyshev model and synchronized swept sine method in nonlinear audio effect modeling," in *Proc. Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 6-10, 2010.

[9] C. McKay, I. Fujinaga, and P. Depalle, "jaudio: A feature extraction library," in *Proceedings of the International Conference on Music Information Retrieval*, London, UK, September 2005, pp. 600–3.

[10] F. Eichas, M. Holters, M. Fink, and U. Zölzer, "Physical modeling of the mxr phase 90 guitar effect pedal," in *Proceedings of Digital Audio Effects (DAFx-14)*, Erlangen, Germany, Sept. 1-5, 2014, pp. 153–158.

[11] P. Dutilleux, "Filters, delays, modulations and demodulations: A tutorial," in *Proceedings of DFX98, Digital Audio Effects Workshop*, Barcelona, Spain, 1998, pp. 4–11.

[12] S. Orfanidis, *Introduction to Signal Procesing*, vol. 1, Prentice-Hall, Inc., 1996.

[13] A. Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique," in *Audio Engineering Society Convention 108*, Paris, France, February 2000.

[14] T. Fujishima, "Realtime Chord Recognition of Musical Sound: a System Using Common Lisp Music," *Proc. of International Computer Music Conference (ICMC), 1999*, pp. 464–467, 1999.

REAL-TIME 3D AMBISONICS USING FAUST, PROCESSING, PURE DATA, AND OSC

Pierre Lecomte,

Lab. de Mécanique des Structures et Systèmes Couplés (LMSSC), Conservatoire National des Arts et Métiers Paris, France
Groupe d'Acoustique de l'Université de Sherbrooke (GAUS), Université de Sherbrooke, Québec, Canada
pierre.lecomte@gadz.org

Philippe-Aubert Gauthier,

GAUS
Université de Sherbrooke, Québec, Canada

ABSTRACT

This paper presents several digital signal processing (DSP) tools for the real-time synthesis of a 3D sound pressure field using Ambisonics technologies. The spatialization of monophonic signal or the reconstruction of natural 3D recorded sound pressure fields is considered. The DSP required to generate the loudspeaker signals is implemented using the FAUST programming language. FAUST enables and simplifies the compilation of the developed tools on several architectures and on different DSP tool format. In this paper, a focus is made on the near-field filters implementation which allows for the encoding of spherical waves with distance information. The gain variation with distance is also taken into account. The control of the synthesis can be made by software controllers or hardware controllers, such as joystick, by the mean of PURE DATA and OPEN SOUND CONTROL (OSC) messages. A visual feedback tool using the PROCESSING programming language is also presented in the most recent implementation. The aim of this research derives from a larger research project on physically-accurate sound field reproduction for simulators in engineering and industrial applications.

1. INTRODUCTION

Ambisonics technologies allow describing 3D sound pressure fields using a projection on a truncated spherical harmonics basis [1]. The resulting 3D encoded sound pressure field can later be manipulated, decoded and reconstructed over several loudspeaker-layouts or even headphones [2]. Ambisonics has two main objectives: the spatialization of virtual sound sources or the reconstruction of recorded sound pressure fields [3]. Several software solutions exist to create, transmit, manipulate, and render sound pressure fields using Ambisonics technologies. See references [4, 5, 6, 7, 8] as examples. Albeit being popular for practical applications in music, sound design and audio context, classical and common Ambisonics implementations suffer from few drawbacks that limit their use for physically-accurate sound field reproduction with applications to environment simulators (vehicles, working environments, etc.) in industrial or engineering context. Indeed, the near-field encoding [2] is rarely provided and the encoding/decoding in 3D context is limited to the first orders, hence limiting the spatial resolution and area size of physically-accurate reproduction. Indeed, if the sound field is controlled up to an order M, the reconstruction area size is frequency-dependent and given by $r = Mc/2\pi$ [9] (where r is the are size radius, c the speed of sound in air, and f the frequency). The near-field support is also detrimental for physically-accurate sound field reproduction as it takes into account the loudspeaker distance from the origin in order to compensate for the loudspeakers spherical waves. In this trend, this work is motivated by the need to develop a practical implementation of Ambisonics for industrial applications such as laboratory reproduction of industrial sound environments, vehicles cabin noise, virtual vibroacoustics models, architectural spaces, etc. In these scenarios, the reproduced sound field must be as close as possible than the target sound field. Some recent examples of such applications potentials are found in Refs. [10, 11, 12, 13]. Typical outcomes are related to listening tests, sound quality testing, perceptual studies and other. On this matter, as mentioned by Vorländer [12] with respect to Ambisonics implementations that often include modifications inspired by auditory perception to increase the listener experience with respect to some expectations, a "generally applicable reproduction system [for sound field simulation] must not introduce any artificial auditory cue which is not part of the simulated sound." [12].

In this context, this paper presents an implementation of Ambisonics technologies for real-time synthesis of 3D sound field up to 5^{th} order. The signal processing is implemented in FAUST[1] (Functional AUdio STream) programming language. This language proposes a functional approach to implement the signal processing and it compiles in efficient C++ code [14]. From this code, DSP tools are provided in several formats such as: VST, LV2, Pure Data, Max/MSP, JACK QT, and others. Thus, from the same FAUST code, one can generate tools working on various operating systems and configurations.

The focus of this paper is on physical encoding and decoding of 3D sound pressure fields with a special attention dedicated to the near field filters development, definition and implementation. After defining the notations in use in Sec. 2, the main equations of Ambisonics are recalled in Sec. 3. In Sec. 4 the implementation in FAUST programming language is addressed with a special attention on near-field filters. Section 5 presents a visual feedback tool using PROCESSING language. This tool helps visualizing in 3D the virtual sources and the loudspeaker levels. Finally, in Sec. 6, the user control interface is addressed, presenting the possibility of interfacing all elements with OSC protocol.

[1] http://faust.grame.fr/

2. COORDINATE SYSTEM AND NOTATIONS

In this section, the different notations used throughout the paper are presented and illustrated.

Spherical coordinate system

The following spherical coordinate system is used throughout this paper and shown in Fig. 1:

$$u_1 = r\cos(\theta)\cos(\delta),\ u_2 = r\sin(\theta)\cos(\delta),\ u_3 = r\sin(\delta) \tag{1}$$

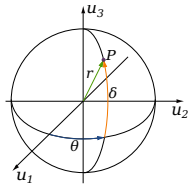


Figure 1: Spherical coordinate system. A point $P(u_1, u_2, u_3)$ is described by its radius r, azimuth θ and elevation δ.

A virtual source position is denoted with its spherical coordinates r_1, θ_1, δ_1. The rendering loudspeakers are arranged on a sphere of radius r_0, as shown in Fig. 2.

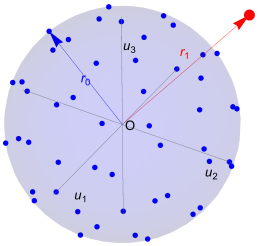


Figure 2: Ambisonics playback layout. Blue: the rendering loudspeakers disposed on a spherical layout of radius r_0. Red: the virtual source at radius position r_1.

Spherical harmonics

The spherical harmonics used in this paper are defined as follow in [1]:

$$Y^{\sigma}_{mn}(\theta,\delta) = \sqrt{(2m+1)\epsilon_n \frac{(m-n)!}{(m+n)!}} P_{mn}(\sin(\delta)) \times \begin{cases} \cos(n\theta) & \text{if} \quad \sigma = 1 \\ \sin(n\theta) & \text{if} \quad \sigma = -1 \end{cases} \tag{2}$$

where P_{mn} are the associated Legendre functions of order m and degree n, m and $n \in \mathbb{N}$ with $n \leq m$, $\sigma = \pm 1$ and $\epsilon_n = 1$ if $n = 0$, $\epsilon_n = 2$ if $n > 0$. The spherical harmonics order are denoted by m and its degree, by n. For each order m there are $(2m+1)$ spherical harmonics. Thus a basis truncated at order M contains $L = (M+1)^2$ functions.

Notations

The Laplace variable is denoted s and the discrete time variable z (discrete domain). A vector is denoted by lowercase bold font $\mathbf{v}$ and a matrix by uppercase bold font $\mathbf{M}$. Superscript ^{T} designates the transposition operation. j is imaginary unit with $j^2 = -1$.

3. AMBISONICS

In this section, the principal Ambisonics equations are recalled. They will later be used for the real-time DSP implementation.

3.1. Encoding

In Ambisonics, the encoding step consists in deriving *B-Format* signals [2] from either monophonic signal (with a spatialization context) or microphone array signals (natural sound field encoding).

3.1.1. Monophonic signal

From a monophonic signal, the encoding can be done for a plane wave with amplitude $a(z)$, azimuth θ_1 and elevation δ_1 direction or a spherical wave, adding a distance information r_1

$$\begin{aligned} B^{\sigma}_{mn}(z) &= a(z) Y^{\sigma}_{mn}(\theta_1, \delta_1) && \text{Plane wave} \\ B^{\sigma}_{mn}(z) &= a(z) F_{m,r_1}(z) Y^{\sigma}_{mn}(\theta_1, \delta_1) && \text{Spherical wave} \end{aligned} \tag{3}$$

In Eq. 3, filters $F_{m,r_1}(z)$ are the forward near-field filters which take into account the finite distance of the virtual source r_1 [2].

3.1.2. Rigid spherical microphone array encoding

For a natural 3D sound pressure field recording made with a rigid spherical microphone array of radius r_a, the B^{σ}_{mn} are given by:

$$B^{\sigma}_{mn}(z) = E_{m,r_a}(z) \sum_{i=1}^{N} Y^{\sigma}_{mn}(\theta_i, \delta_i) w_i p_i(z) \tag{4}$$

$E_{m,r_a}(z)$ are the equalization filters which take into account the diffraction of the rigid sphere [3]. The sound pressure signal at the i^{th} capsule position $(r_a, \theta_i, \delta_i)$ is denoted $p_i(z)$ on the array of N microphones. The B^{σ}_{mn} components are guaranteed to be exact up to order M if the spatial sampling of the spherical microphone array respects the orthonormality criterion of spherical harmonics up to M [15, 16]. Thus, there is possibly a weighting factor w_i for each capsule in Eq. (4) to ensure this condition. The working bandwidth of the array without aliasing is given by: $f \leq Mc/(r_a 2\pi)$ [17], where f is the frequency, c the sound speed, and M the maximum working order for the array.

Equation (4) is for a triplet of indices (m, n, σ). Thus, up to order M, the *B-Format* signals vector is obtained with matrix notation:

$$\mathbf{b}(z) = \mathbf{E}(z) \cdot \mathbf{Y}^T_{\text{mic}} \cdot \mathbf{W}_{\text{mic}} \cdot \mathbf{p}(z) \tag{5}$$

$\mathbf{b}(z)^{(L\times 1)} = [B^1_{00}(z) \cdots B^{\sigma}_{mn}(z) \cdots B^1_{M0}(z)]$. $\mathbf{E}(z)^{(L\times L)}$ is the diagonal matrix of equalization filters with diagonal terms $[E_{0,r_a}(z) \cdots E_{m,r_a}(z) \cdots E_{M,r_a}(z)]$, each $E_{m,r_a}(z)$ term being replicated $(2m+1)$ times on the main diagonal. $\mathbf{Y}^{(N\times L)}_{\text{mic}}$ is the matrix of spherical harmonics up to order M evaluated at each direction (θ_i, δ_i). $\mathbf{W}^{(N\times N)}_{\text{mic}}$ is the diagonal matrix of weightings.

[2] We call here the *B-Format* the signals vector $\mathbf{b}(z)$ $[B^1_{00}(z) \cdots B^{\sigma}_{mn}(z) \cdots B^1_{M0}(z)]$

$\mathbf{p}^{N\times 1} = [p_1(z)\cdots p_i(z)\cdots p_N(z)]$ is the vector of capsule signals.

3.2. Decoding

In this paper, only the basic decoder (mode-matching [18]) is recalled for a full-sphere configuration of radius r_0 respecting the spherical harmonics orthonormality via weighting coefficients [16]. For decoders adapted to irregular loudspeaker layout or other decoding concerns, see for example [6].

If one considers a set of N_2 loudspeakers, the input signal of the l^{th} loudspeaker at position $(r_0, \theta_l, \delta_l)$ is obtained from the B^{σ}_{mn} signals by:

$$s_l(z) = w_l \sum_{m=0}^{M} (F_{m,r_0})^{-1}(z) \sum_{n=0}^{m} \sum_{\sigma=\pm 1} Y^{\sigma}_{mn}(\theta_l, \delta_l) B^{\sigma}_{mn}(z) \tag{6}$$

where $(F_{m,r_0})^{-1}(z)$ are the inverse near field filters or near field compensation filters, which take into account the finite distance r_0 of the rendering loudspeakers [2]. Since the reconstructed sound pressure field is the summation of each sound field generated by each loudspeakers, the vector of input signals $\mathbf{s}$ is given by:

$$\mathbf{s} = \mathbf{W}_{\text{spk}} \cdot \mathbf{Y}_{\text{spk}} \cdot \mathbf{F}^{-1}_{r_0}(z) \cdot \mathbf{b}(z) \tag{7}$$

where $\mathbf{s}(z)^{(N_2\times 1)} = [s_1(z)\cdots s_l(z)\cdots s_{N_2}(z)]$. $\mathbf{F}^{-1}_{m,r_0}{}^{(L\times L)}(z)$ is the diagonal matrix of near field compensation filters with diagonal terms $[1/F_{0,r_0}(z)\cdots 1/F_{m,r_0}(z)\cdots 1/F_{M,r_0}(z)]$, each $1/F_{m,r_0}(z)$ term being replicated $(2m+1)$ times on the main diagonal. $\mathbf{Y}^{(N_2\times L)}_{\text{spk}}$ is the matrix of L spherical harmonics up to order M evaluated at each direction (θ_l, δ_l). $\mathbf{W}^{(N_2\times N_2)}_{\text{spk}}$ is the diagonal matrix of weightings w_l. Note that the resulting matrix $\mathbf{M}_{\text{spk}} = \mathbf{W}_{\text{spk}} \cdot \mathbf{Y}_{\text{spk}}$ is the Ambisonics decoding matrix as defined in Ref.[6].

3.3. Equivalent panning-law

In a spatialization context, recalling Eq. (3) and Eq. (6) along with the additivity theorem of spherical harmonics [19], one can directly compute the loudspeaker input signals:

$$\begin{aligned} s_l(z) &= a(z) w_l \sum_{m=0}^{M} \frac{(2m+1)}{F_{m,r_0}(z)} P_m(\gamma_l) && \text{Plane wave} \\ s_l(z) &= a(z) w_l \sum_{m=0}^{M} (2m+1) \frac{F_{m,r_1}}{F_{m,r_0}}(z) P_m(\gamma_l) && \text{Spherical wave} \end{aligned} \tag{8}$$

where $\gamma_l = \cos(\delta_1)\cos(\delta_l)\cos(\theta_1 - \theta_l) + \sin(\delta_1)\sin(\delta_l)$ is relative to the angle between the virtual source in θ_1, δ_1 and the l^{th} loudspeaker.

4. FAUST IMPLEMENTATION

The implementation of Ambisonics tools such as an encoder, basic decoder, near-field filters, and spatialization tools using equivalent panning-law is made using the FAUST language. An overview of the current developped tools is shown in Fig.3. From left to right, the vertical branches correspond to: 1) microphone signal processing, 2) panning of a monophonic signal as virtual point source with encoding and decoding and 3) panning of a monophonic signal as virtual point source with equivalent panning law. The rightmost branch is the control via OSC and visual feedback. Each box corresponds to a module described in the next sections. The common functions of each modules are implemented in library files, which enable the quick design of new modules by re-using existing FAUST code. In the following section, the near-field filters implementation is detailed.

4.1. Near-field filters

4.1.1. Forward filters

The forward filters are given in the Laplace domain [2]:

$$F_{m,r_1}(s) = \sum_{i=0}^{m} \frac{(m+i)!}{(m-i)!i!2^i} \left(\frac{c}{sr_1}\right)^i \tag{9}$$

with $s = j2\pi f$. To convert this analog filter in the digital z domain, the use of a bilinear transform requires precision arithmetic to be efficient, as pointed out by Adriaensen in [20]. In this latter reference, he proposes another s-to-z mapping scheme to obtain a digital realization of the filter which is robust with single precision floating point format:

$$\begin{aligned} \zeta^{-1} = \frac{z^{-1}}{1-z^{-1}} = \sum_{k=1}^{\infty} (z^{-1})^k \quad \text{for} |z| > 1 \\ s = \frac{2F_s}{1+2\zeta^{-1}} \end{aligned} \tag{10}$$

where F_s is the sampling frequency. The implementation of an m^{th} order forward filter is made by product of sections of the form:

$$\begin{aligned} H_{1,r_1}(z) &= g_1(r_1)(1 + d_{1,1}(r_1)\zeta^{-1}) \\ H_{2,r_1}(z) &= g_2(r_1)(1 + d_{2,1}(r_1)\zeta^{-1} + d_{2,2}(r)\zeta^{-2}) \end{aligned} \tag{11}$$

For example a 3rd order filter F_{3,r_1} is realized with the product of a 1st order section and a 2nd order section: $F_{3,r_1}(z) = H_{1,r_1}(z) \cdot H_{2,r_1}(z)$. The computation of each coefficient g_1, g_2, $d_{1,1}$, $d_{2,1}$, $d_{2,2}$ in Eq. (11) is detailed in [20].

4.1.2. Stabilization with inverse filters

The forward filters $F_{m,r_1}(z)$ present an infinite gain at 0 Hz. Thus, they must be stabilized by multiplying with an inverse filter (or near field compensation filters) $1/F_{m,r_0}(z)$. This can be done at the encoding stage as suggested by Daniel [2]:

$$B^{\sigma}_{mn}(z) = a(z) \frac{F_{m,r_1}}{F_{m,r_0}}(z) Y^{\sigma}_{mn}(\theta_1, \delta_1) \quad \text{Spherical wave} \tag{12}$$

It means that to encode a spherical wave, according to Eq. (12), one should know the rendering array radius r_0 *a priori*. However this is not a major concern. Indeed, if the encoded spherical wave is reconstituted on another layout of radius r_2, one can correct by multiplying the $B^{\sigma}_{mn}(z)$ components by $F_{m,r_0}(z)/F_{m,r_2}(z)$.

4.1.3. Gain correction for spherical source

In [2], the near field filters are defined with the assumption that the amplitude $a(z)$ of the spherical wave in Eq. (12) is taken at the origin O in Fig. 2. Thus, the propagation term $e^{sr_1/c}/(4\pi r_1)$

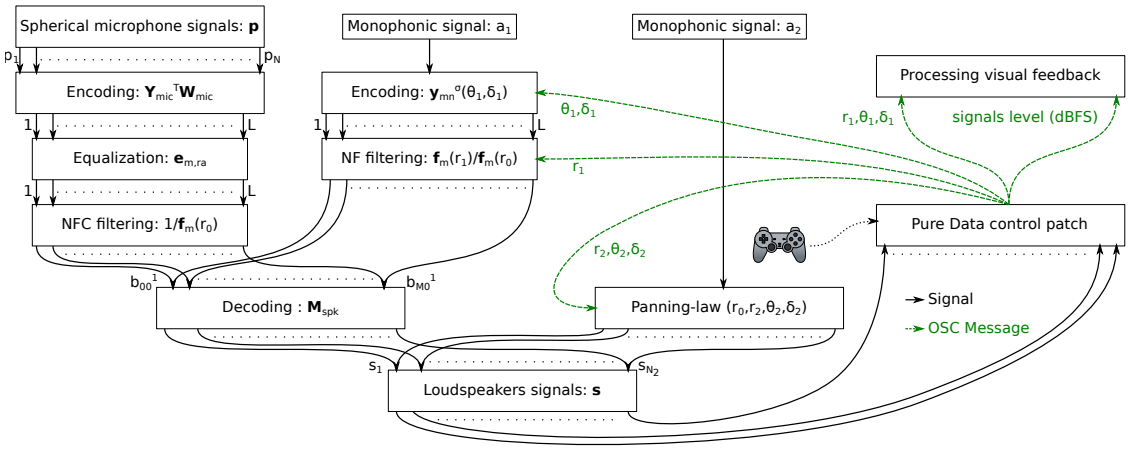


Figure 3: Main features of the current implementation: real-time synthesis of 3D recordings (leftmost vertical branch), spatialization (two centered vertical branches), and control via OSC with visual feedback (rightmost vertical branch). The digital temporal signals are represented by black arrows and the OSC instructions message in green dashed arrows.

(Laplace's domain) is not taken into account. This leads to the definition of Eq. (9) for the forward filters. In the same way, the near-field compensation filters do not take into account the propagation from rendering source to origin O: $e^{sr_0/c}/(4\pi r_0)$. The main consequence is that there is no gain variation with the distance of the virtual source. To correct this, one multiplies the amplitude $a(z)$ for spherical wave in Eq. (12) by: $\frac{r_0}{r_1} z^{-(r_1-r_0)/c}$. The corresponding delay can be fractional depending on r_1 and r_0. In the case of a focused source (i.e. inside the rendering loudspeakers enclosure), the delay is negative and an implementation could require time-reversal method, as in Wave Field Synthesis [21]. However, since this delay is the same for all rendering loudspeakers, according to Eqs. (6) and (8), it is omitted for simplicity. As a result, the pressure sound field of a virtual point source will be reproduced physically in the reproduction zone with correct gain and with a phase shift of $z^{-(r_1-r_0)/c}$ when this latter is omitted. Finally, the encoding and the panning-law of a spherical wave becomes:

$$\begin{aligned} B^{\sigma}_{mn}(z) &= a(z)\frac{r_0}{r_1}\frac{F_{m,r_1}}{F_{m,r_0}}(z)Y^{\sigma}_{mn}(\theta_1,\delta_1) \\ s_l &= a(z)\frac{r_0}{r_1}w_l\sum_{m=0}^{M}(2m+1)\frac{F_{m,r_1}}{F_{m,r_0}}(z)P_m(\gamma_l) \end{aligned} \tag{13}$$

4.1.4. FAUST *implementation*

The near field filters $F_{m,r_1}(z)/F_{m,r_0}(z)$ (as well as the near field compensation filters $1/F_{m,r_0}(z)$) are implemented in a FAUST library `nfc.lib` following Eq. (11) and up to 5^{th} order. The implementation is based on a Direct-Form II [22]. As an example, the FAUST code for the second order section is given as:

```
TFNF2(b0,b1,b2,a1,a2) =
sub~sum1(a1,a2): sum2(b0,b1,b2)
        with {
        sum1(k1,k2,x)=
x:(+~_<:((_':+~_),*(k1))):*(k2),_:+);
        sum2(k0,k1,k2,x)=
x<:*(k0),+~_,_:_,(-<:*(k1),(_':+~_))
*(k2):+):+;
        sub(x,y)= y-x;
};
```

The block diagram for this code is shown in Fig. 4. The coefficients $g_2(r_1), d_{2,1}(r_1), d_{2,2}(r_1)$, $g_2(r_0), d_{2,1}(r_0), d_{2,2}(r_0)$ were precomputed in this figure for simplicity to obtain the corresponding $b_0(r_1), b_1(r_1), b_2(r_1), a_1(r_0), a_2(r_0)$ coefficients, poles an zeroes of the filter. However, `nfc.lib` provides the real-time computation of these coefficients knowing r_1 and r_0, according to [20]. In the current implentation, the near-field filters are provided as independent modules (see Fig. 3), or included in the equivalent panning-law module (see Sec. 4.4)

As an example, the gain frequency response of the filters $r_0/r_1 \times F_{m,r_1}/F_{m,r_0}$ up to order five are shown in Fig. 5 for $r_1 = 1$ m, $r_0 = 3$ m (solid lines) and $r_1 = 3$ m, $r_0 = 1$ m (dashed lines). $c = 340$ m/s and $F_s = 48000$ Hz. For focused sources (i.e $r_1 \leq r_0$, solid lines in Fig. 5), as r_1 decreases, or as r_0 increases and as m increases, the gain of the filters increases at low frequencies. Thus, when encoding a focused spherical source (i.e. $r_1 \leq r_0$), one should be aware of these extreme gains. These "bass-boost" effects create strong artifacts outside the control zone and can easily damage the rendering loudspeakers. A solution could be to impose a minimum r_1 regarding to maximum a $r_0/r_1 \times F_{m,r_1}/F_{m,r_0}$ gain. This maximum gain is then related to the maximum linear excursion of the loudspeakers. Note that another approach for focused sources in Ambisonics can as well be a solution [23].

4.2. Encoding of captured sound field and decoding

This case corresponds to the leftmost vertical branch of Fig. 3. The encoding of a 3D sound field captured by a spherical microphone array as described in Eq. (5) requires a matrix operation as the decoding operation of Eq. (7). This matrix operation is done in FAUST as suggested by Heller [6]:

```
// bus with gains
gain(c) = R(c) with {
  R((c,cl)) = R(c),R(cl);
  R(1)      = _;
```

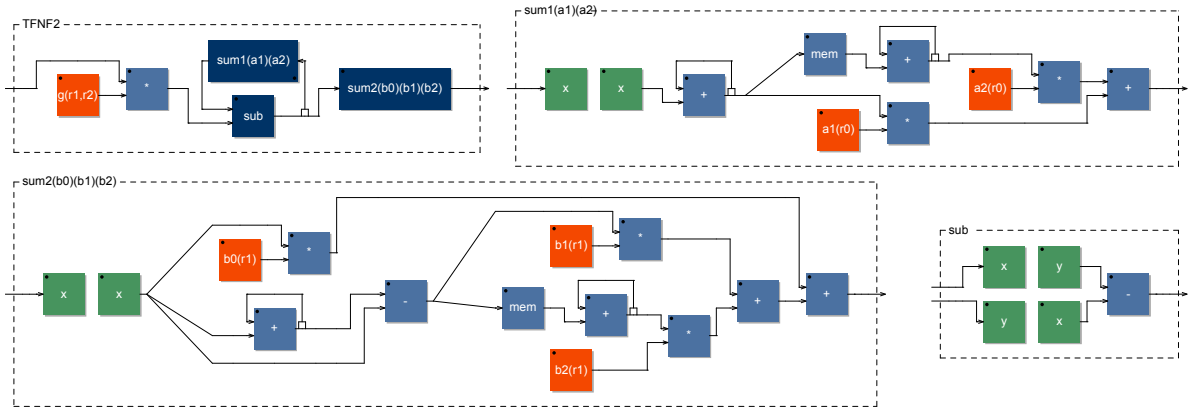


Figure 4: Block diagram representation of the `TFNF2` function. The input signal to be filtered is on the main top left diagram. The others diagrams detail each block in this main diagram. `x` and `y` are input signals in a block. These diagrams where generated using `faust2svg` tool.

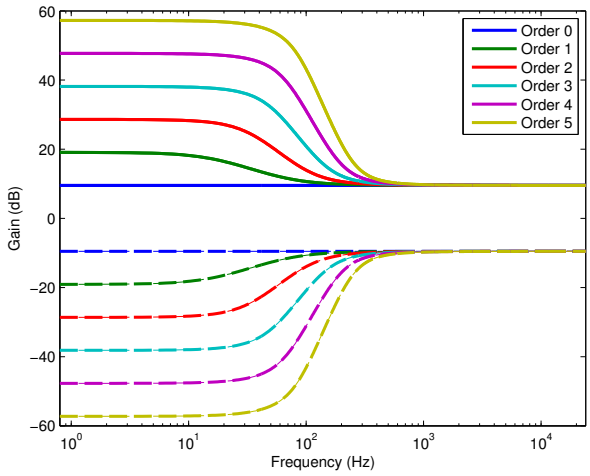


Figure 5: Gain of the frequency response function of filters $\frac{r_0}{r_1}\frac{F_{m,r_1}}{F_{m,r_0}}(f)$ for $r_1 = 1$ m, $r_0 = 3$ m (solid lines) and $r_1 = 3$ m, $r_0 = 1$ m (dashed lines), $m \in \{0, 1, 2, 3, 4, 5\}$, $F_s = 48000$ Hz.

```
  R(0)        = !;
  R(float(0)) = R(0);
  R(float(1)) = R(1);
  R(c)        = *(c);
};

matrix(n,m) = par(i,n,_)
<: par(i,m,gain(a(i)):>_);
// n: number of inputs : column number
// m: number of output : row number
// a(i): row vectors of matrix coefficients
```

In Eqs. (5) and (7) the matrix $\mathbf{Y}_{\text{mic}}^T \cdot \mathbf{W}_{\text{mic}}$ and $\mathbf{W}_{\text{spk}} \cdot \mathbf{Y}_{\text{spk}}$ are pre-computed and then implemented numerically in the FAUST code, row by row, according to the code above. In the current version, the encoding matrix for 2 types of microphone is coded:

- Spherical microphone using Lebedev's grids working up to 1^{st}, 3^{rd} or 5^{th} order as shown in Fig. 9.
- Spherical microphone using Pentakis-Dodecahedron grid working up to 4^{th} order as shown in [3].

In Fig. 3, the obtained modules are sketched under the names "Encoding" and "Decoding".

4.3. Rigid spherical microphone filters

As mentioned in Sec. 3.1.2, the $B_{mn}^{\sigma}(z)$ components derived from a rigid spherical microphone array signals should be filtered by $E_m(z)$ filters to take into account the diffraction of the rigid sphere supporting the capsules. Theses filters present a theoretical infinite gain at 0 Hz and very large "bass-boost" for high orders [24]. They are stabilized by high-pass filters [25] or Tikhonov filters [3], which cut the low frequencies at high orders. Resulting filters are implemented as FIR filters. However, in [26, 27, 24] IIR implementations are proposed with lower orders amplification according to higher orders limitation. These approaches should be investigated in future works for a FAUST implementation since an IIR parametric filters could be interesting to monitor in real-time the performances of a spherical microphone array. For now, in the reported implementation, the filters are implemented as FIR filters. The cut-off frequencies of high-pass filters are chosen with a maximum amplification level. The real-time convolution is made with BRUTEFIR[3] for Linux environment. This module is sketched in Fig. 3 under the name "Equalization".

4.4. Encoding of virtual source and panning law

In the FAUST implementation reported in this paper, the encoding of a virtual source is based on Eq. (3) for a plane wave and Eq. (13) for a spherical wave. In the current state of the implementation, the spherical harmonics are explicitly coded in a `ymn.lib` library up to order five (36 functions). However, they could be computed by recurrence if higher orders would be required. The monophonic

[3] `http://www.ludd.luth.se/~torger/brutefir.html`

signal a_1 is multiplied by corresponding spherical harmonics evaluated at desired direction (θ_1, δ_1) and filtered by near-field filters with desired radii (r_1, r_0) as shown in Fig. 3: Ambisonics signals are then obtained.

The equivalent panning laws of Eq. (8) circumvent the need to encode and decode with matrix operations. Indeed, the computation is reduced to a sum on the orders thanks to the additivity theorem of spherical harmonics. This is of great interest for real-time synthesis. For the moment, the panning-laws are implemented for several Lebedev spheres using $N = 6, 26$, or 50 loudspeakers, as presented in [16]. The Legendre polynomials P_m are explicitely coded in `ymn.lib` up to order five. The weights w_l and angles θ_l, δ_l of the spheres are implemented in a `lebedev.lib` file. This spatialization module is sketched in Fig. 3 under the name "Panning-law": the loudspeakers input signals are computed from a monophonic signal a_2 and spatial parameters $(r_2, \theta_2, \delta_2, r_0)$.

5. VISUAL FEEDBACK WITH PROCESSING

FAUST provides a graphical user interface with VU-Meters. Unfortunately, those meters are organized as classical lines of VU-Meters (see Fig. 7 as an example). Thus, they do not provide an efficient visual cue of where the signal energy is distributed on the loudspeaker spherical layout. Moreover, a 3D representation of the virtual source helps to provide an objective spatial view of the source position. Such type of 3D representation is also much more useful to composers or engineers who are not familiar with under-the-hood DSP. In this context, a visual tool for 3D visual feedback was implemented using PROCESSING[4] language. The code uses an `Atom` class to create a ball representing a loudspeaker. The loudspeakers coordinates are given in a `.cvs` file and are easily adaptable to any configuration. The virtual sources are represented as balls with fading trajectories. The RMS gain (in dBFS) of each loudspeaker given by the Ambisonics solution drives in real-time the size and the color of the loudspeaker's-balls via OSC messages. The position of each source is also received by OSC messages. The `Peasycam` library[5] allows to zoom, move and pan the 3D view. The described visual tool is shown in Fig. 6. The sphere used here is a Lebedev sphere using $N_2 = 50$ loudspeakers as used in [16].

6. USER CONTROL INTERFACE

6.1. Controls by software

The user control interface is provided by FAUST compiled code. Thus, depending on the application it could be a standalone application, a MAX/MSP patch, VST or LV2 plugin, or others. It consists in sliders and entry boxes, to control the parameters $(r_1, \theta_1, \delta_1, r_0)$. Check-boxes are used to mute an order m. VU-Meters give the signals level in dBFS for the $B^{\sigma}_{mn}(z)$ components or s_l.

As an example, a JACK[6] standalone application interface for real-time 3D spatialization using the panning law of Eq. (13) is shown in Fig. 7.

[4] `https://processing.org/`
[5] `http://mrfeinberg.com/peasycam/`
[6] `http://jackaudio.org`

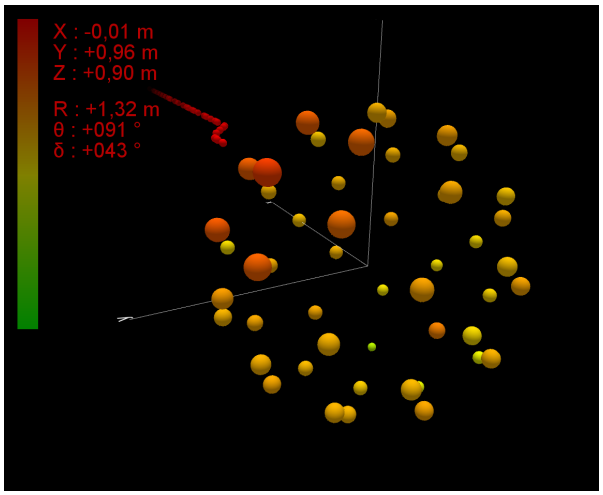


Figure 6: 3D visual feedback tool using PROCESSING language. The virtual source is shown as a red ball with fading trajectory (top left corner). Each loudspeaker is represented by a ball with radius and color driven by loudspeaker signal level (RMS, dBFS). 0 dBFS correspond to the red color on the left color scale and $-\infty$ dBFS to green. All informations are received by OSC messages generated by a joystick and a PURE DATA patch.

6.2. Controls by hardware device and OSC messages

FAUST supports OSC to control the parameters of the DSP tools. For example, with a hardware controller and a PURE DATA patch, it is possible to generate OSC messages which control the plug-in as well as the visual feedback tool of Sec. 5. This correspond to the rightmost branch of Fig. 3. The use of hardware controllers and FAUST generated plug-ins allow for the easy control and recording of the synthesis parameters. As an example, the implemented Ambisonics encoder loaded in ARDOUR[7] Digital Audio Workstation (DAW) as a LV2 plug-in is shown in Fig. 8. With the visual feedback tool described in Sec. 5, this configuration enables to record automation tracks describing "manually-created" trajectories.

7. EXPERIMENTAL SETUP

An experimental setup is presented briefly in this section as an application of the tools presented above.

7.1. 3D sound pressure field capture

The recording of natural sound pressure fields is made with a rigid spherical microphone "MemsBedev"[8] shown in Fig. 9. This microphone was made by 3D printing and uses $N = 50$ microphone capsules on a Lebedev grid [28]. Each capsule is made of 4 MEMS[9] microphones to reduce the background noise. The microphone works up to 5^{th} order, as explained in Ref.[16]. The analog signals are digitalized with a commercial front-end. The tools presented in this article can provide in real-time the Ambisonics components $B^{\sigma}_{mn}(z)$ up to 5^{th} order according to Eq. (5) and Fig. 3.

[7] `http://ardour.org/`
[8] `http://www.cinela.fr/`
[9] MEMS: Micro Electro-Mechanical System

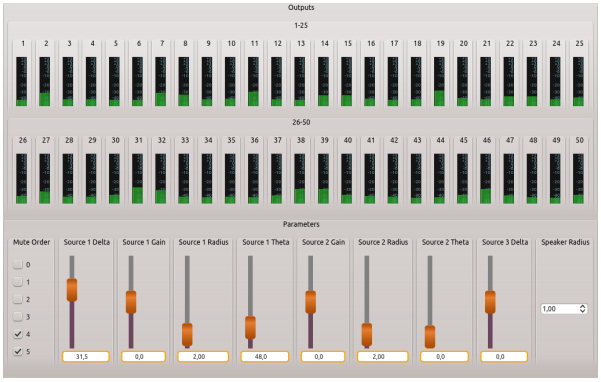


Figure 7: Standalone application interface running under JACK. The sliders allow controlling the gains and positions of two virtual sources (bottom, grey and yellow). The check-boxes (bottom-left) mute and unmute a specific order at the synthesis stage. The entry box allows giving the r_0 radius of the rendering spherical loudspeakers layout (bottom right). The VU-meters give the signals levels in dBFS for each loudspeaker. In this case there are $N_2 = 50$ loudspeakers according to a Lebedev grid.

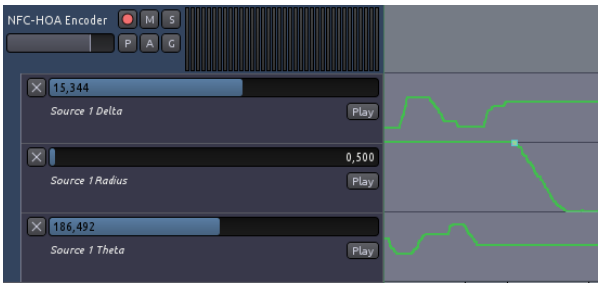


Figure 8: Ambisonics encoder loaded in ARDOUR as a LV2 plug-in. The positions parameters are controlled with the mouse or by an hardware joystick via OSC messages. Resulting automation tracks are shown in green in this figure. The LV2 plug-in is generated from FAUST code using `faust2lv2`.

7.2. 3D sound pressure field synthesis

The decoding of the pressure sound field is made in real-time with a decoder made with FAUST according to Eq. (7) and Fig. 3. The Lebedev spherical loudspeaker layout used for the sound field rendering is shown in Fig. 10. For recording rendering purposes, it is possible to record in 3D in one place and render in real-time the 3D sound pressure field with the loudspeaker sphere located at another location. For spatialization purposes, the user takes place in the sphere and drives the position of the virtual sources using an hardware controller. The hardware controller is linked to a PURE DATA patch and generates OSC messages for the synthesis using FAUST, as summarized in Fig. 3. This experimental setup can thus help to mix 3D audio contents *in situ*.

Figure 9: MemsBedev microphone arrays with $N = 50$ MEMS microphones

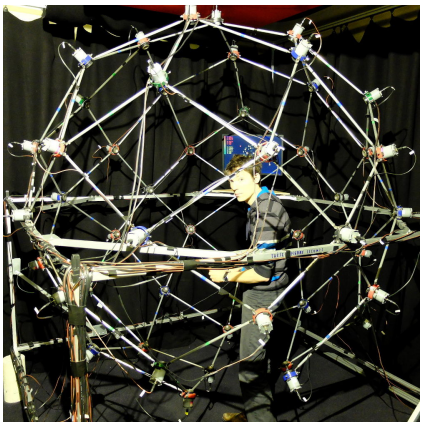


Figure 10: Lebedev spherical loudspeakers layout with $N_2 = 50$ loudspeakers.

8. CONCLUSION

Several integrated tools for real-time 3D sound pressure field synthesis using Ambisonics were developed and presented in this paper. The signal processing was implemented using FAUST, the visual feedback with PROCESSING and the communication between tools with OSC protocol. The near-field filters were implemented in a FAUST library and allow synthesizing spherical waves. The gain correction with distance is also implemented, which is of great importance for engineering applications or simulators. In the current version of the implementation, the synthesis is controlled up to 5^{th} order in 3D. Some future ameliorations could include transformations in Ambisonics domain [29], recent advances in radial filters implementation [27] or inclusion of the Doppler effect for moving sources [30]. The code of this project is freely available online under GPL license[10]. In a near future, the described implementation and experimental setup will be used to reproduced various recorded environments and achieve physical evaluation of the reproduced sound fields.

[10] `https://github.com/sekisushai/ambitools`

9. REFERENCES

[1] Jérôme Daniel, *Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia*, Ph.D. thesis, Université Paris 6, Paris, 2000.

[2] Jérôme Daniel, "Spatial sound encoding including near field effect: Introducing distance coding filters and a viable, new ambisonic format," in *Audio Engineering Society Conference: 23rd International Conference: Signal Processing in Audio Recording and Reproduction*, Helsingør, 2003, pp. 1–15, AES.

[3] Sébastien Moreau, Jérôme Daniel, and Stéphanie Bertet, "3d sound field recording with higher order ambisonics-objective measurements and validation of spherical microphone," in *Audio Engineering Society Convention 120*, Paris, 2006, pp. 1–24, AES.

[4] Matthias Kronlachner, "Plug-in Suite for mastering the production and playback in surround sound and ambisonics," in *Linux Audio Conference*, 2013.

[5] Christian Nachbar, Franz Zotter, Etienne Deleflie, and Alois Sontacchi, "Ambix - A suggested ambisonics format," in *Ambisonics Symposium, Lexington*, 2011.

[6] Aaron J. Heller and Eric M. Benjamin, "The Ambisonic Decoder Toolbox: Extensions for Partial-Coverage Loudspeaker Arrays," in *Linux Audio Conference*, 2014.

[7] Julien Colafrancesco, Pierre Guillot, Eliott Paris, Anne Sèdes, and Alain Bonardi, "La bibliothèque HOA, bilan et perspectives," in *Journées d'Informatique Musicale*, 2013, pp. 189–197.

[8] Matthias Geier and Sascha Spors, "Spatial Audio with the SoundScape Renderer," in *27th Tonmeistertagung–VDT International Convention*, 2012.

[9] Darren B. Ward and Thushara D. Abhayapala, "Reproduction of a plane-wave sound field using an array of loudspeakers," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 6, pp. 697–707, 2001.

[10] Philippe-Aubert Gauthier, Cédric Camier, Thomas Padois, Yann Pasco, and Alain Berry, "Sound Field Reproduction of Real Flight Recordings in Aircraft Cabin Mock-Up," *J. Audio Eng. Soc*, vol. 63, no. 1/2, pp. 6–20, 2015.

[11] Anthony Bolduc, Philippe-Aubert Gauthier, Telina Ramanana, and Alain Berry, "Sound Field Reproduction of Vibroacoustic Models: Application to a Plate with Wave Field Synthesis," in *Audio Engineering Society Conference: 55th International Conference: Spatial Audio*, 2014.

[12] Michael Vorländer, *Auralization: fundamentals of acoustics, modelling, simulation, algorithms and acoustic virtual reality*, Springer Science & Business Media, 2007.

[13] Estelle Bongini, *Modèle acoustique global et synthèse sonore du bruit d'un véhicule: application aux véhicules ferroviaires*, Ph.D. thesis, Université de Provence-Aix-Marseille I, 2008.

[14] Yann Orlarey, Dominique Fober, and Stéphane Letz, "FAUST: an efficient functional approach to DSP programming," *New Computational Paradigms for Computer Music*, vol. 290, 2009.

[15] Boaz Rafaely, "Analysis and design of spherical microphone arrays," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 1, pp. 135–143, Jan. 2005.

[16] Pierre Lecomte, Philippe-Aubert Gauthier, Christophe Langrenne, Alexandre Garcia, and Alain Berry, "On the use of a Lebedev grid for Ambisonics," in *Audio Engineering Society Convention 139*, New York, 2015.

[17] Boaz Rafaely, Barak Weiss, and Eitan Bachmat, "Spatial aliasing in spherical microphone arrays," *IEEE Transactions on Signal Processing*, vol. 55, no. 3, pp. 1003–1010, 2007.

[18] Mark A. Poletti, "Three-dimensional surround sound systems based on spherical harmonics," *Journal of the Audio Engineering Society*, vol. 53, no. 11, pp. 1004–1025, 2005.

[19] George B. Arfken and Hans J. Weber, *Mathematical methods for physicists - sixth edition*, Elsevier, 6th edition, 2005.

[20] Fons Adriaensen, "Near field filters for higher order Ambisonics," http://kokkinizita.linuxaudio.org/papers/hoafilt.pdf, 2006, Accessed: 2013-10-28.

[21] Sascha Spors, Rudolf Rabenstein, and Jens Ahrens, "The theory of wave field synthesis revisited," in *Audio Engineering Society Convention 124*, Amsterdam, 2008, pp. 1–19, AES.

[22] Alan V. Oppenheim and Ronald W. Schafer, *Digital signal processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975.

[23] Jens Ahrens and Sascha Spors, "Focusing of virtual sound sources in higher order Ambisonics," in *Audio Engineering Society Convention 124*, Amsterdam, 2008, pp. 1–9, AES.

[24] Stefan Lösler and Franz Zotter, "Comprehensive radial filter design for practical higher-order Ambisonic recording," *Fortschritte der Akustik, DAGA*, 2015.

[25] Jérôme Daniel and Sébastien Moreau, "Further study of sound field coding with higher order ambisonics," in *Audio Engineering Society Convention 116*, Berlin, 2004, pp. 1–14, AES.

[26] Hannes Pomberger, "Angular and radial directivity control for spherical loudspeaker arrays," 2008.

[27] Robert Baumgartner, Hannes Pomberger, and Matthias Frank, "Practical implementation of radial filters for ambisonic recordings," *Proc. of ICSA, Detmold*, 2011.

[28] V.I. Lebedev, "Values of the nodes and weights of quadrature formulas of Gauss-Markov type for a sphere from the ninth to seventeenth order of accuracy that are invariant with respect to an octahedron," *USSR Computational Mathematics and Mathematical Physics*, vol. 15, no. 1, pp. 44–51, 1975.

[29] Matthias Kronlachner and Franz Zotter, "Spatial transformations for the enhancement of Ambisonic recordings," in *Proceedings of the 2nd International Conference on Spatial Audio, Erlangen*, 2014.

[30] Gergely Firtha and Peter Fiala, "Sound Field Synthesis of Uniformly Moving Virtual Monopoles," *Journal of the Audio Engineering Society*, vol. 63, no. 1/2, pp. 46–53, 2015.

A TOOLKIT FOR EXPERIMENTATION WITH SIGNAL INTERACTION

Øyvind Brandtsegg
Department of Music,
Norwegian University of Science and Technology
(NTNU)
Trondheim, Norway
oyvind.brandtsegg@ntnu.no

ABSTRACT

This paper will describe a toolkit for experimentation with signal interaction techniques, also commonly referred to as *cross adaptive processing*. The technique allows analyzed features of one audio signal to inform the processing of another. Earlier used mainly for mixing and post production purposes, we now want to use it creatively as an intervention in the musical communication between two performers. The idea stems from Stockhausen's use of *intermodulation* in the 1960's, and as such we might also call the updated technique *interprocessing*. Our interest in the technique comes as a natural extension to previous research on *live processing* as an instrumental and performative activity. The automatic control of signal processing routines is related to previous work on adaptive audio effects and automatic mixing. The focus for our investigation and experimentation with the current toolkit will be how this affects the musical communication between performers, and how it changes what they *can and will* play. The program code for the toolkit is available as a github repository[1] under an open source license.

1. INTRODUCTION

The Music Technology section at NTNU Department of Music has researched *live processing* as an instrumental activity for music performance, for example in the ensemble T-EMP (Trondheim Ensemble for Electroacoustic Music Performance). In this ensemble, new modes of improvisation and music making have been explored, utilizing the possibilities inherent in contemporary electroacoustic instrumentation. One particularly interesting aspect of this research is the manner in which live processing affects the communication and interplay between performers. To enhance the focus on possible interventions in this communication, we look into a more direct signal interaction, where analyzed features of one signal are used to control the parameters of processing for another. The term *interprocessing* is derived from Stockhausen's use of *intermodulation*[1] in the 1960's, and is here used to describe any direct signal interaction where features of one signal is allowed to affect the processing of another signal. Where Stockhausen's intermodulation was mainly applied as amplitude modulation, we would like to expand the signal interaction to allow any kind of processing technique, and also include a wide selection of analyzed features from the controlling signals as parametric inputs to the process. With respect to the sonic interaction of two audio signals, this also ties into our earlier work on dynamic convolution [2] and cross convolution techniques [3]. The objective is to find ways of close interaction and sonic merging by enabling musical performative actions of one performer to directly influence the sound of other instruments in the ensemble. As a practical example, say we would let the spectral characteristics of a banjo affect control the immediate filtering of a saxophone, or the noise content of the drums to affect the reverberation of the vocals. Combining analyzed features from several signal allows for a tightly interwoven timbral ensemble interaction (see figure 1).

The toolkit utilize automated control of effect processing parameters, where the automation is based on analyzed features of an audio signal. In this respect it ties closely with the field of adaptive audio effects [4], [5], adaptive modulation, [6], [7] and automatic mixing via cross adaptive techniques [8], [9], [10], [11], [12]. In terms of the instrumental control of the processing it also relates to [13], [14] and [15].

In addition to the signal interaction potential, the toolkit also naturally allows features of a signal to affect its own processing. This can be useful a starting point of experimentation with the analysis signal mappings, and can also be envisioned to yield useful adaptive effects control mappings for studio and post production type effects. The analysis methods and the effect processing methods used in the toolkit are well known from existing DSP literature, it is the configuration as a tool for live and performative experimentation with cross-adaptive effects control that constitute the new or added value of the work. This is intended to allow the mindset of the programmer to be set aside, focusing more on an intuitive and empirical approach for musical experimentation with the techniques. Such experimentation can also lead to a deeper understanding of the analysis techniques involved, and as such may be useful practice for researchers within the field of audio analysis.

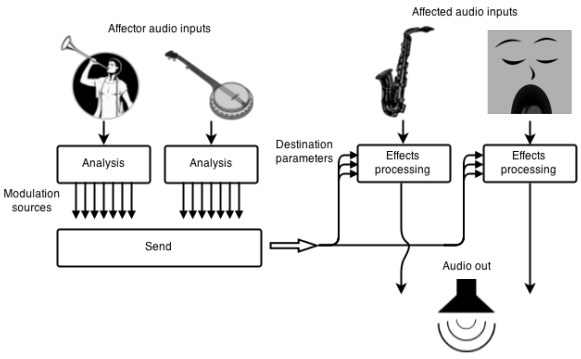


Figure 1: Distribution of analysis signals to several effects

[1] https://github.com/Oeyvind/interprocessing

2. TOOLKIT REQUIREMENTS

The need for a software environment to host and facilitate this kind of audio processing has been suggested by [10] and [16], the latter also providing an example solution within the framework of Max/MSP. For the design of the current toolkit, we have looked specifically at the integration of the cross adaptive techniques into a standardized DAW so that it can easily be combined with other techniques and workflows. The use of open source components has also been preferred.
Since the use of extended signal interaction for live processing touches on musically unfamiliar territory it seems reasonable to make a toolbox for experimentation, and to enable it to be used in such a way that allows the focus of experimentation to remain on the aesthetics and musicality of live processed sound, both in terms of the compositional and the performative. The toolkit is not designed for the general music software end user at this point. This is something that can be built later as a result of experimentation on usability of the different signal interactions. To allow for experimentation the toolkit should be very flexible with regards to configuration and routing of the signals. It should also allow for an intuitive workflow, despite the relatively high number of possible parameter connections/mappings. The toolbox should be easily integrated with other tools, so a method of interfacing with a selection of regular DAWs is strongly preferred. Using a regular DAW as a host program also provides "bread and butter" functionality like a GUI, audio i/o, audio and cpu metering, and audio routing. Methods for analysis of the input signal features has been implemented, as well as routing and mapping of the analysis signals. Integration with an existing set of effects processor implementations is to be preferred in our context, and we have looked at methods to adapt existing effects to allow parametric control by the analysis signals. A methods for integration with standard VST effects has also been implemented. The toolkit is open source and available on several platforms (Linux/Windows/OSX). The audio programming language Csound[2] was selected as the implementation language due to its extensive library of audio processing routines, and also since it fits the requirements of integration with a DAW (via Cabbage[3]) and it is open source and cross platform. For easy integration with a wide selection of DAWs, the toolkit has been implemented with VST wrappers to allow the processors to be used as regular VST plugins.

3. ROUTING

This section will look at possible options for signal routing as a background for the currently chosen model. This relates to the routing of both audio and analysis signals. In the signal interaction, we have an *affected* signal and an *affector* signal. The affected signal is the one where effects processing is being applied, according to analyzed features of the affector signal. Following the conventions of an audio mixer, we would route the signal to be affected into a channel strip and apply an insert effect on this strip. One thing to note here is that the affected signal can be seen as a single source, but the affector signals may come from several audio sources (i.e. using analyzed features of many sources to selectively affect the processing). We can use the term *modulation sources* for the signals resulting from the analysis of the affector signal. Similarly, we can use the term *destination parameters* to refer to the parameters of the effect processing done on the affected signal.

3.1. Sidechaining

Signal interaction can be found in a conventional audio production signal chain in the form of *sidechaining*. Most commonly, it is used for dynamic processing, for example the genre-typical sidechaining compressor of electronic dance music. Here, the kick drum is used to "duck" the synth pads (or a whole submix), creating a rhythmic pumping effect. Another well known application is ducking of background instruments to give precedence to vocals. A form of sidechaining is also used in de-essers, but in that case the sidechain signal is not an independent audio input but a filtered version of the signal being processed. As a general method for our toolbox, sidechaining has the advantage that it is commonly used and most audio mixers have functionality to provide the appropriate signal routing. The disadvantage is that in general, the sidechain input is a single source. The analysis of the sidechain input will also have to be done in the effects processor for the affected signal, possibly duplicated in other effects processors using the same sidechain source. Due to these limitations, another routing model was needed.

3.2. Dual mono

A variant of the sidechain method would be to create special effects processors with dual mono inputs. One input would be the affected signal and the other the affector signal. This method has similar advantages and drawbacks as the sidechaining method. It could be a viable solution where regular sidechaining is problematic for some reason, but it would require a type of signal routing that many would find counter-intuitive or downright complicated (putting the effect on an aux track, using aux sends and panoramic controls to route the two source signals accordingly). Both the sidechaining and the dual mono configurations may be relevant formats for plugins tailored to a specific set of signal interactions, presumably something that might result from this initial phase of experimentation. Such variants could have the analysis methods inlined in the same plugin, potentially reducing the latency between a feature change in the affector signal and a processing change in the affected signal. The analysis would however, in general be limited to features from one single affector signal. A practical solution would be to implement multichannel VST plugins on multichannel tracks as for example available in Reaper[4], that would however greatly reduce the choice of DAW to use as a host. The issue of duplicated efforts regarding the analysis stage also apply to this model, so it is not the most effective and flexible method for routing and mixing of the control signals. For some traditional spectral interaction techniques, like cross synthesis or cross convolution [3], this routing scheme may still be as good solution, due to the necessity of synchronized frame-by-frame processing.

3.3. Control signal layer

If features of the same affector signal is to be used to control several different affected signals, it might seem reasonable to keep all signal analysis in one place to relieve the system of duplicated efforts. This requires a method of sending analysis signals from one

[2] http://www.csounds.com
[3] https://github.com/rorywalsh/cabbage/releases
[4] http://www.reaper.fm/

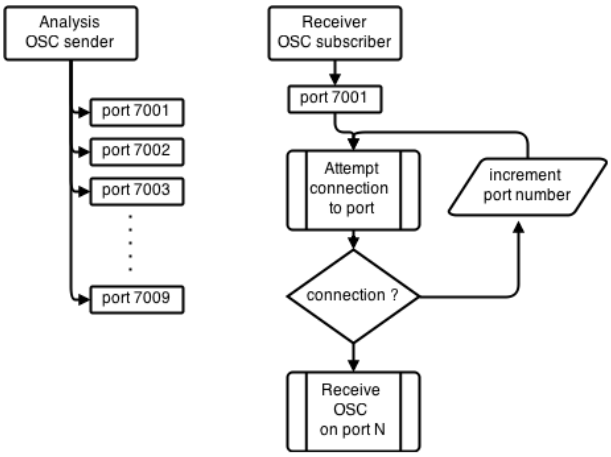


Figure 2: Sending Open Sound Control messages to several network ports on the same computer

process (e.g. audio track in the DAW) to another. Even though this is not implemented as standard in any DAW (with the exeption of using a simple envelope follower as a parametric automation, available in Reaper and other hosts), it can be implemented with relatively widespread tools of interprocess communication (Open Sound Control, ZMQ, named pipes etc). The implementation of a custom routing system as we have done in the toolkit allows for combination of features from several affector signals, in scalable proportions and mappings. With respect to control parameter mapping this is analoguous to the *many to many* mapping described by Hunt et al in [17]. Combining features from several signals in this manner opens up for fine tuned complex interactions between musical signals, but will also require particular care in designing appropriate and musically effective mappings. The provision of a control signal layer to easily experiment with different mappings is assumed to facilitate the design of appropriate settings. To minimize the need for installing third party libraries, OSC was selected as the communication protocol. As we want to distribute the analysis signals (from the affector) to several effects processors (affected signals, also called destinations in the following) we needed to devise a simple form of multicast for the OSC messages. Since the destinations may well reside on the same computer, and the OSC signals are transported via network sockets, we need to open a separate network port for each destination, as only one process can read from a network port at any given time. A somewhat crude solution for this is to send all analysis signals (from all affectors) to a pre-selected series of network ports. Each receiver (destination) process will then scan for an available network port on startup, opening the first available port (figure 2). Additional receivers will scan similarly and open the next available port on startup. In this manner, all analysis signals are available to all receiver plugins, and we can use address filtering to select the signals to be actively used in the effect processing of the affected signal. The method also allows the analysis signals to be available as OSC messages freely routable in the host DAW, and in this manner the toolkit enables cross adaptive control of any VST plugin. The toolkit includes routines for scaling, shaping and mixing of the analysis signals (outlined in section 5), to use these routines in the mapping to any standard VST effect, we provide a special translator plugin (see section 8).

4. SIGNAL ANALYZER PLUGIN

The signal analyzer plugin provides a selection of analysis routines. In addition to the amplitude (rms) analysis, we have used spectrally based analysis methods from the timbre toolbox [18] as well as selection of pitch tracking methods. The timbral analysis methods include *centroid, spread, skewness, kurtosis, flatness, crest and flux*. The centroid represents the spectral center of gravity, while the spread represents the spread of the spectrum around its mean value. The spectral skewness gives a measure of the asymmetry of the spectrum around its mean value, while kurtosis gives a measure of the flatness of the spectrum around its mean value. These parameters (centroid, spread, skewness and kurtosis) are commonly referred to as the first four statistical moments of the spectrum. Further, the spectral flatness measure is obtained by comparing the geometrical mean and the arithmetical mean of the spectrum, it gives an indication of the balance between tonal and noisy components in the sound. Similarly, the spectral crest is obtained by comparing the maximum value and arithmetical mean of the spectrum. Finally, the spectral flux represents the amount of variation of the spectrum over time. The timbre toolbox paper (ibid.) also describes several other analysis measures. Selection of the parameters to be used in our study here has been made in part on the basis of which measures can be calculated in a real-time streaming manner (on a frame by frame basis, without knowing the whole duration or evolution of the sound). An educated guess about the expected usefulness and redundancy of the different measures in our context has also affected the selection.

Figure 3: The analyzer plugin GUI

The pitch tracking methods are partly borrowed from the repertoire of Csound, and an additional method has been implemented based on *epoch analysis* [19]. The pitch tracking methods from the Csound repertoire is *ptrack, plltrack, and pitchamdf*. The ptrack method use a STFT method and extracts an estimated pitch for its fundamental frequency, based on an original algorithm by M.Puckette et. al [20]. The plltrack method use a phase-locked loop algorithm based on [21]. The pitchamdf method uses an average magnitude difference function [22]. The active pitch tracking method can be chosen from the GUI of the analyzer plugin (figure 3), and the effective pitch tracking result can be monitored as a sine wave audio output from the analyzer. The pitch monitor signal can be turned on or off via a GUI control. To enable a stable pitch tracking output, median filtering has been applied to the pitch tracker output. The size of this median filter can be adjusted in the GUI (*pitch filter size*)

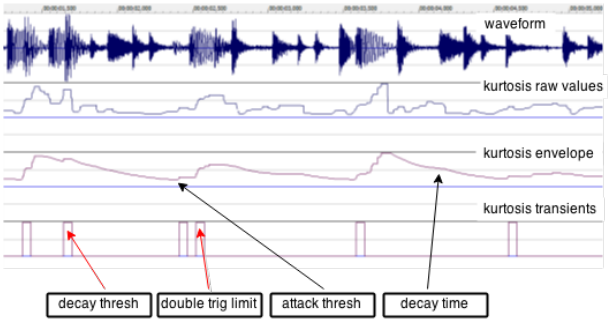


Figure 4: Transient detection, here used on kurtosis.
Arrows in red indicating transients that could be filtered out by raising the parameter values

Transient indicators for amplitude, centroid and spectral kurtosis are extracted using rate of change analysis. The signals are conditioned with an envelope follower filter and mapped to a perceptual scale (e.g. dB for amplitude) before transient detection. The sensitivity of the transient detection can be adjusted with an *attack threshold* parameter. To limit the amount of false trigging, some filtering methods have been implemented. When a transient is detected, the current signal level is recorded, and a *decay threshold* sets the relative negative change needed in the input signal before a new transient is allowed to be registered. The envelope filtering on the signal to be analyzed has an adjustable *decay time* to smooth out fluctuations after a peak in the signal, and this works in tandem with the decay threshold to reduce the amount of false positives. Finally, a timer is used as a secondary means to limit the rate of transients, ensuring that a certain (*double trig limit)* amount of time must pass after a transient has been recorded before another transient is allowed (figure 4). The transient detection algorithm has been developed empirically by the author, inspired by numerous sources over the last few years. It was originally devised for detecting amplitude transients, but here adapted to also work on other kinds of signals. The pitch transients has separate triggers for upwards and downwards pitch change. The aforementioned envelope filtering is then duplicated to create the two different envelopes needed. The detection parameters are the same for upwards and downwards pitch transients, and the detection threshold is in semitones (pitch change needed before issuing a trigger). However, due to inherent weaknesses of the pitch tracking algorithms, the transient detection parameters must be regarded as candidates for empirical adjustment.

5. SCALING, SHAPING, TRANSLATION

5.1. Normalization

Due to the fact that the different analysis track can produce signals within widely varying ranges, normalization of the analysis signal is done before it being sent from the analyzer plugin (figure 5). This allows signals to be interchanged and routed more freely without too many surprises due to out of range values. The analysis response will of course vary significantly on different input sounds with widely differing characteristics, so the said normalization is a trade off aquired by empirical testing. The purpose of the normalization is as far as possible to keep the ranges of the different analyzed features within the same range. Features like skewness, kurtosis and flatness may still vary to an extent that no all-purpose solution has been found. Special treatment is done on the pitch tracking, where two versions of the signal are created. One version is simply normalized by dividing by the max pitch value, another is divided by the effective pitch range and offset with the minimum pitch (so as to create a more full range 0.0-1.0 normalized signal). These signals are called "cps" and "pitch" respectively. The normalized signals will be scaled to the appropriate range for the destination parameters on the receiving side. As an additional convenience, the raw pitchtracking values are available ("cps_raw" parameter) for straightforward routing of pitch to e.g. filter cutoff frequencies. Use of the raw valued parameter selectively bypasses normalization in the analyzer and also bypasses autoscaling (see section 5.4) to the destination parameter range in the receiver plugin.

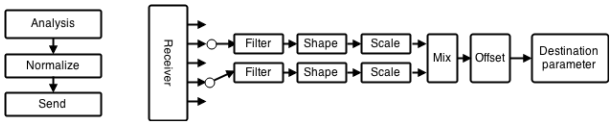


Figure 5: Normalization, signal conditioning, mixing of different modulation sources, offset and route to destination.

5.2. Filters

Some of the analysis tracks contains transient triggers (for amplitude, pitch tracking, centroid and spectral kurtosis). Since these signals are very brief pulses, an envelope generator is triggered in the receiver plugin upon reception of these signals. The rise (attack) and fall (decay) times can be set in the GUI (see figure 6 for an excerpt showing a single destination parameter). For continuous signals, a similar kind of filtering is implemented, making a smoothing filter with separate rise and fall times. The algorithm for the smoothing filter is lifted from Csound's *follow2* opcode, which in turn attributes the algorithm to Jean-Marc Jot. The filter is an exponential moving average, with different coefficients used for rising and falling slopes. It can be described as:

$$Y_N = X_N + (C * (Y_{N-1} - X_N))$$

where

$$C = \begin{cases} 0.001^{(1/(\tau_a * f_s))} & \text{if } (X_N > Y_{N-1}) \\ 0.001^{(1/(\tau_b * f_s))} & \text{otherwise} \end{cases}$$

where τ_a is the rise time and τ_b is the fall time

Envelope generator or filter is selected automatically according to signal type. The filtering is done prior to scaling, as the scaling may invert the sign of the modulation signal and it was assumed it would be more intuitive to control the rise and fall time with respect to the input signal before the (possible) inversion.

5.3. Shaping

After filtering, the signal is shaped (also called *warping* in [23]) by a curve parameter. This is to allow a dynamic and gradual change between log/linear/exponential mappings. The shaping can be described by this algorithm:

$$Y = (1 - \exp(X * curve))/(1 - \exp(curve))$$

Where we use a range of -5.0 to 5.0 for the curve parameter. A curve value of close to zero yields a linear mapping (no shaping), but note that an actual curve value of zero will have to be handled by an exception. A curve value of 1.0 approximates an exponential mapping, while larger values provide increasingly steeper curves. Similarly a curve value of -1.0 approximates a logarithmic mapping, with an increasingly steeper curve for higher negative values.

5.4. Scaling and offset

Each modulator signal can be scaled to set the degree of modulation to the destination parameter. The scaling is done on the receiving side, so that individual scaling can be set for each modulation source to each destination parameter. The modulator range is automatically scaled to the range of the destination parameter via global variables for min and max, so that a normalized modulator signal should be able to use the full range of the destination parameter. In special cases the modulator signal may have a smaller range, depending on the characteristics of the analysis input signal. For these cases, the scaling can be boosted by an additional switch (x10). The offset for each parameter is normally in the same range as the min and max values for the parameter. However, with some routing/mappings, we might want to extend the offset range (e.g. if the mapping of the affector signal constantly makes it go out of range). For this purpose, additional switches has been added to the offset setting, allowing it to extend its range to +/- 1x the original range.

Figure 6: Example of destination parameter GUI

6. EXAMPLE PROCESSORS

Some example processors have been implemented to start working with the toolkit. Little is known as to what type of effects may be musically useful for live cross-adaptive processing, which is also part of the incentive for making a toolkit for experimentation. Some effects has been chosen due to a clearly identifiable or obvious relationship between parameter variations and sonic results. For example *stereo panner, tremolo/AM, and a lowpass filter with distortion*. Other effects has been chosen due to expectations of musical expressiveness, like *time modification* (by means of phase-locked vocoder processing[5]), *stereo delays and reverb*. Yet another type is effects that has the potential of strongly imprinting sonic characteristics (from the modulator) onto the processed sound, for example convolvers and physical models. In this category, a simple waveguide was implemented, with the audio input to the effect being used to excite the physical model. If the fundamental frequency of the waveguide is modulated for example by the pitch of the modulating signal, we get a tight sonic interaction between the two audio inputs. Each of the above effects in and of itself may provide less than exciting musical results, but the combination of several effects modulated by several different characteristics of the modulator signal seems to have the potential for a rich and multi-dimensional sonic interaction.
The program code for these effects can be found in the github repository[6]. In addition it may be useful to implement a selection of granular delays and transformation effects, flanger/chorus, pitch modulation, spectral panners and other spectral modulations, and dynamic convolver effects, to name a few.

7. ADAPTING EXISTING EFFECTS AND UPDATING THE TOOLKIT

Making a script to automatically modify existing effects implemented in Csound would be handy. However, if such a script should be able to take any implemented effect and modify it to become a signal-interaction-enabled effect we would have to make assumptions about how the parameter control was implemented in the effect to be modified. Rather than making such assumptions, we have made a Python script (*codeUtility.py*) that automatically generates essential include files and also generates the relevant parts of the GUI widget code. The code repository provides a template Csound file for this purpose. To modify an existing effect, one will have to make a list of the control parameters and their associated range. This can be entered as a list into the python script codeUtility.py, and this script will generate the relevant code (when run with python codeUtility.py *effectName*). The GUI code will have to be copied and pasted into the new effect, and the header and score section of the file needs to be modified according to the template effects file. The necessary modifications has been marked with comments in capital letters in the template.csd file. Python writes the GUI code to *effectName*_gui_scratchpad.inc, from where it can be copied into the csd. The GUI caption and plugin id (line 19 in the template.csd) should also be edited to reflect the newly created effect. As the toolkit is still in it's early stages of development, it is highly probable the parameter set of the analysis needs to be updated and expanded. Additional analysis methods should be put in the analyze_audio.inc file. The *readme* file in the repository provides

[5] http://www.csounds.com/manual/html/mincer.html
[6] https://github.com/Oeyvind/interprocessing

additional details as to what components need to be updated to allow the new parameter set to be picked up by the system.

8. CONNECTING TO STANDARD VST EFFECTS

The OSC messages from the analyzer can be used to control standard VST plugins or a host program parameter. Flexible modern DAWs provide mapping options for OSC messages to any control parameter in the host. To aid in mapping and scaling in relation to the control of standard VST plugins, a special OSC translator plugin was devised. This plugin provides the same signal conditioning and mixing as the example effects in the toolkit. The difference is that it will output its destination parameter value via OSC, using the OSC address *parmN* with N being an integer in range 1-8. Selection of network port is available in the plugin GUI. This OSC message can easily be routed to any destination parameter in the host. For more info on setting up OSC control in Reaper there are details online [7], other hosts will have similar methodologies. Some bandwidth issues were encountered when using these OSC message for host automation (the parameter values in the receiving effect would choke and stop moving after a short time). Apparently, the rate of transmission can overflow the host OSC input buffer. A quick attempt was made to reduce the rate of transmission without adding significant latency or jitter; By quantizing values to be sent to the host (to 0.001 steps) and sending only when the value changes, the host seemed capable of handling the automation signal stably for a long time (> 1 hour).

9. CONCLUSIONS AND FURTHER WORK

We have shown a toolkit for experimentation with signal interaction as a technique for adaptive parameter control of audio effect processors. The system includes methods for audio analysis, as well as routing, mapping and scaling of modulation sources. A number of example effects processors has been implemented as proof of concept and as a starting point for further investigation, and an interface to enable control of generic VST plugins (or any host program automatable parameter) has been shown. Some demonstration sounds of possible sonic interactions have been published at the author's Soundcloud page[8]. Initial experimentation with the toolkit has shown it to be a useful and potentially musically valid technique. First impressions also include the potential to use the toolkit to familiarize oneself with the different analysis concepts, as the mapping of analysis tracks directly to changes in the processing of audio gives a very immediate feedback on the features tracked by the different forms of analysis. Further work needs to be done on practical and musical exploration of the technique, and the mapping between sound features and effects controls can be developed further. The *playability* of *the expressional capabilities* [14] of the system is of special interest in this context, also a subject related to instrumental training for this specific music performance system. For extensions to the mapping we may look at higher level sound descriptors and feature combinations discussed in [23] as well as the relationship between musical gestures and actions with regards to playability, possibly touching on machine learning issues as discussed in [24] and [15]. A series of specially designed effect processing methods can also be envisioned, where the experiences from work with the current toolkit can inform the design of effects that has modulation parameters designed for cross-adaptive control, with an emphasis on the expected relation between musical gestures and the processed sound (of another instrument). The incentive for the toolkit has been to provide some means of experimentation, since little is known about the actual musical usefulness of this kind of interprocessing. The results of experimentation with the toolkit may lead to implementation of more targeted effects processors. In this case, the analysis may be implemented as an integral part of the processor, relieving the need for inter-plugin communication and enabling lower latencies and tighter signal interaction. Higher level sound descriptors could be implemented, and a better interface for live performance may be devised as a means of making the tool accessible to a larger base of users of music software for production and performance.

[7] http://www.reaper.fm/sdk/osc/osc.php

[8] https://soundcloud.com/brandtsegg/sets/interprocessing-demo-sounds

10. ACKNOWLEDGMENTS

I would like to acknowledge the suggestions, corrections and valuable discussions from the research environment around music technology at NTNU. In addition, significant input has come from the Csound community and developers.

11. REFERENCES

[1] A. Moritz, "Introduction to hymnen," `http://home.earthlink.net/~almoritz/hymnenintro.htm`, 2003 (accessed May 19, 2015).

[2] Ø. Brandtsegg and S. Saue, "Experiments with dynamic convolution techniques in live performance," *Linux Audio Conference*, 2013.

[3] L.E. Myhre, A.H. Bardoz, S. Saue, Ø. Brandtsegg, and J. Tro, "Cross convolution of live audio signals for musical applications," in *Proceedings of the 10th International Symposium on Computer Music Multidisciplinary Research*, 2013, pp. 878–885.

[4] V. Verfaille, U. Zolzer, and D. Arfib, "Adaptive digital audio effects (a-DAFx): a new class of sound transformations," *Audio, Speech and Language Processing, IEEE Transactions on [see also Speech and Audio Processing, IEEE Transactions on]*, vol. 14, no. 5, pp. 1817–1831, 2006.

[5] Vincent Verfaille and Daniel Arfib, "ADAFx: Adaptive digital audio effects," in *Proceedings of the COST-G6 Workshop on Digital Audio Effects (DAFx-01)*, Limerick, Ireland, 2001, pp. 10–4.

[6] Victor Lazzarini, Joseph Timoney, and Thomas Lysaght, "The generation of natural-synthetic spectra by means of adaptive frequency modulation.," *Computer Music Journal*, vol. 32, no. 2, pp. 9–22, 2008.

[7] Victor Lazzarini, Joseph Timoney, Jussi Pekonen, and Vesa Välimäki, "Adaptive phase distortion synthesis," *DAFx 09 proceedings of the 12th International Conference on Digital Audio Effects, Politecnico di Milano, Como Campus, Sept. 1-4, Como, Italy*, pp. 1–8, 2009.

[8] E. Perez-Gonzalez and J. D. Reiss, "Automatic mixing," in *Digital Audio Effects, Second Edition*, U. Zoelzer, Ed., book section 13, pp. 523–550. John Wiley & Sons, Ltd, Chichester, UK, 2011.

[9] J. D. Reiss, "Intelligent systems for mixing multichannel audio," in *17th International Conference on Digital Signal Processing (DSP2011)*, 2011, pp. 1–6.

[10] Enrique Perez-Gonzalez and Joshua Reiss, "Improved control for selective minimization of masking using interchannel dependancy effects," in *Proc. of the 11th Int. Conference on Digital Audio Effects (DAFx-08)*, Sept. 2008.

[11] Brecht De Man and Joshua D. Reiss, "Adaptive control of amplitude distortion effects," in *53rd Conference of the Audio Engineering Society*, January 2014.

[12] Stuart Mansbridge, Saorise Finn, and Joshua D. Reiss, "An Autonomous System for Multitrack Stereo Pan Positioning," in *AES 133rd Convention*, Oct. 2012.

[13] Cornelius Poepel and Roger B. Dannenberg, "Audio signal driven sound synthesis," in *ICMC 2005 International Computer Music Conference*, Barcelona, Spain, September 2005, ICMC, pp. 391–394.

[14] T. Todoroff, "Control of digital audio effects," in *Dafx: Digital Audio Effects*, U. Zoelzer, Ed. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[15] S. Fasciani, *Voice-controlled Interface for Digital Musical Instruments*, Ph.D. thesis, National University of Singapore, 2014.

[16] M. Stabile, "Adapt: A networkable plug-in host for dynamic creation of real-time adaptive digital audio effects," M.S. thesis, University of California, Santa Barbara, 2010.

[17] Andy Hunt, Marcelo Wanderley, and Ross Kirk, "Towards a model for instrumental mapping in expert musical interaction," *Proceedings of the 2000 International Computer Music Conference*, pp. 209–212, 2000.

[18] G. Peeters, B. L. Giordano, P. Susini, N. Misdariis, and S. McAdams, "The timbre toolbox: Extracting acoustic descriptors from musical signals," *Journal of The Acoustical Society Of America*, vol. 130, pp. 2902–2916, 2011.

[19] B Yegnanarayana and S Gangashetty, "Epoch-based analysis of speech signals," *Sadhana*, vol. 36, no. 5, pp. 651–697, 2011.

[20] M. Puckette, T. Apel, and D. Zicarelli, "Real-time audio analysis tools for pd and msp," in *Proceedings of the International Computer Music Conference*, 1998, pp. 109–112.

[21] U. Zolzer, S.V. Sankarababu, and S. Moller, "Pll-based pitch detection and tracking for audio signals," in *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2012 Eighth International Conference on*, July 2012, pp. 428–431.

[22] M. Ross, H. Shaffer, A. Cohen, R. Freudberg, and H. Manley, "Average magnitude difference function pitch extractor," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 22, no. 5, pp. 353–362, Oct 1974.

[23] Vincent Verfaille, Marcelo M. Wanderley, and Philippe Depalle, "Mapping strategies for gestural and adaptive control of digital audio effects," *Journal of New Music Research*, vol. 35, no. 1, pp. 71–93, 2006.

[24] E. Metois, *Musical Sound Information: MusicalGestures and Embedding Synthesis*, Ph.D. thesis, Massachusetts Institute of Technology, 1997.

IMPROVING THE ROBUSTNESS OF THE ITERATIVE SOLVER IN STATE-SPACE MODELLING OF GUITAR DISTORTION CIRCUITRY

Ben Holmes and Maarten van Walstijn

Sonic Arts Research Center
School of Electronics, Electrical Engineering and Computer Science,
Queen's University Belfast
Belfast, Northern Ireland, U.K.
{bholmes02,m.vanwalstijn}@qub.ac.uk

ABSTRACT

Iterative solvers are required for the discrete-time simulation of nonlinear behaviour in analogue distortion circuits. Unfortunately, these methods are often computationally too expensive for real-time simulation. Two methods are presented which attempt to reduce the expense of iterative solvers. This is achieved by applying information that is derived from the specific form of the nonlinearity. The approach is first explained through the modelling of an asymmetrical diode clipper, and further exemplified by application to the Dallas Rangemaster Treble Booster guitar pedal, which provides an initial perspective of the performance on systems with multiple nonlinearities.

1. INTRODUCTION

In physical modelling of analogue distortion circuitry, the greatest challenges are typically posed by the modelling of nonlinear components, such as diodes, triodes, and bipolar junction transistors (BJTs). In recent literature, this topic has attracted specific attention in relation to real-time implementation, which necessitates a sharp trade off between accuracy and efficiency, with a further possible requirement of parametric control, i.e. allowing on-line updates of the system parameters. Various modelling paradigms have emerged to meet these demands, including Wave Digital Filters (WDF) [1, 2], state-space models (including the K-method and variants thereof) [3, 4, 5, 6], and Port-Hamiltonian Systems [7]. Each of these appoaches can make use of a precomputed lookup table (LUT) that stores the nonlinear behaviour, thus avoiding the need to solve a multidimensional system of implicit nonlinear equations on-line (see, e.g. [8]). The downside of the use of LUTs is that it complicates parametric control, in particular when dealing with multivariate nonlinearities. One way to address this is by decomposing the nonlinearity, which significantly reduces the computational complexity, although accurate simulation of complex circuits will require very large table sizes [9]. For univariate cases (i.e. circuits with a single nonlinearity or with multiple, separable nonlinearities), WDFs are exceptionally suited to real-time implementation, offering both efficiency and modularity [10]. However, these properties do not readily extend to modelling systems with multiple, non-separable nonlinearities, in which case device-specific simplifying assumptions have to be made to avoid multivariate root-finding [11, 12].

A more general approach is offered by state-space methods, but initial formulations were not particularly suited to parametric control due to the need for computationally expensive matrix inversions. An elegant solution was offered in [13], proposing a Nodal DK formulation that employs strategic matrix decomposition to reduce the inversion costs associated with parameter updates, without sacrificing the beneficial feature of automated derivation of the state-space equations. Nevertheless, the approach still requires numerically solving a system of nonlinear equations, which is commonly achieved with Newton's method or variants thereof. Such iterative methods entail the risk of not converging to a suitably accurate solution within a limited number of iterations, a problem that is most prevalent when driving the circuit with signals of high amplitude and/or frequency, and that is further exacerbated when increasing the number of non-separable system nonlinearities.

In this paper we present two new adaptations of Newton's method which exploit the form of the nonlinear function of the selected system to help limit the computational cost of finding the root. A key feature is their amenability to parameter updates through the use of analytic expressions. The performance of these methods with the Nodal DK-method is evaluated through comparison with existing root-finding methods in terms of robustness and computational efficiency.

2. NODAL DK-METHOD

The Nodal DK-method was first developed in [3] to algorithmically generate state-space models of nonlinear audio circuits. The method applies Modified Nodal Analysis (MNA) to build a computable system from nodal equations, and uses the trapezoidal rule to discretise reactive components. The specific method used in this paper to model circuits is described in [13]. The state space model is represented by

$$\boldsymbol{x}[n] = \boldsymbol{A}\boldsymbol{x}[n-1] + \boldsymbol{B}\boldsymbol{u}[n] + \boldsymbol{C}\boldsymbol{f}(\boldsymbol{v}_{\mathrm{n}}[n]) \tag{1}$$

$$\boldsymbol{y}[n] = \boldsymbol{D}\boldsymbol{x}[n-1] + \boldsymbol{E}\boldsymbol{u}[n] + \boldsymbol{F}\boldsymbol{f}(\boldsymbol{v}_{\mathrm{n}}[n]) \tag{2}$$

$$\boldsymbol{v}_{\mathrm{n}}[n] = \boldsymbol{G}\boldsymbol{x}[n-1] + \boldsymbol{H}\boldsymbol{u}[n] + \boldsymbol{K}\boldsymbol{f}(\boldsymbol{v}_{\mathrm{n}}[n]) \tag{3}$$

where $\boldsymbol{x}$ is the state variable, $\boldsymbol{u}$ is the model input, $\boldsymbol{y}$ is the model output, and $\boldsymbol{f}(\boldsymbol{v}_{\mathrm{n}})$ represents the terminal currents of the nonlinear elements relative to the nonlinear voltage $\boldsymbol{v}_{\mathrm{n}}$. Coefficient matrices $\boldsymbol{A} - \boldsymbol{H}$ and $\boldsymbol{K}$ control the linear combinations of each variable used to update the state and output. The model is updated by first finding the nonlinear voltage state, which is then used to update the state variable. To find the nonlinear voltage state, $\boldsymbol{v}_{\mathrm{n}}$, (3) must be solved numerically. This amounts to finding the root of the function

$$\boldsymbol{g}(\boldsymbol{v}_{\mathrm{n}}[n]) = \boldsymbol{p}[n] + \boldsymbol{K}\boldsymbol{f}(\boldsymbol{v}_{\mathrm{n}}[n]) - \boldsymbol{v}_{\mathrm{n}}[n] \tag{4}$$

where $\boldsymbol{p}[n] = \boldsymbol{G}\boldsymbol{x}[n-1] + \boldsymbol{H}\boldsymbol{u}[n]$.

3. NUMERICAL ROOT FINDING METHODS

Initially, a wide selection of root-finding methods were trialled to assess which met conditions that suggest real-time capability. The methods must both: be extendible to multivariate cases, and converge within a specified number of iterations.

In the domain of audio circuit modelling, nonlinear elements are based upon physical properties. Functions based upon these properties typically have unique roots, and sufficiently well-conditioned gradients that many root finding methods utilise. We therefore define the term non-convergent as a measure of robustness, where for cases on specific computational systems either: the current value exceeds values representable by normal floating point arithmetic; or the number of iterations exceeds a limit that can be completed in an allocated amount of time.

3.1. First Order Methods

Newton's method uses a linear approximation to the nonlinear function to successively find better approximations to the root of the function. Several methods use this technique as a basis, of which four are discussed. A more comprehensive understanding of these methods can be obtained from the literature [14].

3.1.1. Newton's Method

The iterative method employed by Newton's method is typically expressed as

$$\boldsymbol{v}_{i+1} = \boldsymbol{v}_i - \boldsymbol{J}^{-1}(\boldsymbol{v}_i)\boldsymbol{g}(\boldsymbol{v}_i) \tag{5}$$

where $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$ are the current and next iterate, $\boldsymbol{g}(\boldsymbol{v}_i)$ is function at the current iterate known as the residual, and $\boldsymbol{J}(\boldsymbol{v}_i)$ is the Jacobian matrix.

To detect when a root has been found, the inequality $\mid \boldsymbol{v}_{i+1} - \boldsymbol{v}_i \mid < \mathbf{TOL}$ must be satisfied, which specifies the error is less than a certain tolerance, represented by $\mathbf{TOL}$. The tolerance is selected by the user, and often informed by the required accuracy of the result, and the system's numerical precision.

3.1.2. Damped Newton's Method

By applying damping to Newton's method, iterations that increase the residual can be corrected. This is accomplished by reducing the step size until the residual at the new iterate is less than the residual at the previous iterate. This is applied to (5) as a scalar multiplier of the step, so that

$$\boldsymbol{v}_{i+1} = \boldsymbol{v}_i - 2^{-m}\boldsymbol{J}^{-1}(\boldsymbol{v}_i)\boldsymbol{g}(\boldsymbol{v}_i) \tag{6}$$

where the value of m is the smallest integer that satisfies the inequality [14]

$$||\boldsymbol{g}\left(\boldsymbol{v}_i - 2^{-m}\boldsymbol{J}^{-1}(\boldsymbol{v}_i)\boldsymbol{g}(\boldsymbol{v}_i)\right)|| \leq ||\boldsymbol{g}(\boldsymbol{v}_i)||. \tag{7}$$

The value of m is found by iteratively incrementing the value until the condition is satisfied. Damped Newton's method has been shown to be successful for nonlinearities that are more likely to demonstrate non-convergence, for example BJTs [15, 16].

3.1.3. Chord Method

The most expensive operation in Newton's method is the calculation of the inverse Jacobian. To lessen the computational cost of the method, it is possible to only calculate the Jacobian at the initial iterate, and use this at each successive iterate. A disadvantage of this method is that if the Jacobian at the initial iterate causes a step that overshoots the root, the overshoot is more likely to happen successively, causing divergence from the root.

3.1.4. Secant Method

The secant method uses a difference method to calculate the Jacobian. In univariate cases, it has been successfully applied in the simulation of a triode [12]. For multivariate models the method extends to Broyden's method. To numerically approximate the Jacobian, Broyden's method requires an initial Jacobian which it then updates using a difference method. Upon initial testing, Broyden's method was less robust than the Chord method. For this reason it was not included in the final comparison.

3.2. Quadratic Methods

Halley's method extends Newton's method using both Jacobian and Hessian matrices to form a quadratic approximation to the nonlinear function. Supporting literature demonstrates that Halley's method has faster convergence than Newton's method [17]. It was found that Halley's method was less robust than Newton's method, and it was for this reason Halley's method was not included in the final comparison.

Brent's method implements a difference approach to form a quadratic function [18]. Additional bracketing and conditions are applied to improve robustness. For univariate cases, this method proved to be the most robust method, and exhibited good convergence. However, the method has not been extended to multiple dimensions so was not included within the final comparison.

3.3. A Semi-Analytic Form

The Lambert W function provides analytical solutions for equations of the form

$$\mathrm{W}(z)\mathrm{e}^{\mathrm{W}(z)} = z. \tag{8}$$

This has been applied successfully to diodes with series resistance both in a general case [19] and using Wave Digital Filters [20].

The Lambert W function is also applicable to state-space models. It does not extend to multivariate cases and therefore was not included in the final comparison, but is functional for univariate cases. This can be shown using a generic circuit featuring a single diode modelled using the Shockley equation from (12). The nonlinear function from (4) must then be re-arranged into the form of (8) to find $\mathrm{W}(z)$ and z. For the diode case this gives

$$\mathrm{W}(z) = -\frac{K(f(v_\mathrm{n})) - I_\mathrm{S})}{NV_\mathrm{T}}, \quad z = -\frac{KI_\mathrm{S}}{NV_\mathrm{T}}\mathrm{e}^{\frac{p+KI_\mathrm{S}}{NV_\mathrm{T}}}. \tag{9}$$

Where K is the coefficient from (4) in scalar form. Solving for $f(v_\mathrm{n})$ then yields

$$f(v_\mathrm{n}) = -\frac{NV_\mathrm{T}}{K}\mathrm{W}(z) - I_\mathrm{S}. \tag{10}$$

An accurate model of anti-parallel diodes can be formed by adapting (10), taking the absolute value of p and multiplying $f(v_\mathrm{n})$ by $\mathrm{sgn}(p)$ [20] to incorporate the polarity. This relies on the assumption in (18), where for this case $a \approx b$. For asymmetrical diodes, a conditional statement must be applied, incorporating both the polarity of $f(v_\mathrm{n})$ and the different coefficients in the Shockley equation.

4. EXPLOITING THE FORM OF THE NONLINEAR FUNCTION

With respect to the variable $\boldsymbol{v}_\mathbf{n}$, the function of $\boldsymbol{g}(\boldsymbol{v}_\mathrm{n})$ in (4) can be decomposed into a constant term, a linear term, and a nonlinear

term:

$$\boldsymbol{g}(\boldsymbol{v}_\mathrm{n}[n]) = \boldsymbol{p}[n] + \underbrace{\boldsymbol{K}\boldsymbol{f}(\boldsymbol{v}_\mathrm{n}[n])}_{\boldsymbol{g}_\mathrm{n}(\boldsymbol{v}_\mathrm{n}[n])} - \underbrace{\boldsymbol{v}_\mathrm{n}[n]}_{\boldsymbol{g}_\mathrm{l}(\boldsymbol{v}_\mathrm{n}[n])} \tag{11}$$

where $\boldsymbol{g}_\mathrm{n}(\boldsymbol{v}_\mathrm{n}[n])$, and $\boldsymbol{g}_\mathrm{l}(\boldsymbol{v}_\mathrm{n}[n])$ indicate nonlinear and linear function components. In this section we propose two new Newton-based methods that employ system knowledge derived from this decomposition: the **Capped Step** and **New Iterate** methods. These methods are explained using two case studies, covering both univariate and multivariate nonlinearities. Sound examples and models can be found at:
`http://bholmesqub.github.io/DAFx15/`.

4.1. Univariate Case: Asymmetrical Diode Clipper

A univariate nonlinearity is exemplified here by a diode clipper, which has been covered extensively in the literature [21, 6]. The circuit uses the exponential nature of the voltage-current relation of the diode to limit the voltage output. The specific diode clipper used here can be seen in Figure 1, and features anti-parallel diodes in a 2:1 ratio.

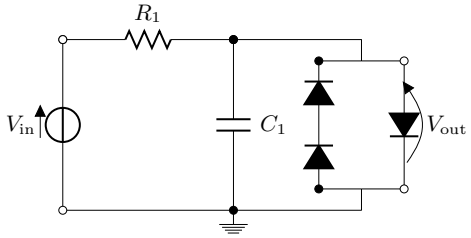


Figure 1: *Schematic of the modelled asymmetrical diode clipper.*

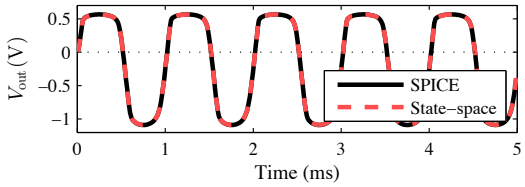


Figure 2: *A* 2 V, 1 kHz *sine wave processed by both SPICE and state-space diode clippers.* $f_s = 176.4\,\mathrm{kHz}$

The Shockley model is used as the component model for the diodes, representing the current through a diode as

$$I_\mathrm{D} = I_\mathrm{S}\left(\mathrm{e}^{\frac{V_\mathrm{D}}{NV_\mathrm{T}}} - 1\right) \tag{12}$$

where I_S is the reverse saturation current, V_D is the voltage across the diode, V_T is the thermal voltage, and N is the ideality factor. Noting in this case $v_\mathrm{n} = V_\mathrm{D}$, the asymmetric combination forms the nonlinear term

$$f(v_\mathrm{n}) = I_\mathrm{S}\left(\mathrm{e}^{\frac{v_\mathrm{n}}{NV_\mathrm{T}}} - 1\right) - I_\mathrm{S}\left(\mathrm{e}^{\frac{-v_\mathrm{n}}{2NV_\mathrm{T}}} - 1\right) \tag{13}$$

where the factor of $1/2$ in the second exponent represents the two diodes, as each diode carries half of the voltage drop across the terminals. This relies on the assumption that the diodes are identical.

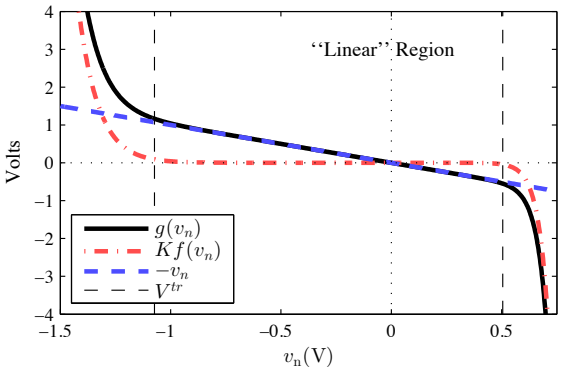


Figure 3: *Decomposed regions of the diode clipper nonlinearity where* $p[n] = 0$, $f_s = 176.4\,\mathrm{kHz}$. $V_+^\mathrm{tr} = 0.5052\,\mathrm{V}$, $V_-^\mathrm{tr} = -1.0731\,\mathrm{V}$.

For this specific model, the following component values were used: $R_1 = 2200\,\Omega$, $C_1 = 0.01\,\mu\mathrm{F}$, $I_\mathrm{S} = 2.52\,\mathrm{nA}$, $N = 1.752$, $V_\mathrm{T} = 25.8\,\mathrm{mV}$. The diode values are taken from LTspice IV [22], and refer to a 1N4148 signal diode. The state-space model has been validated using SPICE, which is illustrated in Figure 2.

4.1.1. Capping the Newton Step

A problematic case for Newton-based methods arises when the gradient at the initial iterate causes a Newton step that overshoots the root of the function. The exponential nature of the examined nonlinear terms prevents this for large values of $p[n]$. When $p[n]$ is small, the nonlinear term becomes significantly smaller than the linear term, which can cause an overshoot if the root is not in close proximity. In extreme cases, the residual exceeds values representable by normal floating point arithmetic. As seen in Section 3.1.2, applying damping to Newton's method aids this with the trade-off of sub-iterations.

An alternative approach is to set a maximum step size, for example with a simple comparative function:

$$\overline{\Delta v_\mathrm{n}} = \begin{cases} \mathrm{sgn}(\Delta v_\mathrm{n})V^\mathrm{lim}, & |\Delta v_\mathrm{n}| > V^\mathrm{lim} \\ \Delta v_\mathrm{n}, & |\Delta v_\mathrm{n}| \leq V^\mathrm{lim} \end{cases} \tag{14}$$

where $\overline{\Delta v_\mathrm{n}}$ and Δv_n represent the capped and unaltered step size, V^lim is the limit placed upon it, and the signum function adjusts the polarity. For this to be successful, a limit must be specified that is large enough to prevent drastically increasing the number of iterations required. A suitable value is defined by finding the transitional voltages beyond which the nonlinear term is dominant, as illustrated in Figure 3 (which also compares the decomposition of the nonlinear function from (11)). The distance of this voltage from the origin is applied as the limit, such that $V^\mathrm{lim} = |V^\mathrm{tr}|$, where V^tr is the transitional voltage.

4.1.2. Defining System-Specific Transitional Voltages

To find the transitional voltages of the nonlinear function, the gradient information of the nonlinear and linear terms are compared. For the univariate case, this amounts to finding the two values of v_n for which $dg_\mathrm{l}/dv_\mathrm{n} = dg_\mathrm{n}/dv_\mathrm{n}$. Applying this using (18) to

separate terms yields

$$-1 = \frac{KI_\mathrm{S}}{NV_\mathrm{T}} \mathrm{e}^{\frac{v_\mathrm{n}}{NV_\mathrm{T}}}, \quad -1 = \frac{KI_\mathrm{S}}{2NV_\mathrm{T}} \mathrm{e}^{\frac{-v_\mathrm{n}}{2NV_\mathrm{T}}}. \tag{15}$$

Solving these equations for v_n finds the transitional voltages, expressed as:

$$V_+^\mathrm{tr} = NV_\mathrm{T} \log\left(-\frac{NV_\mathrm{T}}{KI_\mathrm{S}}\right), V_-^\mathrm{tr} = -2NV_\mathrm{T} \log\left(-\frac{2NV_\mathrm{T}}{KI_\mathrm{S}}\right). \tag{16}$$

4.1.3. Setting a Strategic Initial Iterate

Typically, the solution from the previous sample is used as an initial iterate to find the solution at the current sample. Fast convergence then relies on the assumption of small inter-sample differences, but this breaks down with inputs of high-frequency and/or amplitude, depending also on the sampling frequency. An alternative to this is to use an approximation to the nonlinear function, which will place the initial iterate at a position which prevents overshoot of the root (as discussed in Section 4.1.1) and is independent of the past sample. This forms the basis of the New Iterate method, which attempts to reduce the dependency of convergence on the input and sampling frequency.

The proposed approximation to the univariate version of (11) is formed by removing the linear term, which is accurate when v_n is large. If the nonlinear term $f(v_\mathrm{n}[n])$ is an invertible function, this allows for an analytical solution for $v_\mathrm{n}[n]$, where the general univariate form is

$$v_\mathrm{n}^\mathrm{NI}[n] = f^{-1}\left(-\frac{p[n]}{K}\right). \tag{17}$$

To apply this to the asymmetrical diode clipper, the nonlinear function can be separated into positive and negative terms, using the assumption

$$\left|\mathrm{e}^{a|v_\mathrm{n}|} - 1\right| \gg \left|\mathrm{e}^{-b|v_\mathrm{n}|} - 1\right| \tag{18}$$

where a and b are positive constants. The two separate functions can then be inverted to solve for the new initial iterate

$$v_\mathrm{n}^\mathrm{NI}[n] = \begin{cases} NV_\mathrm{T} \log\left(1 - \frac{p[n]}{KI_\mathrm{S}}\right), & p[n] \geq 0 \\ -2NV_\mathrm{T} \log\left(1 + \frac{p[n]}{KI_\mathrm{S}}\right), & p[n] < 0 \end{cases} \tag{19}$$

where $p[n]$ is used to determine the polarity.

4.2. Multivariate Case: Dallas Rangemaster

To exemplify systems with more than one nonlinearity, the Dallas Rangemaster is modelled. The Rangemaster is an early "treble booster" pedal which increases the amplitude of the guitar signal to drive the amplifier into further saturation, particularly at higher frequencies. Figure 4 illustrates the complete schematic of the model, with R_4 modelling the load of the circuit. The pedal features one parameter which changes the gain, but for the purpose of comparison it was set to maximum.

The nonlinear behaviour is caused by the PNP BJT, which is modelled using the Ebers-Moll injection model. The Ebers-Moll model represents the current through each terminal (Base, Collector, and Emitter) as a combination of the voltages across its terminals. For a complete model, only two of these equations are required as the third can be found using superposition [21]. The current-voltage relationships can thus be represented by

Table 1: *Component values of the Rangemaster circuit.*

R_1	470 kΩ	R_4	1 MΩ	C_2	4.7 pF
R_2	68 kΩ	VR_1	10 kΩ	C_3	47 µF
R_3	3.9 kΩ	C_1	47 µF	C_4	10 pF

$$I_\mathrm{B} = \frac{I_\mathrm{S}}{\beta_\mathrm{F}}\left(\mathrm{e}^{\frac{V_\mathrm{EB}}{V_\mathrm{T}}} - 1\right) + \frac{I_\mathrm{S}}{\beta_\mathrm{R}}\left(\mathrm{e}^{\frac{V_\mathrm{EB}-V_\mathrm{EC}}{V_\mathrm{T}}} - 1\right) \tag{20}$$

$$I_\mathrm{C} = I_\mathrm{S}\left(\mathrm{e}^{\frac{V_\mathrm{EB}}{V_\mathrm{T}}} - 1\right) - I_\mathrm{S}\frac{\beta_\mathrm{R}+1}{\beta_\mathrm{R}}\left(\mathrm{e}^{\frac{V_\mathrm{EB}-V_\mathrm{EC}}{V_\mathrm{T}}} - 1\right) \tag{21}$$

where β_F and β_R are the forward and reverse common-emitter current gain. The original Rangemaster used a germanium BJT, but for the model generic parameters were used: $I_\mathrm{S} = 10\,\mathrm{fA}$, $\beta_\mathrm{F} = 200$, $\beta_\mathrm{R} = 2$ and V_T remains the same as for the diode clipper case. The full nonlinear function is expressed by

$$\boldsymbol{g}(\boldsymbol{v}_\mathrm{n}) = \boldsymbol{p} + \boldsymbol{K}\begin{bmatrix} I_\mathrm{B} \\ I_\mathrm{C} \end{bmatrix} - \begin{bmatrix} V_\mathrm{EB} \\ V_\mathrm{EC} \end{bmatrix}. \tag{22}$$

The component values are shown in Table 1. The state-space model was validated with SPICE, which is illustrated in Figure 5. To produce this result, both simulations were initialised with steady-state solutions.

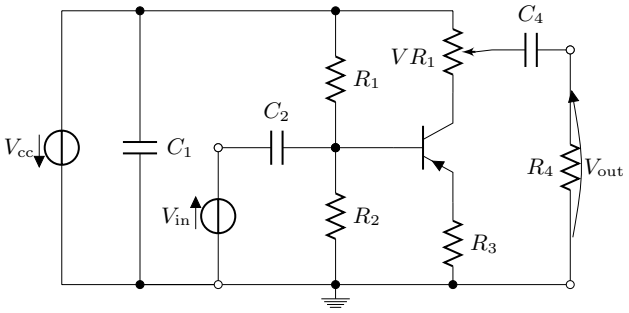


Figure 4: *Schematic of the modelled Dallas Rangemaster Treble Booster.*

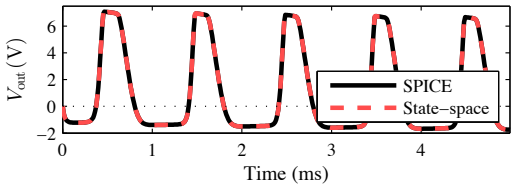


Figure 5: *A* 200 mV, 1 kHz *sine wave processed by both SPICE and state-space Rangemasters.* $V_\mathrm{cc} = 9\,\mathrm{V}$, $f_s = 176.4\,\mathrm{kHz}$.

4.2.1. Setting a Multivariate Initial Iterate

Finding an approximation of a multivariate function follows the same process as applied to the univariate case. To find the inverted

form of the Ebers-Moll functions, they must be decomposed. To accomplish this, the Ebers-Moll functions can be expressed as the product of a square matrix and a vector:

$$\begin{bmatrix} I_{\mathrm{B}} \\ I_{\mathrm{C}} \end{bmatrix} = \boldsymbol{L} \begin{bmatrix} \mathrm{e}^{\frac{V_{\mathrm{EB}}}{V_{\mathrm{T}}}} - 1 \\ \mathrm{e}^{\frac{V_{\mathrm{EB}} - V_{\mathrm{EC}}}{V_{\mathrm{T}}}} - 1 \end{bmatrix}, \quad \boldsymbol{L} = I_{\mathrm{S}} \begin{bmatrix} \frac{1}{\beta_{\mathrm{F}}} & \frac{1}{\beta_{\mathrm{R}}} \\ 1 & -\frac{\beta_{\mathrm{R}}+1}{\beta_{\mathrm{R}}} \end{bmatrix}. \tag{23}$$

The simplified nonlinear equation of the nodal DK-method can then be solved for the vector containing the exponents:

$$-\boldsymbol{Q}^{-1}\boldsymbol{p} = \begin{bmatrix} \mathrm{e}^{\frac{V_{\mathrm{EB}}}{V_{\mathrm{T}}}} - 1 \\ \mathrm{e}^{\frac{V_{\mathrm{EB}} - V_{\mathrm{EC}}}{V_{\mathrm{T}}}} - 1 \end{bmatrix} \tag{24}$$

where $\boldsymbol{Q} = \boldsymbol{K}\boldsymbol{L}$. Values for V_{EB} and V_{EC} are then solved for by separately inspecting the terms in (24), where V_{EC} is found from the lower term after first determining V_{EB} from the upper term:

$$V_{\mathrm{EB}}^{\mathrm{NI}} = V_{\mathrm{T}}\log\big(1 - \hat{p}_1\big), \quad V_{\mathrm{EC}}^{\mathrm{NI}} = V_{\mathrm{EB}}^{\mathrm{NI}} - V_{\mathrm{T}}\log\big(1 - \hat{p}_2\big) \tag{25}$$

where $\hat{\boldsymbol{p}} = \boldsymbol{Q}^{-1}\boldsymbol{p}$.

4.2.2. Defining System-Specific Transitional Voltages

Each term of the Ebers-Moll functions depends upon V_{EB}, which complicates the process of finding independent transitional voltages. By creating a new voltage vector $\hat{\boldsymbol{v}} = \begin{bmatrix} V_{\mathrm{EB}} & V_{\mathrm{CB}} \end{bmatrix}^T$ using the substitution $V_{\mathrm{CB}} = V_{\mathrm{EB}} - V_{\mathrm{EC}}$, two voltages are provided of which to find the transitions. The equation $\partial \boldsymbol{g}_{\mathrm{n}}/\partial \hat{\boldsymbol{v}} = \partial \boldsymbol{g}_{\mathrm{l}}/\partial \hat{\boldsymbol{v}}$ is then used to find each transition. Two transitions are found for V_{EB},

$$V_{\mathrm{EB}}^{\mathrm{tr}} = V_{\mathrm{T}}\log\left(-\frac{V_{\mathrm{T}}}{Q_{11}}\right) \text{ and } V_{\mathrm{EB}}^{\mathrm{tr}} = V_{\mathrm{T}}\log\left(-\frac{V_{\mathrm{T}}}{Q_{21}}\right), \tag{26}$$

and one transition is found for V_{CB},

$$V_{\mathrm{CB}}^{\mathrm{tr}} = V_{\mathrm{T}}\log\left(\frac{V_{\mathrm{T}}}{Q_{22}}\right). \tag{27}$$

As the solution for $V_{\mathrm{CB}}^{\mathrm{tr}}$ is found using the partial derivative w.r.t. V_{CB}, V_{EB} is ignored allowing the limit relative to V_{EC} to be defined as $V_{\mathrm{EC}}^{\mathrm{lim}} = |V_{\mathrm{EC}}^{\mathrm{tr}}| = |-V_{\mathrm{CB}}^{\mathrm{tr}}|$.

4.2.3. Capping the Multivariate Newton Step

To apply capping to a multivariate step, the same function from (14) can be applied individually to each term. The lower of the two values from (26) is applied as the limit for V_{EB}. In the case of the modelled BJT, these values are in close proximity so that the difference in performance is negligible.

5. COMPUTATIONAL COST

To assess the efficiency of the root-finding methods, they were compared in terms of the number of operations required to converge. The Lightspeed Matlab toolbox [23] was used to provide costs of floating point operations (FLOPs). Integer operation costs were set equal to the floating point equivalent. Branch operations were given the same cost as logical and relational operators. Control dependencies were ignored for simplicity as they are difficult to represent using an operation cost. These choices inform two specifications about the theoretical hardware used for the simulation: the integer and floating point hardware performs equally, and there is no instruction level parallelism (i.e. operation pipelining). The cost of each operation used within the algorithms is stated in Table 2.

Table 2: *Cost of individual operations.*

Operation	Cost
$+,\ -,\ \times$	1
logical, relational, branch	2
abs()	4
sgn()	5
$\div$	8
exp()	40
$\|\|\boldsymbol{x}\|\|_2$	$2M + 7$
Solve using LU	$M^3 + \frac{1}{2}M^2 + \frac{29}{2}M - 8$

5.1. Method Costs

Using the values and expressions from Table 2, the cost of each method was determined. Each cost is determined based upon the number of dimensions it is solving for, M, and the number of iterations it performs, i. Additionally, the Damped Newton method requires sub-iterations, denoted by i_{s}. The costs of calls to the function and Jacobian are represented by C_{F} and C_{J} respectively. C_{lim} and C_{iter} represent the initial cost of calculating the transitional voltages and the approximate initial iterate. These values are found at each time step, assuming each method is applicable to audio rate parametric control.

The cost of each method is denoted using subscript: C_{N} for Newton's method; C_{D} for Damped Newton's method; C_{C} for the Chord method; C_{CS} for Newton's method with the capped step applied; and C_{NI} for Newton's method with the new initial iterate.

$$\begin{aligned} C_{\mathrm{N}} &= M^3 + \frac{1}{2}M^2 + \frac{29}{2}M + C_{\mathrm{J}} + C_{\mathrm{F}} - 8 \\ &\quad + i\left(M^3 + \frac{1}{2}M^2 + \frac{35}{2}M + C_{\mathrm{J}} + C_{\mathrm{F}} + 8\right) \end{aligned} \tag{28}$$

$$\begin{aligned} C_{\mathrm{D}} &= M^3 + \frac{1}{2}M^2 + \frac{29}{2}M + C_{\mathrm{J}} + C_{\mathrm{F}} - 8 \\ &\quad + i\left(M^3 + \frac{1}{2}M^2 + \frac{43}{2}M + C_{\mathrm{J}} + C_{\mathrm{F}} + 12\right) \\ &\quad + i_{\mathrm{s}}\left(6M + C_{\mathrm{F}} + 6\right) \end{aligned} \tag{29}$$

$$\begin{aligned} C_{\mathrm{C}} &= M^3 + \frac{1}{2}M^2 + \frac{29}{2}M + C_{\mathrm{J}} + C_{\mathrm{F}} - 8 \\ &\quad + i\left(M^3 + \frac{1}{2}M^2 + \frac{35}{2}M + C_{\mathrm{F}} + 8\right) \end{aligned} \tag{30}$$

$$C_{\mathrm{CS}} = C_{\mathrm{N}} + 21M + 21iM + C_{\mathrm{lim}} \tag{31}$$

$$C_{\mathrm{NI}} = C_{\mathrm{N}} + C_{\mathrm{iter}} \tag{32}$$

Table 3 contains the cost of constant values for both the diode clipper and the Rangemaster models. Using this information, numerical values were obtained for the cost of an iteration and the initial computation for each algorithm. These are displayed in Table 4.

6. RESULTS

Test simulations were designed to compare the performance of each method against two properties: the amount of oversampling

Table 3: *Cost in operations of constant values for both diode clipper and Rangemaster models.*

Variable	Diode Clipper	Rangemaster
C_{lim}	32	124
C_{iter}	37	130
C_{F}	105	234
C_{J}	121	359

Table 4: *Model-specific cost in operations for the computation required for one iteration and the initial computation of each method.*

	Diode Clipper		Rangemaster	
Method	Init.	Iter.	Init.	Iter.
Newton	234	253	624	646
Damped	234	$261 + 117i_{\text{s}}$	624	$658 + 252i_{\text{s}}$
Chord	234	132	624	287
New It.	271	253	754	646
Capped	287	274	790	688

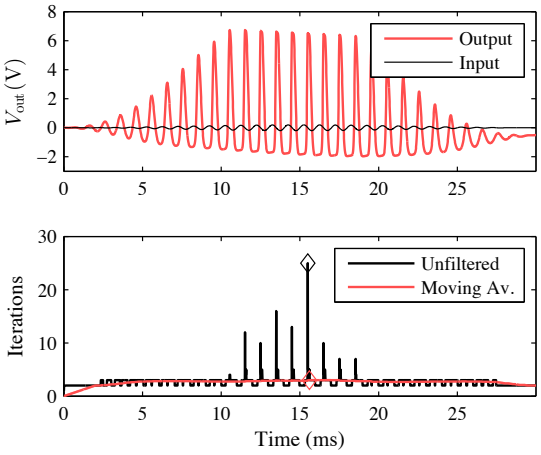


Figure 6: *Input/Output and iteration count of a* 1 kHz, 200 mV *sine wave modulated by a hann window processed by the Rangemaster state-space model using Newton's method,* $f_s = 88.2$ kHz. *Unfiltered and moving average filter results shown, and maximum values marked with* $\diamond$.

applied, and the peak voltage of the input. Oversampling is compared to test how efficient each method is on computational systems with different processing capabilities.

A 30 period, 1 kHz sine wave was used to drive the models. The sine wave was modulated by a Hann window so that the amplitude varied across the range of the nonlinearity. For both circuits, the peak voltage of the input was chosen to match what can be expected from a real circuit. As a diode clipper is typically situated after amplification, the highest peak voltage was set at 9 V, which presumes the system uses a dual-rail ± 9 V power supply. The Rangemaster is designed to be placed at the start of a guitarist's signal chain, so the input reflects a guitar's output. For this reason a representative highest peak voltage was set at 300 mV, although it is noted guitar output voltages can exceed this. The power supply voltage for the Rangemaster model, V_{cc} was set to 9 V.

To ensure a fair comparison, the parameters of the root finding methods were set constant between models and methods. The tolerance was set to 10^{-12}, and the maximum number of iterations was set to 100. Observed inefficiency of Damped Newton's method was corrected by limiting the number of sub-iterations to 3.

Results from the simulations were filtered to emulate the buffering of a real system. Figure 6 shows an example of the unfiltered iterations, and the iterations after being processed by a moving average filter with a window of 2 ms. Table 5 shows results of a set of 16 simulations. Both maximum iteration and operation counts are provided, for which a filtered version and unfiltered version are displayed. Figures 7 and 8 illustrate the performance of the diode clipper and Rangemaster over a range of amplitudes, with no oversampling.

The most notable result from these simulations is that both Chord and Newton's methods exhibit non-convergent behaviour in a variety of tests in which the other three methods are convergent. Of these remaining methods, each has several test cases in which it is the most efficient.

One exclusive feature is the uniform behaviour of the New Iterate method. This is clearly observable from the consistent behaviour relative to sampling frequency, with the maximum variation of 1 iteration (peak) for the case of the Rangemaster with a peak voltage of 300mV. Figure 7 and 8 confirm this behaviour relative to input voltage, although with higher variance.

7. CONCLUSIONS

In this paper two novel root-finding methods were presented using system derived knowledge to improve robustness. The results indicate that for cases of moderate peak voltage and higher sampling frequency, Newton's method is sufficiently robust and relatively efficient. However, for more challenging cases (i.e. cases of high peak voltage and/or low sampling frequency), Newton's method was found to be non-convergent. In principle this can be addressed by using Damped Newton's method, although for several tests it proved to be less efficient than both proposed methods.

The uniform behaviour of the New Iterate method allows the setting of a fixed number of iterations without risking non-convergence, thus alleviating control dependencies. This cannot be achieved by Damped Newton's method, as a branch instruction is required to reduce the step size. The Capped Step method can be configured without control dependencies, but due to its high variance finding a fixed number of iterations is non-trivial. Control dependencies were not considered in this paper as they require focus at a hardware level, but they are known to significantly decrease processor performance [24]. This property suggests that considerable efficiency could be gained using a fixed number of iterations with a method as opposed to a conventional configuration. To assess the consequences of control dependencies, further investigation is required.

A key aspect of the proposed iterative methods is that they rely on the availability of an analytic inverse of either the nonlinear term of the equation to be solved for or its first derivative. This criterion is generally satisfied since the components in distortion circuits are normally modelled with monotone analytical

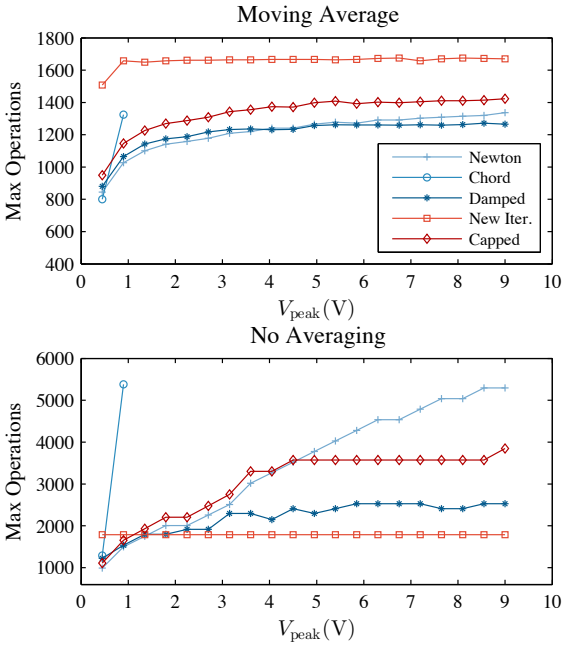


Figure 7: *Maximum operations against input gain for the diode clipper model, no oversampling applied. (Top) The peak averaged iteration cost (Bottom) The peak iteration cost.*

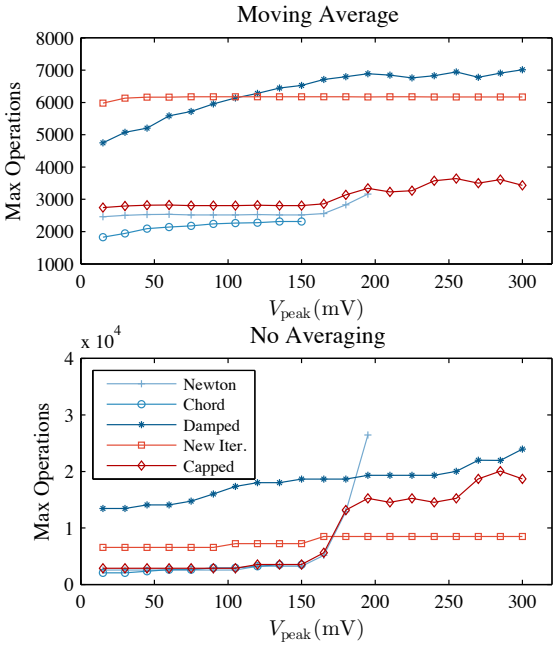


Figure 8: *Maximum operations against input gain for the Rangemaster model, no oversampling applied. (Top) The peak averaged iteration cost (Bottom) The peak iteration cost.*

functions. However one possible limitation is that the analytic inverse function for a specific component model contains significantly more terms than in the cases presented in this study, which may then increase the computational costs accordingly. Hence a further interesting research direction to explore in future research is to test the methodology on more complex component models.

8. REFERENCES

[1] J. Pakarinen and M. Karjalainen, "Enhanced Wave Digital Triode Model for Real-Time Tube Amplifier Emulation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 738–746, May 2010.

[2] G. de Sanctis and A. Sarti, "Virtual Analog Modeling in the Wave-Digital Domain," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 715–727, May 2010.

[3] D. T. Yeh, J. S. Abel, and J. O. Smith, "Automated Physical Modeling of Nonlinear Audio Circuits For Real-Time Audio Effects; Part I: Theoretical Development," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 728–737, May 2010.

[4] I. Cohen and T. Helie, "Simulation of a guitar amplifier stage for several triode models: examination of some relevant phenomena and choice of adapted numerical schemes," in *Audio Engineering Society Convention 127*. 2009, Audio Engineering Society.

[5] J. Macak and J. Schimmel, "Real-Time Guitar Preamp Simulation Using Modified Blockwise Method and Approximations," *EURASIP Journal on Advances in Signal Processing*, 2011.

[6] K. Dempwolf, M. Holters, and U. Zölzer, "Discretization of parametric analog circuits for real-time simulations," in *Proc. of the 13th International Conference on Digital Audio Effects (DAFx'10)*, 2010.

[7] A. Falaize and T. Helie, "Passive Simulation of Electrodynamic Loudspeakers for Guitar Amplifiers: A Port-Hamiltonian Approach," in *Proc. of the International Conference On Noise and Vibration Engineering (ISMA)*, Le Mans, France, 2014.

[8] J. Macak, *Real-Time Digital Simulation of Guitar Amplifiers as Audio Effects*, Ph.D. thesis, Brno University of Technology, 2012.

[9] J. Macak, J. Schimmel, and M. Holters, "Simulation of fender type guitar preamp using approximation and state-space model," in *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-15), York, UK*, 2012.

[10] U. Zölzer, Ed., *DAFX: Digital Audio Effects*, John Wiley & Sons, Chichester, U.K., 2nd edition, 2011.

[11] S. D'Angelo, *Virtual Analog Modeling of Nonlinear Musical Circuits*, Ph.D. thesis, Aalto University, Helsinki, Finland, 2014.

[12] S. D'Angelo, J. Pakarinen, and V. Valimaki, "New Family of Wave-Digital Triode Models," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 21, no. 2, pp. 313–321, 2013.

Table 5: *Results from simulations of both the diode clipper and Rangemaster models, $f_s = 44.1\,\mathrm{kHz}$. Average notates a moving average filter has been applied, Peak notates no filtering. Entries marked "-" indicate the method was non-convergent.*

	$1 \times f_s$				$2 \times f_s$				$4 \times f_s$				$8 \times f_s$			
	Average		Peak		Average		Peak		Average		Peak		Average		Peak	
	Its.	Ops.	Its.	Ops.	Its.	Ops.	Its.	Ops	Its.	Ops.	Its.	Ops.	Its.	Ops.	Its.	Ops.
Diode Clipper, $V_{peak} = 1\,\mathrm{V}$																
Newton	3.2	1038	5	1499	2.8	953	4	1246	2.6	896	3	993	2.3	810	3	993
Damped	3.2	1064	5	1539	2.8	976	4	1278	2.6	917	3	1017	2.3	828	3	1017
Chord	9.1	1434	48	6570	5.7	981	15	2214	4.2	787	8	1290	3.4	679	6	1026
New It.	5.5	1670	6	1789	5.5	1656	6	1789	5.4	1644	6	1789	5.4	1631	6	1789
Capped	3.2	1158	5	1657	2.8	1066	4	1383	2.6	1004	3	1109	2.3	911	3	1109
Diode Clipper, $V_{peak} = 4.5\,\mathrm{V}$																
Newton	4.0	1240	13	3523	3.3	1080	8	2258	3.0	982	5	1499	2.7	927	4	1246
Damped	3.8	1233	7	2295	3.3	1100	6	1917	3.0	1005	5	1539	2.7	949	4	1278
Chord	-	-	-	-	-	-	-	-	6.8	1132	99	13302	4.7	854	17	2478
New It.	5.5	1667	6	1789	5.6	1685	6	1789	5.7	1718	6	1789	5.8	1742	6	1789
Capped	4.0	1371	12	3575	3.3	1203	8	2479	3.0	1097	5	1657	2.7	1038	4	1383
Rangemaster, $V_{peak} = 100\,\mathrm{mV}$																
Newton	2.9	2518	3	2562	2.8	2442	3	2562	2.6	2308	3	2562	2.3	2091	3	2562
Damped	5.2	5085	11	12902	3.9	3619	7	6994	3.1	2769	5	4670	2.3	2185	4	3508
Chord	5.7	2259	8	2920	4.5	1911	6	2346	3.8	1703	5	2059	3.4	1587	4	1772
New It.	8.4	6176	9	6568	8.4	6177	9	6568	8.4	6156	9	6568	8.0	5922	9	6568
Capped	2.9	2808	3	2854	2.8	2726	3	2854	2.6	2583	3	2854	2.3	2352	3	2854
Rangemaster, $V_{peak} = 300\,\mathrm{mV}$																
Newton	-	-	-	-	-	-	-	-	2.8	2427	41	27110	2.5	2241	19	12898
Damped	6.4	7013	19	23962	5.0	5122	20	25124	3.9	3852	23	28610	3.0	2867	22	27448
Chord	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
New It.	8.4	6169	12	8506	8.4	6177	13	9152	8.4	6151	13	9152	8.0	5922	13	9152
Capped	3.8	3434	26	18678	3.2	3006	21	15238	2.7	2667	21	15238	2.5	2510	13	9734

[13] M. Holters and U. Zölzer, "Physical Modelling of a Wah-Wah Pedal as a Case Study for Application of the Nodal DK Method to Circuits with Variable Parts," in *Proc. of the 14th Internation Conference on Digital Audio Effects*, Paris, France, Sept. 2011.

[14] C. T. Kelley, *Solving nonlinear equations with Newton's method*, Fundamentals of algorithms. Society for Industrial and Applied Mathematics, Philadelphia, 2003.

[15] K. Dempwolf and U. Zölzer, "Discrete State-Space Model of the Fuzz-Face," in *Proceedings of Forum Acusticum*, Aalborg, Denmark, June 2011, European Acoustics Association.

[16] F. Eichas, M. Fink, M. Holters, and U. Zölzer, "Physical Modeling of the MXR Phase 90 Guitar Effect Pedal," in *Proc. of the 17 th Int. Conference on Digital Audio Effects (DAFx-14)*, Erlangen, Germany, Sept. 2014.

[17] T. R. Scavo and J. B. Thoo, "On the Geometry of Halley's Method," *The American Mathematical Monthly*, vol. 102, no. 5, pp. 417, May 1995.

[18] R. P. Brent, "An Algorithm with Guaranteed Convergence for Finding the Zero of a Function," *The Computer Journal*, vol. 14, no. 4, pp. 422–425, 1971.

[19] T. Banwell and A. Jayakumar, "Exact analytical solution for current flow through diode with series resistance," *Electronics Letters*, vol. 36, no. 4, pp. 291–292, Feb. 2000.

[20] R. C. D. Paiva, S. D'Angelo, J. Pakarinen, and V. Valimaki, "Emulation of Operational Amplifiers and Diodes in Audio Distortion Circuits," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 10, pp. 688–692, Oct. 2012.

[21] D. T. Yeh and J. O. Smith, "Simulating guitar distortion circuits using wave digital and nonlinear state-space formulations," *Proc. of the Digital Audio Effects (DAFx'08)*, pp. 19–26, 2008.

[22] "LTspice IV," [Online]. Available: `http://www.linear.com/ltspice` - accessed 17/05/2015.

[23] T. Minka, "The Lightspeed Matlab Toolbox," [Online]. Available: `http://research.microsoft.com/en-us/um/people/minka/software/lightspeed/` - accessed 22/04/2015.

[24] J. L. Hennessy and D. A. Patterson, "Instruction-Level Parallelism: Concepts and Challenges," in *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann/Elsevier, Waltham, MA, 5th edition, 2012.

GUARANTEED-PASSIVE SIMULATION OF AN ELECTRO-MECHANICAL PIANO: A PORT-HAMILTONIAN APPROACH

Antoine Falaize, *

IRCAM, CNRS (UMR 9912)
Analysis/Synthesis Team
Paris, France
antoine.falaize@ircam.fr

Thomas Hélie,

IRCAM, CNRS (UMR 9912)
Analysis/Synthesis Team
Paris, France
thomas.helie@ircam.fr

ABSTRACT

This paper deals with the time-domain simulation of a simplified electro-mechanical piano. The physical model is composed of a hammer (nonlinear component), a cantilever beam (damped linear resonator) and a pickup (nonlinear transducer). In order to ensure stable simulations, a method is proposed, which preserves passivity, namely, the conservative and dissipative properties of the physical system. This issue is addressed in 3 steps. First, each physical component is described by a passive input-output system, which is recast in the port-Hamiltonian framework. In particular, a passive finite dimensional model of the Euler-Bernoulli beam is derived, based on a standard modal decomposition. Second, these components are connected, providing a nonlinear finite dimensional port-Hamiltonian system. Third, a numerical method is proposed, which preserves the power balance and passivity. Numerical results are presented and analyzed.

1. INTRODUCTION

This paper address the time-domain simulation of a multi-physics musical instrument, namely, an electro-mechanical piano. A particular attention is devoted to preserving the passivity of the original physical system, especially in the discrete-time domain (that is, no energy is artificially created in the numerical system during the simulation). Such an approach has been considered in *e.g.* [1, 2], with an emphasis on numerical issues. In contrast, other methods such as *physically informed sound synthesis* [3, 4], can lead to numerical systems whose stability is difficult to ensure.

Here, we consider the *port-Hamiltonian* approach, introduced in the 1990's [5, 6, 7]. Port-Hamiltonian systems are extensions of classical Hamiltonian systems [8]: they model open dynamical systems made of energy storage components, dissipative components, and some connection ports through which energy can transit. This leads to a state-space representation of multiphysics systems structured according to energy flow, thus encoding the passivity property including for nonlinear cases.

As depicted in figure 1, the model is composed of a hammer $\mathcal{H}$ (nonlinear component with hysteresis), a cantilever beam $\mathcal{B}$ (damped linear infinite dimensional resonator) and a pickup $\mathcal{P}$ (nonlinear mechano-electric transducer). In addition, the pickup is connected to a TLC analog filter (not shown in figure 1). The physical modeling of those elements are available in the literature and are recalled.

This paper is organized as follows. The port-Hamiltonian (pH) framework is recalled in section 2 and we show how this formalism ensures the passivity of the models in continuous time. Section 3 describes the lumped components $\mathcal{H}$, $\mathcal{P}$ and the RLC circuit, and provides their modeling into pH formalism Section 4 is devoted to the pH formulation of the Euler-Bernoulli beam $\mathcal{B}$ through its finite-dimensional approximation, based on a standard modal decomposition. These components are connected in section 5, providing a nonlinear finite-dimensional port-Hamiltonian system. In section 6, a numerical method is proposed, which preserves the power balance and thus the passivity in discrete time. Numerical results are presented and analyzed in section 7.

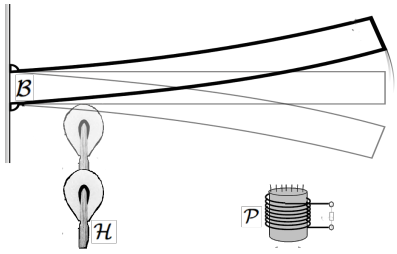


Figure 1: Schematic of the simplified electromechanical piano, with hammer $\mathcal{H}$, beam $\mathcal{B}$ and pickup $\mathcal{P}$.

2. PORT-HAMILTONIAN SYSTEMS

In this section, we introduce the *port-Hamiltonian* (pH) formalism [5, 6, 7] and present an introductory example. It is shown how this structure guarantees the passivity of the model in continuous time.

2.1. Considerations on energy and passivity

Denote $\mathtt{E}(t) \geq 0$ the energy stored in an open physical system. If the system is autonomous and conservative, its time variation $\frac{\mathtt{dE}}{\mathtt{d}t}(t)$ is zero. If the system is controlled (non-autonomous) and conservative, $\frac{\mathtt{dE}}{\mathtt{d}t}(t)$ is the power $\mathtt{S}(t)$ received from the sources through the external ports. If the system includes dissipative phenomena with dissipated power $\mathtt{D}(t) \geq 0$, the evolution of energy satisfies the following *power balance*:

$$\frac{\mathtt{dE}}{\mathtt{d}t}(t) = -\mathtt{D}(t) + \mathtt{S}(t). \tag{1}$$

* The contribution of both authors has been done at IRCAM Laboratory, Paris, within the context of the French National Research Agency sponsored project HAMECMOPSYS. Further information is available at http://www.hamecmopsys.ens2m.fr.

Such systems are passive in the sens that $\frac{\mathrm{d}\mathtt{E}}{\mathrm{d}t} \leq \mathtt{S}$. In particular, if the sources are not activated, $\frac{\mathrm{d}\mathtt{E}}{\mathrm{d}t} \leq 0$. The dynamic input-to-output behavior of such system is the result of the intricate power exchange between isolated lumped or distributed components. For finite-dimensional systems, those components are sorted as (or can be a combination of): s components that store energy $\mathtt{E} \geq 0$ (moving mass, capacitors), d components that dissipate power $\mathtt{D} \geq 0$ (mechanical damping, transistors), p external ports that convey power $\mathtt{S}$ ($\in \mathbb{R}$) from sources (mechanical loads, 9V batteries) or any external system (active, dissipative or mixed). The behavior of each component is described by a relation between two sets of *power variables*: flows $\mathfrak{f}$ (velocities, currents, variation of magnetic flux) and the associated efforts $\mathfrak{e}$ (forces, voltages, magnetomotive forces). Based on *receiver convention*, the received power is $\mathtt{P} = \mathfrak{f}^\intercal \mathfrak{e}$.

The total stored energy is expressed as a *storage functional* (Hamiltonian) H of an appropriate *state* $\mathbf{x}$. It is built from the sum of the locally stored energies $\mathtt{E} = \mathrm{H}(\mathbf{x}) = \sum_{n=1}^{s} \mathrm{H}_n(\mathbf{x}_n)$. Typically, for a mass m, the state can be the momentum $x = m\frac{\mathrm{d}q}{\mathrm{d}t}$ and the positive definite function is $\mathrm{H}_m(x) = \frac{1}{2m}x^2$. Storage power variables $(\mathfrak{f}_\mathcal{S}, \mathfrak{e}_\mathcal{S})$ are related to the variation of the state $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$ and the gradient of the storage function $\nabla\mathrm{H}(\mathbf{x})$, the product of which is precisely the *received power*: $\mathfrak{f}_\mathcal{S}^\intercal \mathfrak{e}_\mathcal{S} = \frac{\mathrm{d}\mathtt{E}}{\mathrm{d}t} = \nabla\mathrm{H}(\mathbf{x})^\intercal \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$. For the mass, it yields $\mathfrak{f}_m = \frac{\mathrm{d}q}{\mathrm{d}t} = H'_m$ and $\mathfrak{e}_m = m\partial_t^2 q = \frac{\mathrm{d}x}{\mathrm{d}t}$.

The total dissipated power is expressed with respect to an appropriate *dissipation variable* $\mathbf{w}$ and is built from the sum of the locally dissipated powers $\mathtt{D}(t) \equiv \mathtt{D}\big(\mathbf{w}(t)\big) = \sum_{n=1}^{d} \mathtt{D}_n\big(\mathbf{w}_n(t)\big)$. Typically, for a fluid-type damper α, w can be a velocity $w = \frac{\mathrm{d}q}{\mathrm{d}t}$ and $\mathtt{D}_\alpha(w) = \alpha w^2$. As for storage components, a mapping of the dissipative power variables $(\mathfrak{f}_\mathcal{D}, \mathfrak{e}_\mathcal{D})$ is provided, based on the factorization $\mathtt{D}(\mathbf{w}) = \mathbf{w}^\intercal \mathbf{z}(\mathbf{w})$, introducing a *dissipation function* $\mathbf{z}$. For the damper, $\mathfrak{f}_\alpha = \frac{\mathrm{d}q}{\mathrm{d}t}$ and $\mathfrak{e}_\alpha = z_\alpha(w) = \alpha w$.

For an input/output system, we arrange source variables $(\mathfrak{f}_\mathcal{P}, \mathfrak{e}_\mathcal{P})$ in two vectors: one is considered as an *input* $\mathbf{u}$ and the other as the associated *output* $\mathbf{y}$ so that the power received form sources is $\mathtt{S} = \mathfrak{f}_\mathcal{P}^\intercal \mathfrak{e}_\mathcal{P}$.

2.2. State-space representation of port-Hamiltonian systems

The algebraic-differential system of equations that governs a passive system is obtained by applying conservation laws (Newton, Kirchhoff) to the interconnection network. This system can be recast into the *port-Hamiltonian systems* formalism [7, eq 2.53]:

$$\underbrace{\left(\begin{array}{c} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} \\ \hline \mathbf{w} \\ \hline -\mathbf{y} \end{array}\right)}_{\mathbf{a}} = \underbrace{\left(\begin{array}{c|c|c} \mathbf{J_x} & -\mathbf{K} & -\mathbf{G_x} \\ \hline \mathbf{K}^\intercal & \mathbf{J_w} & -\mathbf{G_w} \\ \hline \mathbf{G_x}^\intercal & \mathbf{G_w}^\intercal & \mathbf{J_y} \end{array}\right)}_{\mathbf{J}} \cdot \underbrace{\left(\begin{array}{c} \nabla\mathrm{H}(\mathbf{x}) \\ \hline \mathbf{z}(\mathbf{w}) \\ \hline \mathbf{u} \end{array}\right)}_{\mathbf{b}}, \quad (2)$$

where matrices $\mathbf{J_x}$, $\mathbf{J_w}$ and $\mathbf{J_y}$ are skew-symmetric ($\mathbf{J}^\intercal = -\mathbf{J}$), and $\nabla\mathrm{H} : \mathbb{R}^s \to \mathbb{R}^s$ denotes the gradient of the total energy (Hamiltonian) w.r.t. the vector of the states $\mathbf{x}$. The pH system (2) fulfills the definition of passivity (see *e.g.* [9]), according to the following property.

Property 2.1 (Power Balance)**.** *The variation of the total energy* $\mathtt{E} = \mathrm{H}\big(\mathbf{x}\big)$ *of a system governed by (2) is given by (1), with the total dissipated power* $\mathtt{D} = \mathbf{z}(\mathbf{w})^\intercal \cdot \mathbf{w} \geq 0$ *and the total incoming power on external ports* $\mathtt{S} = \mathbf{u}^\intercal \cdot \mathbf{y}$.

Proof. We have $\mathbf{b}^\intercal \cdot \mathbf{a} = \frac{\mathrm{d}\mathtt{E}}{\mathrm{d}t} + \mathtt{D} - \mathtt{S}$. Now $\mathbf{b}^\intercal \cdot \mathbf{a} = \mathbf{b}^\intercal \cdot \mathbf{J} \cdot \mathbf{b} = 0$ since $\mathbf{J}$ is skew-symmetric. □

2.3. Example

Consider the forced mass-spring-damper system in figure 2, with $s = 2$, $d = 1$ and $p = 1$, described as follows. For the mass and the damper, quantities are defined as previously with $x_1 = m\frac{\mathrm{d}q}{\mathrm{d}t}$ and $\mathbf{w} = [\frac{\mathrm{d}q}{\mathrm{d}t}]$. For the (linear) spring k, the state and the positive definite function can be the elongation (here the position of the mass) $x_2 = q$ and $\mathrm{H}_2(q) = \frac{k}{2}q^2$ so that $\mathfrak{e}_k = \mathrm{H}'_2$ and $\mathfrak{f}_k = \frac{\mathrm{d}x_2}{\mathrm{d}t}$. Port variables are arranged as input $\mathbf{u} = [\mathfrak{e}_{\text{ext}}]$ (applied force) and output $\mathbf{y} = [\mathfrak{f}_{\text{ext}}]$. Applying Newton's second law to this simple system yields

$$\left(\begin{array}{c} \mathfrak{e}_m \\ \hline \mathfrak{f}_k \\ \hline \mathfrak{f}_\alpha \\ \hline -\mathfrak{f}_{\text{ext}} \end{array}\right) = \left(\begin{array}{cc|c|c} 0 & -1 & -1 & +1 \\ \hline +1 & 0 & 0 & 0 \\ \hline +1 & 0 & 0 & 0 \\ \hline -1 & 0 & 0 & 0 \end{array}\right) \cdot \left(\begin{array}{c} \mathfrak{f}_m \\ \hline \mathfrak{e}_k \\ \hline \mathfrak{e}_\alpha \\ \hline -\mathfrak{e}_{\text{ext}} \end{array}\right).$$

From the constitutive laws of components, this equation exactly restores the form (2), block by block.

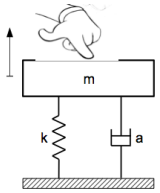


Figure 2: Damped harmonic oscillator with excitation.

3. LUMPED PARAMETERS COMPONENTS

In this this section, the standard modeling of the hammer and the pickup are recast as elementary port-Hamiltonian systems. Those systems are used in section 5 to form the global pH system.

3.1. Hammer $\mathcal{H}$

Here, we consider the standard piano hammer modeling, as depicted in [10, 11, 12]. It is composed of a simple mass with nonlinear spring and hysteresis effects due to the shape memory of the felt [13, 14, 2]. The actuation of the hammer is modeled as a simple force applied on the equivalent mass according to [15]. Denoting by q_h the position of the center of gravity, and m_h the total mass leads to the following dynamics [2]:

$$m_h\partial_t^2 q_h = -f_k(c) - f_\alpha(c) + f_h \quad (3)$$

where c is the crush of the felt, $f_k(c) = k_h c^\beta$ is the nonlinear spring effect, $f_\alpha(c) = \alpha_h \frac{\mathrm{d}(c^\beta)}{\mathrm{d}t}$ models the hysteresis effect and f_h is an external force. The contact with the beam is distributed according to $\omega(z) = 1(z - z_h)_{[-a_h/2, +a_h/2]}$, denoting by $z \in [0, l_b]$ the spacial coordinate along the beam with length l_b, and z_h and a_h respectively the position and width of the hammer [2]. The crush is $c = \lfloor l_h + q_h - \widehat{q}_b \rfloor_0$ with l_h the distance between the top of the felt at rest and q_h, $\lfloor x \rfloor_0$ the max of x and 0, and $\widehat{q}_b = \int_0^{l_b} q(z)\omega(z)\mathrm{d}z$ the weighted average of the beam's transverse displacement $q(z, t)$ (detailed in section 4). The pH model of the hammer (for instance disconnected from the beam) is derived as in example 2.3, and reads as follows.

Port-Hamiltonian modeling of the hammer $\mathcal{H}$

$$\mathbf{x}_\mathcal{H} = (m_h \tfrac{\mathtt{d}q_h}{\mathtt{d}t}, c)^\intercal,$$
$$\mathrm{H}_\mathcal{H}(\mathbf{x}_\mathcal{H}) = \tfrac{1}{2m_h}[\mathbf{x}_\mathcal{H}]_1^2 + \tfrac{k_h}{(\beta+1)}\lfloor[\mathbf{x}_\mathcal{H}]_2 + l_h\rfloor_0^{\beta+1}$$
$$\mathbf{w}_\mathcal{H} = \tfrac{\mathtt{d}c}{\mathtt{d}t}, \mathbf{z}_\mathcal{H}(\mathbf{w}_\mathcal{H}, \mathbf{x}_\mathcal{H}) = \tfrac{\alpha_h}{\beta}\lfloor[\mathbf{x}_\mathcal{H}]_2 + l_h\rfloor_0^{\beta-1}\mathbf{w}_\mathcal{H}$$
$$\mathbf{u}_\mathcal{H} = f_h, \quad \mathbf{y}_\mathcal{H} = \tfrac{\mathtt{d}q_h}{\mathtt{d}t}$$
$$\mathbf{J}_\mathbf{x} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \mathbf{K} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \mathbf{G}_\mathbf{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$
$$\mathbf{J}_\mathbf{w} = 0, \mathbf{G}_\mathbf{w} = 0, \mathbf{J}_\mathbf{y} = 0$$

3.2. Pickup $\mathcal{P}$

Several arrangement of physically inspired signal processing modules to recover the nonlinear behavior of electric guitar pickups are available (see *e.g* [16, 17]). Here we consider the physical modeling approach of [18, 19]. According to the gyrator-capacitor approach [20, 21], we adopt the magnetic flux variation $\frac{\mathtt{d}\phi}{\mathtt{d}t}$ and the magnetomotive force h as magnetic power variables $\frac{\mathtt{d}\phi}{\mathtt{d}t}h = \mathtt{P}$. The magnet produces a constant magnetic field $h_0 = \frac{b_0}{\mu_0}$ which penetrates and magnetizes the beam (made of ferromagnetic material). This leads to consider in first approximation the beam as a magnetic dipole with constant amplitude. The flux ϕ_c of the total magnetic field in the coil is then the sum of the flux from the magnet $\phi_0 = a_c\mu_0 h_0$ and the flux from the magnetized beam which depends on the distance with the coil (of cross section a_c). Here, we consider a single vertical polarization of the beam, and the model in [19] reads:

$$\phi_c = \left(a_c + \frac{\Delta_\mu a_b}{(q_p(z_p) + l_p)^2}\right)\phi_0 \tag{4}$$

where $\Delta_\mu = \frac{\mu_{\text{rel}}-1}{\mu_{\text{rel}}+1}$, l_p is the distance from the beam at rest, q_p the vertical displacement of the beam measured at point z_p, and a_b is the section area of the beam. Thus, the magnet acts as a conservative gyrator modulated by the beam movement:

$$\begin{pmatrix} \frac{\mathtt{d}\phi_c}{\mathtt{d}t} \\ h_c \end{pmatrix} = \begin{pmatrix} 0 & f_p \\ \frac{1}{f_p} & 0 \end{pmatrix}\begin{pmatrix} \frac{\mathtt{d}\phi_0}{\mathtt{d}t} \\ h_0 \end{pmatrix}. \tag{5}$$

with $(\frac{\mathtt{d}\phi_c}{\mathtt{d}t}, h_c)$ the power variables associated to the ferromagnetic core of the coil, $q(z_p)$ the displacement of the beam measured at the pick-up position z_p, l_p the distance between the pickup and the beam at rest, $k_p = 2\Delta_\mu a_b a_c \mu_0$ and

$$f_p\left(q(z_p,t), \frac{\mathtt{d}q(z_p,t)}{\mathtt{d}t}\right) = \frac{k_p}{(q(z_p,t)+l_p)^3}\frac{\mathtt{d}q(z_p,t)}{\mathtt{d}t}. \tag{6}$$

From Ampere's theorem and Faraday's law, the coil is a constant magneto-electric gyrator [20, 21]:

$$\begin{pmatrix} v_c \\ i_c \end{pmatrix} = \begin{pmatrix} 0 & N \\ \frac{1}{N} & 0 \end{pmatrix}\begin{pmatrix} \frac{\mathtt{d}\phi_c}{\mathtt{d}t} \\ h_c \end{pmatrix}. \tag{7}$$

with N the number of wire turns and (v_c, i_c) the tension and current in the coil. Thus, the pickup acts as a modulated voltage source to the electronic circuit. Here we consider a simple RLC circuit, as shown in figure 3, with a high resistive load as mentioned in [18] so that the input current is $i_0 = 0$. Finally the port-Hamiltonian model of the pick-up is made of a two storage components (inductance and capacitance, with magnetic flux ϕ_L and charge q_C as respective states), a single dissipative component (resistance of the wire) and two ports (constant magnetomotive force and input current), as detailed below and in figure 3.

Port-Hamiltonian modeling of the pick-up $\mathcal{P}$

$$\mathbf{x}_\mathcal{P} = (\phi_L, q_C)^\intercal,\ \mathrm{H}_\mathcal{P}(\mathbf{x}_\mathcal{P}) = \tfrac{1}{2L_p}[\mathbf{x}_\mathcal{P}]_1^2 + \tfrac{1}{2C_p}[\mathbf{x}_\mathcal{P}]_2^2$$
$$\mathbf{w}_\mathcal{P} = i_r, \quad \mathbf{z}_\mathcal{P}(\mathbf{w}_\mathcal{P}) = r_p i_r$$
$$\mathbf{u}_\mathcal{P} = (h_0, i_0)^\intercal, \quad \mathbf{y}_\mathcal{P} = (\tfrac{\mathtt{d}\phi_0}{\mathtt{d}t}, v_0)^\intercal$$
$$\mathbf{J}_\mathbf{x} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \mathbf{K} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \mathbf{G}_\mathbf{x} = \begin{pmatrix} Nf_p & 0 \\ 0 & 1 \end{pmatrix}$$
$$\mathbf{J}_\mathbf{w} = 0, \mathbf{G}_\mathbf{w} = \mathbb{0}, \mathbf{J}_\mathbf{y} = \mathbb{0}$$

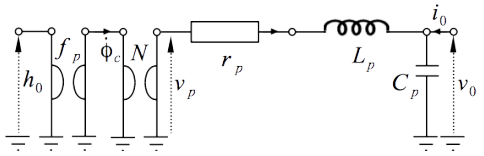


Figure 3: Schematic of the pick-up, where the voltage due to the beam movement is $v_p = Nf_p h_0$ with f_p as in equation (6).

4. BEAM $\mathcal{B}$

Here, we shall model the beam $\mathcal{B}$ in the port-Hamiltonian framework. We use the Euler-Bernoulli modeling of cantilever beam with damping, which results in a linear partial-differential equation. Although infinite dimensional systems perfectly fit in the pH framework (see [22, 23]), we firstly apply a standard modal decomposition and recast the resulting set of ordinary differential equations as a finite dimensional pH system.

4.1. Euler-Bernoulli modeling

The Euler-Bernoulli modeling of damped beam deflect $q(z,t)$ is

$$\rho\partial_t^2 q + \alpha\partial_t q + \kappa\partial_z^4 q = f \tag{8}$$

with initial conditions $q(z,0) = \frac{\mathtt{d}q}{\mathtt{d}t}(z,0) = 0$, and the following boundary conditions: no displacement at the base (bc1) $q(0,t) = 0$, no bending at the base (bc2) $\partial_z q(0,t) = 0$, no bending moment at the free end (bc3) $\partial_z^2 q(1,t) = 0$, no shearing force acting at the free end (bc4) $\partial_z^3 q(1,t) = 0$. ρ is a mass, κ is a stiffness and α is a fluid-like damping, each defined per unit length. Note the total internal energy is given by [24]:

$$\mathtt{E}_\mathcal{B} = \frac{1}{2}\int_0^{l_b}\left(\kappa\left(\partial_z^2 q\right)^2 + \rho\left(\frac{\mathtt{d}q}{\mathtt{d}t}\right)^2\right)\mathtt{d}z. \tag{9}$$

4.2. Finite dimensional approximation

The linear boundary value problem (8) admits an orthogonal basis of eigenfunctions $\mathcal{B} = \{\psi_m\}_{m\in\mathbb{N}_*}$ on the Hilbert space $L^2(0, l_b)$. Functions ψ_m which define the spacial modes are given in appendix 11 and shown in figure 4.

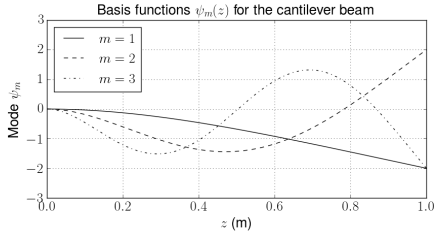


Figure 4: Eigenvectors of the Euler-Bernoulli cantilever beam.

Namely, they satisfy: (i) the boundary conditions (bc1-4); (ii) $\partial_z^4\psi(z) = k^4\psi(z)$; (iii) for all $(m,p) \in \mathbb{N}_*^2$, $\langle \psi_m, \psi_p \rangle = \delta_{m,p}$ (Kronecker's symbol) where the scalar product on $L^2(0,l_b)$ is defined by $\langle f, g \rangle = \int_0^{l_b} f(z)g(z)\mathtt{d}z$. This corresponds to select the modes k_m according to $\cos k_m l_b = \frac{-1}{\cosh k_m l_b}$, (see figure 5).

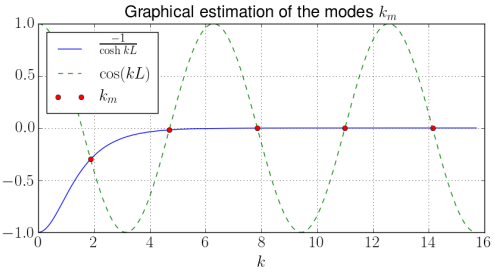


Figure 5: Graphical determination of the k_m.

We define $f = F^\intercal\Psi$, where $\Psi = (\psi_1, \dots, \psi_M)^\intercal$ is the vector of eigenmodes so that $F = (F_1, \dots, F_M)^\intercal = \langle f, \Psi \rangle$ is the projection of f on Ψ, and $\mathbf{q}_\mathcal{B} = \langle q, \Psi \rangle$. The relations satisfied by the $\mathbf{q}_\mathcal{B}$ are obtained by projecting equation (8) on base $\mathcal{B}$. This yields the following ordinary differential equations:

$$\rho\frac{\mathtt{d}^2\mathbf{q}_\mathcal{B}}{\mathtt{d}t^2} + \alpha\frac{\mathtt{d}\mathbf{q}_\mathcal{B}}{\mathtt{d}t} + \kappa\mathbf{L}\mathbf{q}_\mathcal{B} = F, \tag{10}$$

with $\mathbf{L} = \mathrm{diag}(k_1^4, \dots, k_M^4)$ which rewrites $\frac{\mathtt{d}\mathbf{x}_\mathcal{B}}{\mathtt{d}t} = \mathbf{A}\mathbf{x}_\mathcal{B} + \mathbf{B}\mathbf{u}_\mathcal{B}$ with $\mathbf{x}_\mathcal{B} = \left(\mathbf{q}_\mathcal{B}, \rho\frac{\mathtt{d}\mathbf{q}_\mathcal{B}}{\mathtt{d}t}\right)^\intercal$, $\mathbf{B} = (\mathbb{0}, \mathbf{I_d})^\intercal$, $\mathbf{u}_\mathcal{B} = F$ and

$$\mathbf{A} = \begin{pmatrix} \mathbb{0} & \frac{1}{\rho}\mathbf{I_d} \\ -\kappa\mathbf{L} & -\frac{\alpha}{\rho}\mathbf{I_d} \end{pmatrix}$$

where $\mathbb{0}$ is the null matrix and $\mathbf{I_d}$ is the identity matrix. Note the transverse velocity is given by $\frac{\mathtt{d}q(z,t)}{\mathtt{d}t} = v(z,t) = \mathbf{v}(\mathbf{t})^\intercal\mathbf{\Psi}(\mathbf{z})$ with $\mathbf{v}(\mathbf{t}) = \mathbf{B}^\intercal\frac{1}{\rho}\mathbf{x}_\mathcal{B}(\mathbf{t})$.

4.3. Port-Hamiltonian formulation

From (9) and the modal reconstruction $q = \mathbf{q}_\mathcal{B}^\intercal\Psi$, the total energy of the beam is the Hamiltonian $\mathrm{H}_\mathcal{B}(\mathbf{x}_\mathcal{B}) = \frac{1}{2}\mathbf{x}_\mathcal{B}^T\mathbf{W}\mathbf{x}_\mathcal{B}$, with

$$\mathbf{W} = \begin{pmatrix} \kappa\mathbf{L} & \mathbb{0} \\ \mathbb{0} & \rho^{-1}\mathbf{I_d} \end{pmatrix},$$

since $\langle \partial_z^2\Psi, \partial_z^2\Psi^\intercal \rangle = \mathbf{L}$. The output is $\mathbf{y}_\mathcal{B} = \mathbf{B}^\intercal\nabla\mathrm{H}_\mathcal{B} = \frac{\mathtt{d}\mathbf{q}_\mathcal{B}}{\mathtt{d}t}$ so that the incoming power is $\mathtt{P} = \mathbf{y}_\mathcal{B}^\intercal\mathbf{u}_\mathcal{B} = \int_0^{l_b}\mathbf{y}_\mathcal{B}^\intercal\Psi\Psi^T\mathbf{u}_\mathcal{B}\mathtt{d}z$ (with $\mathbf{u}_\mathcal{B} = F$). The resulting port-Hamiltonian system is given below.

Port-Hamiltonian modeling of the beam $\mathcal{B}$

$$\mathbf{x}_\mathcal{B} = \left(\mathbf{q}_\mathcal{B}, \rho\frac{\mathtt{d}\mathbf{q}_\mathcal{B}}{\mathtt{d}t}\right)^\intercal, \quad \mathrm{H}_\mathcal{B}(\mathbf{x}_\mathcal{B}) = \tfrac{1}{2}\mathbf{x}_\mathcal{B}^\intercal\mathbf{W}\mathbf{x}_\mathcal{B}$$
$$\mathbf{w}_\mathcal{B} = \tfrac{\mathtt{d}\mathbf{q}_\mathcal{B}}{\mathtt{d}t}, \quad \mathbf{z}_\mathcal{B}(\mathbf{w}_\mathcal{B}) = \alpha\mathbf{w}_\mathcal{B}$$
$$\mathbf{u}_\mathcal{B} = F = \langle f, \Psi \rangle, \quad \mathbf{y}_\mathcal{B} = \tfrac{\mathtt{d}\mathbf{q}_\mathcal{B}}{\mathtt{d}t}$$
$$\mathbf{J_x} = \begin{pmatrix} \mathbb{0} & \mathbf{I_d} \\ -\mathbf{I_d} & \mathbb{0} \end{pmatrix}, \mathbf{K} = \begin{pmatrix} \mathbb{0} \\ \mathbf{I_d} \end{pmatrix}, \mathbf{G_x} = \begin{pmatrix} \mathbb{0} \\ \mathbf{I_d} \end{pmatrix}$$
$$\mathbf{J_w} = \mathbb{0}, \mathbf{G_w} = \mathbb{0}, \mathbf{J_y} = \mathbb{0}$$

5. INTERCONNECTION

In this section we derive the global port-Hamiltonian modeling of the system $(\mathcal{H}, \mathcal{B}, \mathcal{P})$ from the interconnection of the elementary pH systems derived in sections 3 and 4. First we connect the mechanical components $(\mathcal{H}, \mathcal{B})$ and second the pick-up which is not energetically but geometrically coupled to the former part.

The connection of two pH systems is again a pH system (see [25]). The state $\mathbf{x}$, the dissipative variable $\mathbf{w}$ and dissipative functions $\mathbf{z}$ are obtained by concatenating the subsystems, and the Hamiltonian is the sum of the local Hamiltonians.

5.1. Mechanical part

The input of the pH system that models the beam is the projection of the applied force $f(z,t)$ on the modal basis $\Psi(z)$. When interconnected with the hammer, this is given by the force due to the felt compression $f_k + f_\alpha$ distributed according to $\omega(z)$ (see section 3.1): $\mathbf{u}_\mathcal{B} = F = \Omega f_\mathcal{H}$ with $\Omega = \langle \omega, \Psi \rangle$. Correspondingly, the variation of the felt's crush is computed from the averaged velocity of the beam. This interconnection results in a single-input/single-output pH system (the input being the force applied to the hammer).

5.2. Complete port-Hamiltonian modeling

Finally, we include the modeling of the pickup and analog circuit. As already stated, the mechanical part is not energetically coupled to the electromagnetic part, and the complete modeling is obtained by concatenating the interconnection $(\mathcal{H}, \mathcal{B})$ with the pickup $\mathcal{P}$. This yields the following pH system.

Port-Hamiltonian modeling of the system $\mathcal{H} + \mathcal{B} + \mathcal{P}$

$$\mathbf{x} = (\mathbf{x}_\mathcal{H}, \mathbf{x}_\mathcal{B}, \mathbf{x}_\mathcal{P})^\intercal, \ \mathrm{H}(\mathbf{x}) = \mathrm{H}_\mathcal{H}(\mathbf{x}_\mathcal{H}) + \mathrm{H}_\mathcal{B}(\mathbf{x}_\mathcal{B}) + \mathrm{H}_\mathcal{P}(\mathbf{x}_\mathcal{P})$$
$$\mathbf{w} = (\mathbf{w}_\mathcal{H}, \mathbf{w}_\mathcal{B}, \mathbf{w}_\mathcal{P})^\intercal, \ \mathbf{z}(\mathbf{w}) = \left(\mathbf{z}_\mathcal{H}(\mathbf{w}_\mathcal{H}), \mathbf{z}_\mathcal{B}(\mathbf{w}_\mathcal{B}), \mathbf{z}_\mathcal{P}(\mathbf{w}_\mathcal{P})\right)^\intercal$$
$$\mathbf{u} = (f_h, h_0, i_0)^\intercal, \quad \mathbf{y} = (\tfrac{\mathtt{d}q_h}{\mathtt{d}t}, \tfrac{\mathtt{d}\phi_0}{\mathtt{d}t}, v_0)^\intercal$$
$$\mathbf{J_x} = \begin{pmatrix} 0 & -1 & \mathbb{0} & \mathbb{0} & 0 & 0 \\ 1 & \mathbb{0} & \mathbb{0} & -\Omega^\intercal & 0 & 0 \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbf{I_d} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \Omega & -\mathbf{I_d} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ 0 & 0 & \mathbb{0} & \mathbb{0} & 0 & -1 \\ 0 & 0 & \mathbb{0} & \mathbb{0} & 1 & 0 \end{pmatrix},$$
$$\mathbf{K} = \begin{pmatrix} 1 & \mathbb{0} & 0 \\ 0 & \mathbb{0} & 0 \\ \mathbb{0} & \mathbb{0} & \mathbb{0} \\ -\Omega & \mathbf{I_d} & \mathbb{0} \\ 0 & \mathbb{0} & 1 \\ 0 & \mathbb{0} & 0 \end{pmatrix}, \mathbf{G_x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} \\ 0 & Nf_\mathcal{P} & 0 \\ 0 & \mathbb{0} & 1 \end{pmatrix}$$
$$\mathbf{J_w} = \mathbb{0}, \mathbf{G_w} = \mathbb{0}, \mathbf{J_y} = 0$$

6. NUMERICAL SCHEME

To ensure stable simulation of stable dynamical system $\frac{\mathtt{d}\mathbf{x}}{\mathtt{d}t} = \mathbf{f}(\mathbf{x})$, many numerical schemes focus on the approximation quality of the time derivative, combined with operation of the vector field $\mathbf{f}$. Here, we adopt an alternative point of view, by transposing the power balance (1) in the discrete time-domain to preserve passivity. This is achieved by numerical schemes that provide a discrete version of the chain rule for computing the derivative of $\mathtt{E} = \mathrm{H}\circ\mathbf{x}$. This is the case of Euler scheme, for which first order approximation of the differential applications $\mathtt{d}\mathbf{x}(t, \mathtt{d}t) = \frac{\mathtt{d}\mathbf{x}}{\mathtt{d}t}(t) \cdot \mathtt{d}t$ and

$\mathtt{dH}(\mathbf{x},\mathbf{dx}) = \nabla \mathrm{H}(\mathbf{x})^\intercal \cdot \mathbf{dx}$ on the sample grid $t \equiv kT,\ k \in \mathbb{Z}$ are given by

$$\begin{aligned} \delta\mathbf{x}(k,T) &= \mathbf{x}(k+1) - \mathbf{x}(k), & (11)\\ \delta\mathrm{H}\big(\mathbf{x},\delta\mathbf{x}\big) &= \mathrm{H}\big(\mathbf{x}+\delta\mathbf{x}\big) - \mathrm{H}\big(\mathbf{x}\big) & (12)\\ &= \nabla_d \mathrm{H}\big(\mathbf{x},\mathbf{x}+\delta\mathbf{x}\big)^\intercal \cdot \delta\mathbf{x}. \end{aligned}$$

For mono-variate storage components ($\mathrm{H}(\mathbf{x}) = \sum_{n=1}^{s} \mathrm{H}_n(x_n)$), the solution can be built element-wise with the n-th coordinate given by

$$[\nabla_d \mathrm{H}\big(\mathbf{x},\mathbf{x}+\delta\mathbf{x}\big)]_n = \begin{cases} \frac{h_n\left(x_n+\delta x_n\right) - h_n\left(x_n\right)}{\delta x_n} & \text{if } \delta x_n \neq 0,\\ h_n'(x_n) & \text{otherwise.} \end{cases} \quad (13)$$

A discrete chain rule is indeed recovered

$$\frac{\delta \mathtt{E}(k,T)}{T} = \nabla_d \mathrm{H}\big(\mathbf{x}(k),\mathbf{x}(k+1)\big)^\intercal \cdot \frac{\delta\mathbf{x}(k,T)}{T} \quad (14)$$

so that the following substitution in (2)

$$\begin{array}{rcl} \frac{\mathtt{dx}}{\mathtt{dt}}(t) & \rightarrow & \frac{\delta\mathbf{x}(k,T)}{T}\\ \nabla \mathrm{H}(\mathbf{x}) & \rightarrow & \nabla_d \mathrm{H}\big(\mathbf{x}(k),\mathbf{x}(k+1)\big) \end{array} \quad (15)$$

leads to

$$\begin{aligned} 0 &= \mathbf{b}(k)^\intercal \cdot \mathbf{J} \cdot \mathbf{b}(k) = \mathbf{b}(k)^\intercal \cdot \mathbf{a}(k)\\ &= \underbrace{\left[\nabla_d \mathrm{H}^\intercal \cdot \frac{\delta\mathbf{x}}{\delta t}\right](k)}_{\frac{\delta \mathtt{E}(k,T)}{T}} + \underbrace{\mathbf{z}(\mathbf{w}(k))^\intercal \cdot \mathbf{w}(k)}_{\mathtt{D}(k)} - \underbrace{\mathbf{u}(k)^\intercal \cdot \mathbf{y}(k)}_{\mathtt{S}(k)}. \end{aligned} \quad (16)$$

For pH systems composed of a collection of linear energy storing components with quadratic Hamiltonian $\mathrm{H}_n(x_n) = \frac{x_n^2}{2C_n}$, we define $\mathbf{Q} = \mathrm{diag}(C_1 \cdots C_s)^{-1}$ so that the discrete gradient (13) reads

$$\nabla_d \mathrm{H}\big(\mathbf{x},\mathbf{x}+\delta\mathbf{x}\big) = \mathbf{Q}\left(\mathbf{x}(k) + \tfrac{\delta\mathbf{x}(k)}{2}\right), \quad (17)$$

which restores the midpoint rule. For nonlinear case, (13) leads to another numerical scheme depending on the nonlinearity, still preserving passivity.

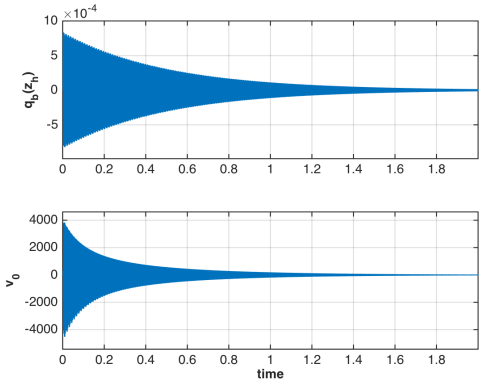


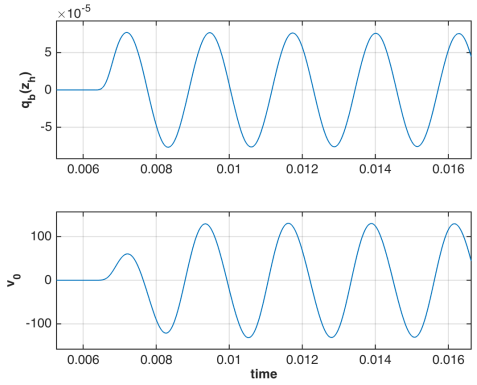
Figure 6: Temporal evolution of the beam displacement measured at pickup position $q(z_h)$ and output voltage v_0 ($f_h = 2000N$).

7. SIMULATION RESULTS

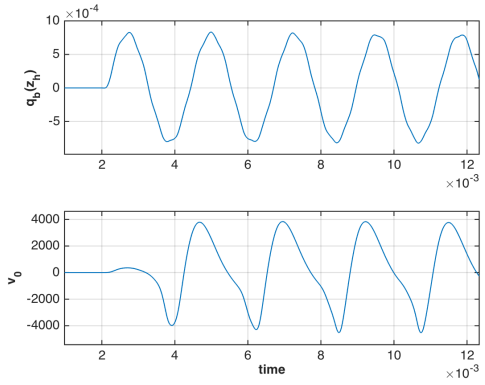
In this section, the numerical scheme (15) is applied on the modeling presented in section 5. First, we discuss the physical parameters used for the simulation. Second, results are shown.

7.1. Physical parameters

The sample rate is 48kHz. The parameters of the hammer are [2]: $m_h = 5\text{g}$, $\beta = 2$, $k_h = 10^6\text{N.m}^{-1}$, $a_h = 1\text{cm}$, $z_h = 2.35\text{cm}$, $\alpha_h = 0.1\text{N.s.m}^{-1}$ and $l_h = 1$. It is supposed initially at rest, 20cm below the beam. The force on the hammer is $f_h = 200\text{N}$ and then 2000N, during 1ms. For a cylindrical bream with radius $r = 2\text{mm}$ made of steel, the mass and stiffness per unit length are respectively $\rho = \varrho\pi r^2\text{kg.m}^{-1}$ and $\kappa = E_s\frac{\pi r^4}{4}$, with $\varrho = 7750\text{kg.m}^{-3}$ the density and $E_s = 180.10^9\text{N.m}^{-2}$ the young modulus of steel. The length l_b is chosen so that the frequency of the first mode (without damping) corresponds to the desired tone, here 440Hz, which yields $l_b = 7.83\text{cm}$. The pickup is supposed to be positioned $l_p = 1\text{mm}$ below the beam, at $z_h = 6.26\text{cm}$, and we set arbitrarily $Nk_ph_0 = 10^-6$. The cutoff frequency of the RLC circuit is 500Hz, with $C_p = 330\text{nF}$, $L_p = 307\text{mH}$ and $r_p = 1\text{k}\Omega$.



(a) $f_h = 200N$.



(b) $f_h = 2000N$.

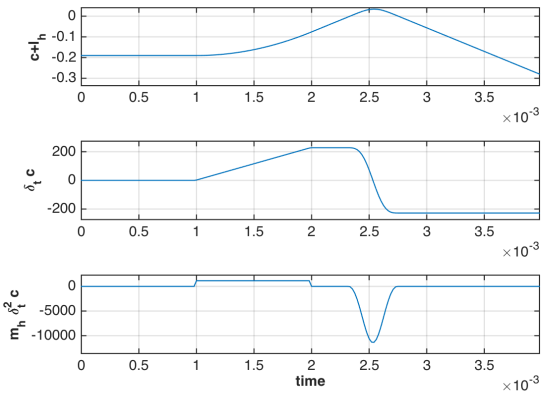
Figure 7: Zoom on the beam displacement measured at pickup position $q(z_h)$ and output voltage v_0.

7.2. Waveforms

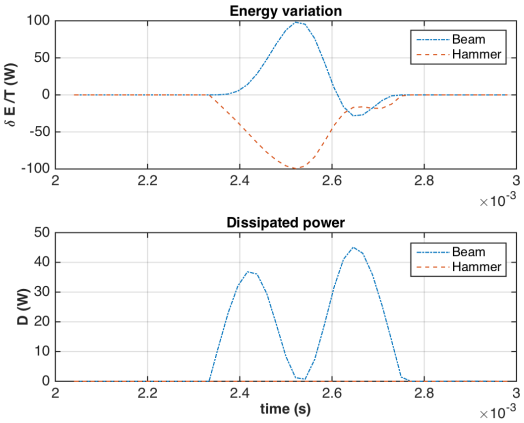
The waveform associated to the beam displacement measured at pickup position $q(z_h)$ and the output voltage v_0 are shown in figures 6. The force applied on the hammer is (i) $f_h = 200$N (figure 7a) and (ii) $f_h = 2000$N (figures 6 and 7b), during 1ms. In case (i), the beam's displacement measured by the pickup $q(z_h)$ corresponds to 6% of the distance pickup-beam l_p, which coincide with the linear behavior. In case (ii), $q(z_h) \simeq 70\% l_p$, which causes the observed change in the waveform.

7.3. Mechanical energy

The dynamics of the hammer is shown in figure 8a for the case (ii) ($f_h = 2000$N). We see it accelerates between 1ms and 2ms and impacts the beam at $t_i \simeq 2.5$ms. During the impact, a part of the energy transferred from the hammer to the beam is dissipated in the later (see figure 8b). The energy in the beam is shown in figure 9. We see the numerical error on the power balance is close to the machine epsilon.



(a) Position, velocity and inertial force of the hammer.



(b) Energy transfer between the hammer and the beam.

Figure 8: Impact hammer $\mathcal{H}$ - beam $\mathcal{B}$

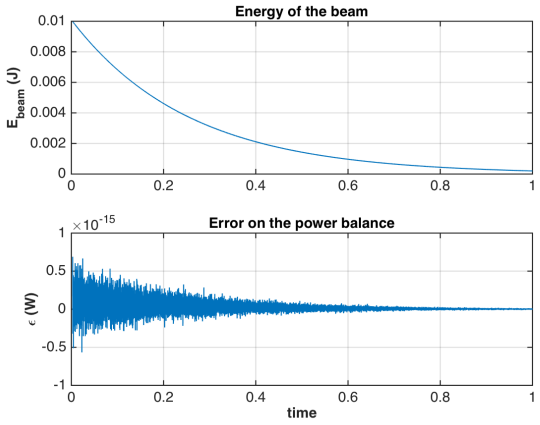


Figure 9: Mechanical energy and error on the power balance $\epsilon = \frac{\delta \mathrm{E}_{beam}}{T} + \mathrm{D}_{beam}$.

7.4. Electromagnetic energy

The source of magnetomotive force is modulated according to section 3.2, which can me modeled as a voltage source (with arbitrary amplitude since the output RLC circuit is linear). Note such a source can be locally a sink of power, as seen in figure 10 where the power of the source pass slightly under 0. Again, the numerical error on the power balance is close to the machine epsilon (figure 11).

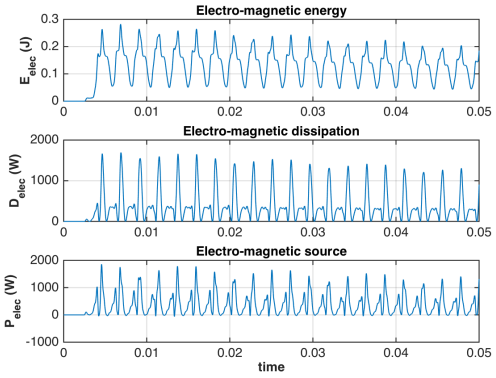


Figure 10: Energy, power dissipation and source of the electro-magnetic part.

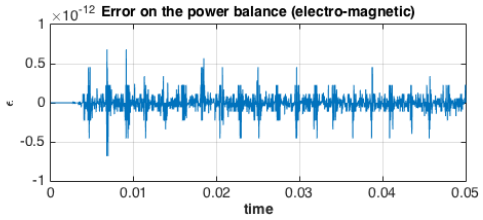


Figure 11: Error on the electro-magnetic power balance.

8. CONCLUSIONS

In this paper, a nonlinear finite-dimensional model of a simplified electro-mechanical piano has been developed, based on a set of elementary components (hammer, beam and pickup), in the framework of port Hamiltonian systems. This formalism decomposes the system into conservative, dissipative and source parts. A numerical method has been derived, which preserves this decomposition and the power balance in the discrete time domain. The analysis of numerical results proves the relevancy of the method: first, the nonlinearity provides an sound enrichment with the force of the hammer; second, the analysis of the power exchanges and of the total energy shows that passivity is fulfilled.

A perspective of this work is to refine the modeling of the mechano-electric transducer. First, the pickup could be placed in the axis of the beam, as in the case of the Fender Rhodes piano. Second, the modeling should include the energetic exchange due to the coupling between the beam and the magnet, by considering the Maxwell force. Another perspective is to estimate the physical parameters from a real device to increase the sound realism. Finally, second order explicit numerical schemes (see *e.g.* [26]) could be examined to improve accuracy and reduces the computational cost.

9. ACKNOWLEDGMENTS

The authors acknowledge the members of the french national research agency project HaMecMoPSys for support in port-Hamiltonian theory, and Cyril Touzé for lectures on modal decomposition.

10. REFERENCES

[1] Stefan Bilbao, Alberto Torin, and Vasileios Chatziioannou, "Numerical modeling of collisions in musical instruments," *Acta Acustica united with Acustica*, vol. 101, no. 1, pp. 155–173, 2015.

[2] Juliette Chabassier, Antoine Chaigne, and Patrick Joly, "Modeling and simulation of a grand piano," *The Journal of the Acoustical Society of America*, vol. 134, no. 1, pp. 648–665, 2013.

[3] Gianpaolo Borin, Davide Rocchesso, and Francesco Scalcon, "A physical piano model for music performance," in *Proceedings: International Computer Music Conference 1997, Thessaloniki, Hellas, 25-30 september 1997*. The International Computer Music Association, 1997, pp. 350–353.

[4] Balázs Bank, "Nonlinear interaction in the digital waveguide with the application to piano sound synthesis," in *Proc. International Computer Music Conference (ICMC'00)*, 2000, pp. 54–57.

[5] BM Maschke, Arjan J Van Der Schaft, and Peter C Breedveld, "An intrinsic hamiltonian formulation of network dynamics: Non-standard poisson structures and gyrators," *Journal of the Franklin institute*, vol. 329, no. 5, pp. 923–966, 1992.

[6] Arjan van der Schaft, "Port-hamiltonian systems: an introductory survey," in *Proceedings of the International Congress of Mathematicians Madrid, August 22–30, 2006*, 2007, pp. 1339–1365.

[7] Vincent Duindam, Alessandro Macchelli, Stefano Stramigioli, and Herman Bruyninckx, *Modeling and Control of Complex Physical Systems: The Port-Hamiltonian Approach*, Springer Science & Business Media, 2009.

[8] Jerrold E Marsden and Tudor Ratiu, *Introduction to mechanics and symmetry: a basic exposition of classical mechanical systems*, vol. 17, Springer Science & Business Media, 2013.

[9] Hassan K Khalil and JW Grizzle, *Nonlinear systems*, vol. 3, Prentice hall New Jersey, 1996.

[10] Neville Horner Fletcher and Thomas D Rossing, *The physics of musical instruments*, Springer Science & Business Media, 1998.

[11] Xavier Boutillon, "Model for piano hammers: Experimental determination and digital simulation," *The Journal of the Acoustical Society of America*, vol. 83, no. 2, pp. 746–754, 1988.

[12] N Giordano and JP Winans II, "Piano hammers and their force compression characteristics: Does a power law make sense?," *The Journal of the Acoustical Society of America*, vol. 107, no. 4, pp. 2248–2255, 2000.

[13] Anatoli Stulov, "Experimental and theoretical studies of piano hammer," in *Proceedings of the Stockholm Music Acoustics Conference*, 2003.

[14] Balázs Bank, Federico Avanzini, Gianpaolo Borin, Giovanni De Poli, Federico Fontana, and Davide Rocchesso, "Physically informed signal processing methods for piano sound synthesis: a research overview," *EURASIP Journal on Applied Signal Processing*, vol. 2003, pp. 941–952, 2003.

[15] Harry C Hart, Melville W Fuller, and Walter S Lusby, "A precision study of piano touch and tone," *The Journal of the Acoustical Society of America*, vol. 6, no. 2, pp. 80–94, 1934.

[16] Rafael CD Paiva, Jyri Pakarinen, and Vesa Välimäki, "Acoustics and modeling of pickups," *Journal of the Audio Engineering Society*, vol. 60, no. 10, pp. 768–782, 2012.

[17] Luca Remaggi, Leonardo Gabrielli, R Cauduro Dias de Paiva, Vesa Välimäki, and Stefano Squartini, "A pickup model for the clavinet," in *Digital Audio Effects Conference 2012 (DAFx-12)*, 2012.

[18] Nicholas G Horton and Thomas R Moore, "Modeling the magnetic pickup of an electric guitar," *American journal of physics*, vol. 77, no. 2, pp. 144–150, 2009.

[19] K. T. McDonald, "Electric guitar pickups," in *Pedagogic note*. May 2007, Princeton University, Department of Physics, `http://puhep1.princeton.edu/~mcdonald/examples/guitar.pdf`,.

[20] David C Hamill, "Lumped equivalent circuits of magnetic components: the gyrator-capacitor approach," *IEEE transactions on power electronics*, vol. 8, no. 2, pp. 97–103, 1993.

[21] David C Hamill, "Gyrator-capacitor modeling: a better way of understanding magnetic components," in *Applied Power Electronics Conference and Exposition, 1994. APEC'94. Conference Proceedings 1994., Ninth Annual*. IEEE, 1994, pp. 326–332.

[22] Alessandro Macchelli and Claudio Melchiorri, "Modeling and control of the timoshenko beam. the distributed port hamiltonian approach," *SIAM Journal on Control and Optimization*, vol. 43, no. 2, pp. 743–767, 2004.

[23] Javier Andres Villegas, *A port-Hamiltonian approach to distributed parameter systems*, Ph.D. thesis, 2007.

[24] Leonard Meirovitch, *Principles and techniques of vibrations*, vol. 1, Prentice Hall New Jersey, 1997.

[25] Joaquín Cervera, AJ Van Der Schaft, and Alfonso Baños, "Interconnection of port-hamiltonian systems and composition of dirac structures," *Automatica*, vol. 43, no. 2, pp. 212–225, 2007.

[26] Nicolas Lopes, Nicolas Hélie, and Antoine Falaize, "Explicit second-order accurate method for the passive guaranteed simu- lation of port-hamiltonian systems.," in *5th IFAC Workshop on Lagrangian and Hamiltonian Methods for Non Linear Control (Lyon, France)*, July 2015, Accepted.

11. APPENDIX: ORTHONORMAL BASIS

The linear boundary value problem (8) admits an orthogonal basis of eigenfunctions $\mathcal{B} = \{\psi_m\}_{m\in\mathbb{N}_*}$ on the Hilbert space $L^2(0, l_b)$, namely, the spacial modes $\psi_m(z)$ which satisfy: (i) the boundary conditions (bc1-4); (ii) $\partial_z^4\psi(z) = k^4\psi(z)$; (iii) for all $(m, p) \in \mathbb{N}_*^2$, $\langle\psi_m, \psi_p\rangle = \delta_{m,p}$ (Kronecker's symbol), where the scalar product on $L^2(0, l_b)$ is defined by $\langle f, g\rangle = \int_0^{l_b} f(z)g(z)\mathtt{d}z$. This corresponds to select the modes k_m according to $\cos k_m l_b = \frac{-1}{\cosh k_m l_b}$:

$$
\begin{aligned}
\psi_m(z) &= \gamma\widehat{\psi}_m(z)\\
\widehat{\psi}_m(z) &= \theta_m\left(\sin k_m z - \sinh k_m z\right) + \cos k_m z - \cosh k_m z\\
\theta_m &= \tfrac{\sin k_m l_b - \sinh k_m l_b}{\cos k_m l_b + \cosh k_m l_b}\\
\gamma &= \Big(\tfrac{k_m l_b(\cos 2k_m l_b + \cosh 2k_m l_b - 2)}{2k_m(\cos k_m l_b + \cosh k_m l_b)^2}\\
&\qquad \tfrac{\cosh k_m l_b(2\sin k_m l_b + \cosh k_m l_b \sin 2k_m l_b)}{2k_m(\cos k_m l_b + \cosh k_m l_b)^2}\Big)^{\frac{1}{2}}.
\end{aligned}
$$

ON THE LIMITS OF REAL-TIME PHYSICAL MODELLING SYNTHESIS WITH A MODULAR ENVIRONMENT

Craig J. Webb
Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
craigwebb@physicalaudio.co.uk

*Stefan Bilbao**
Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
sbilbao@ed.ac.uk

ABSTRACT

One goal of physical modelling synthesis is the creation of new virtual instruments. Modular approaches, whereby a set of basic primitive elements can be connected to form a more complex instrument have a long history in audio synthesis. This paper examines such modular methods using finite difference schemes, within the constraints of real-time audio systems. Focusing on consumer hardware and the application of parallel programming techniques for CPU processors, useable combinations of 1D and 2D objects are demonstrated. These can form the basis for a modular synthesis environment that is implemented in a standard plug-in architecture such as an Audio Unit, and controllable via a MIDI keyboard. Optimisation techniques such as vectorization and multi-threading are examined in order to maximise the performance of these computationally demanding systems.

1. INTRODUCTION

A principle objective of physical modelling synthesis is to emulate existing acoustic or analog electronic instruments. However, another wide-ranging goal is to create entirely new instruments that do not necessarily have existing real-world counterparts—yet which generate sounds of a natural character. If the goal is the latter, then one often-used approach is to provide the user (the instrument designer) with a collection of primitive or canonical objects, as well as a means of connecting them, so as to allow for the generation of relatively complex sound-producing objects. Such a modular approach of course has a long history in audio synthesis, and deeper roots in the world of analog electronic synthesizers. In terms of physical modelling synthesis, modularity underlies various synthesis methodologies including mass-spring networks [1, 2] modal synthesis [3], as well as scattering-based structures [4].

A different approach, also allowing for modular design, involves the use of direct spatiotemporal integrators such as finite difference time domain methods. Such methods have various benefits, such as minimal pre-computation and memory usage, and also provable stability conditions [5]—which are extremely important under the very general design conditions that a musician will almost certainly demand. Such methods can be used to create large-scale models of instruments that can be coupled inside a 3D virtual space [6]. However, the simulation of such complex systems is computationally very intensive, to the extent that even with high performance computing sound output can not be produced near the real-time threshold. Simulation for certain systems, however, is now coming within range of real-time. Simpler 1D and 2D linear objects have manageable levels of computation, and can form the basis of a modular system when using multiple connected objects. Nonlinear behaviour can be introduced through the connection elements, leading to a wide range of possible sonic outputs. The purpose of this study is to demonstrate the audio programming techniques used in order to develop a useable real-time system which runs on consumer hardware (i.e., a laptop/basic desktop machine, with implementation in a standard plug-in architecture such as an Audio Unit [7]).

The remaining sections of this paper are as follows: Section 2 presents model equations for stiff strings and plates, and for the nonlinear connections between these objects, as well as a general description of the numerical update equations in the run-time loop. Section 3 examines the maximum size of the most expensive element, the linear plate, that can be computed in one second at a sample rate of 44.1kHz, and the various optimisations that can be used for CPU computation. Section 4 applies the same testing, but within a working Audio Unit plug-in. Finally, Section 5 details the possibilities and issues involved in designing full modular systems using multiple objects, whilst Section 6 gives concluding remarks. A number of audio examples can be found at this webpage: http://www2.ph.ed.ac.uk/∼cwebb2/Site/Plugins.html

2. INSTRUMENT MODELS AND FINITE DIFFERENCE SCHEMES

A complete modular percussion synthesis environment, based on user-specified interconnections among a set of bars and plates, has been described in detail in [8]. In that case, the environment was implemented in the Matlab language, and performance was a long way from being real-time, in all but the simplest configurations. In this section, the model's basic operation is briefly summarized.

2.1. Components

The current system is composed of a collection of stiff strings/bars and rectangular plates, vibrating, in isolation, under linear conditions. For such an object in isolation, and subject to an excitation force, the dynamics are described by the following equation:

$$\mathcal{L}u + g_e f_e = 0 \tag{1}$$

Here, $u(\mathbf{x}, t)$ represents the transverse displacement of the object, as a function of time t, and at spatial coordinate $\mathbf{x} \in \mathcal{D}$. If the object is a stiff string, the spatial domain is a one-dimensional interval $\mathcal{D} = [0, L]$, for some length L, and for a plate, the domain

* This work was supported by the European Research Council, under grant number StG-2011-279068-NESS

is a two-dimensional rectangular region $\mathcal{D} = [0, L_x] \times [0, L_y]$, for side lengths L_x and L_y. See Figure 1.

In the case of a stiff string, the operator $\mathcal{L} = \mathcal{L}_s$ is defined as

$$\mathcal{L}_s = \rho A \partial_t^2 - T\partial_x^2 + EI\partial_x^4 + 2\rho A\sigma_0\partial_t - 2\rho A\sigma_1\partial_t\partial_x^2 \quad (2)$$

Here, ρ is the material density, in kg/m^3, A is cross-sectional area, in m^2, T is tension, in N, E is Young's modulus, in Pa, I is the moment of inertia, in m^4, and σ_0 and σ_1, both non-negative, are parameters allowing for some control over frequency dependant loss. ∂_t and ∂_x represent partial differentiations with respect to time t and the spatial coordinate x, respectively. The stiff string equation must be complemented by two boundary conditions at each end of the domain $\mathcal{D}$. In the current environment, these may be chosen as clamped, simply supported or free, separately at each end of the string. Note that under low tension, the system describes the behaviour of a stiff bar.

In the case of a plate, the operator $\mathcal{L} = \mathcal{L}_p$ is defined as

$$\mathcal{L}_p = \rho H \partial_t^2 + \frac{EH^3}{12\left(1-\nu^2\right)}\Delta^2 + 2\rho H\sigma_0\partial_t - 2\rho H\sigma_1\partial_t\Delta \quad (3)$$

Here, ρ, E, σ_0 and σ_1 are as before, H is thickness, in m, and ν is Poisson's ratio. Here, Δ is the Laplacian operator in two spatial dimensions. The plate equation must be complemented by two boundary conditions at each edge of the domain $\mathcal{D}$. Here, tensioning effects (to simulate a membrane) have not been included, as computational costs become too large for real-time under most conditions—it is not difficult to introduce an extra term above allowing for such tensioning effects.

In either case, the term $g_e f_e$ represents an excitation. Here, $g_e = g_e(\mathbf{x})$ is a spatial distribution representing the region over which the excitation is applied, and $f_e(t)$ is an externally applied force. For typical striking or plucking gestures, $g_e(\mathbf{x})$ is highly localised (and perhaps idealised to a Dirac delta function). For a strike or pluck, f_e is also usually localised. For example, the function f_e defined by

$$f_e(t) = \frac{f_0}{2}\left(1 - \cos\left(q\pi\frac{t-t_0}{T}\right)\right), \qquad t_0 \le t \le t_0 + T \quad (4)$$

and which is zero otherwise is a good approximation to the force history of a strike, occurring at time $t = t_0$, of duration T seconds, and of maximum force f_0 when $q = 2$, and a pluck when $q = 1$.

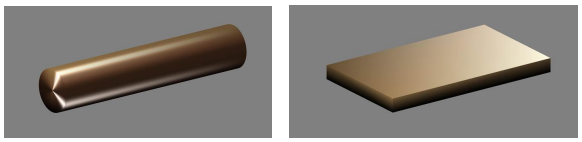


Figure 1: *Basic stiff string or bar (left) and plate (right) elements.*

2.2. Connecting elements

Consider now two components a and b with transverse displacements u_a and u_b, under a single connection:

$$\mathcal{L}_a u_a + g_a f_c = 0 \qquad \mathcal{L}_b u_b - g_b f_c = 0 \quad (5)$$

Here, $\mathcal{L}_a$ and $\mathcal{L}_b$ are linear operators of the types given in (2) and (3), each defined by an appropriate set of material and geometric parameters; the components can be of either stiff string of plate type. Here, g_a and g_b are again distributions, of dimension appropriate to the particular component, selecting a region of interaction for the connection element. $f_c(t)$ is the connection force in N. In particular, it acts in an equal and opposite sense on the two objects (see Figure 2).

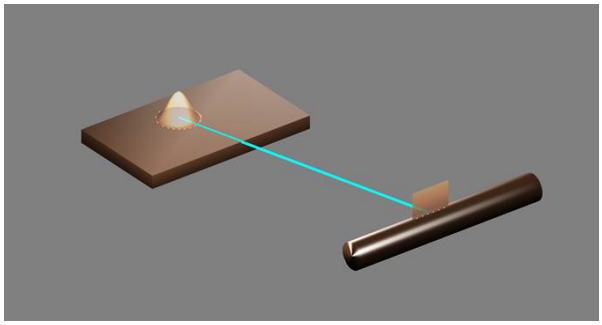


Figure 2: *Connection between stiff string element and plate element.*

Until the form of the connection has been specified, the system is not complete. To this end, define an interaction distance $\eta_c(t)$, averaged over the connection region, by

$$\eta_c = \int_{\mathcal{D}_a} g_a u_a dV_a - \int_{\mathcal{D}_b} g_b u_b dV_b \quad (6)$$

where $\mathcal{D}_a$ and $\mathcal{D}_b$ are the domains of definition of the two components, and where dV_a and dV_b are differential elements of appropriate dimension for the two components. The connection force $f_c(t)$ can be related to the interaction distance $\eta_c(t)$ by

$$f_c = K_1\eta_c + K_3\eta_c^3 + R\frac{d\eta_c}{dt} \quad (7)$$

for connection parameters K_1,K_3 and R, all assumed non-negative. Such a connection may be thought of as a combination of a linear spring, a cubic nonlinear spring, and a linear damper. Many other choices are of course possible, but such a configuration already allows for a wide variety of connection types and, furthermore, in the numerical case, leads to an efficient implementation in which energy-based stability guarantees are available, see [8]. Such numerical stability guarantees are particularly important in the case of arbitrary nonlinear modular networks in the hands of a composer/designer/musician. Once a single excitation and a single connection element have been described, it is not a large step to describe a network composed of a multitude of objects, linked by an arbitrary number of connections, and under an arbitrary number of excitations, as in Figure 3.

2.3. Finite Difference Schemes and State Space Update Form

In a finite difference implementation, each object is represented by a set of values defined over a grid—1D in the case of a stiff string, and 2D in the case of a plate. The complete mechanics of the construction of finite difference schemes for such objects is provided in [8], and is far too involved to be re-presented here, particularly as this paper is concerned mainly with real-time implementation. It is useful nonetheless to briefly describe the vector-matrix update equations for a complete system.

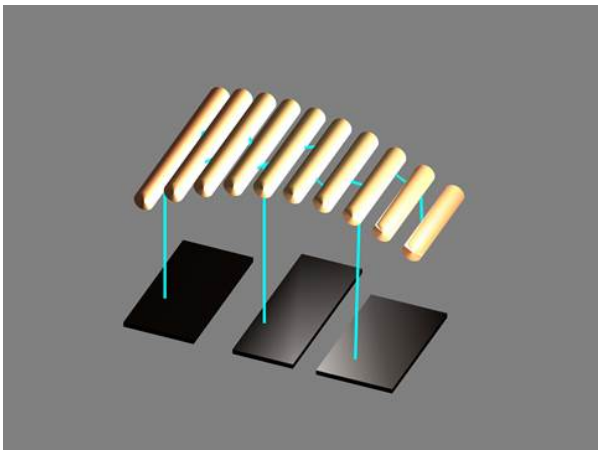


Figure 3: *Interconnected network of plate and stiff string elements.*

Consider an object of either stiff string or plate type in isolation, subject to a single excitation, and for which the dynamics obey (1). Any such object may be approximated by values over a grid—in 1D, for a stiff string, or 2D for a plate (Figure 4). Suppose now that the column vector $\mathbf{u}^n$ represents the concatenation of all values for all system components at time step n; here, time steps are separated by T_s seconds, where the sample rate F_s is defined as $F_s = 1/T_s$. The defining equations are of second order in the time variable t, and in the discrete case, two-step recursions are minimal in terms of memory usage, and are a natural choice.

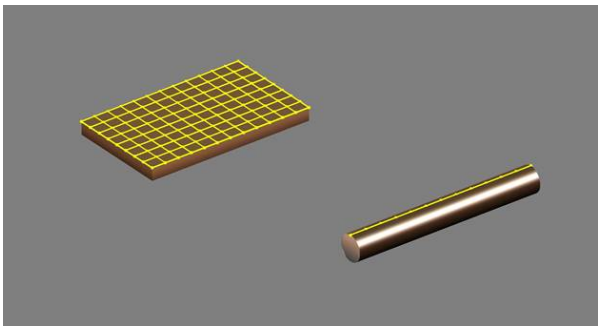


Figure 4: *Grids for plate and stiff string elements.*

The following recursion, operating at an audio sample rate, may be derived from the combination of external excitations, as given in (1) and connections, as per (5), for a system constructed of multiple components and connections, using standard procedures:

$$\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1} - \mathbf{G}_e\mathbf{f}_e^n - \mathbf{G}_c\mathbf{f}_c^n \tag{8}$$

Here the total state size (that of $\mathbf{u}^n$) is N, the total number of connections is N_c, and the total number of separate excitations is N_e. The constant matrices $\mathbf{B}$ and $\mathbf{C}$ are square and $N \times N$, sparse, and incorporate the effects of boundary conditions in the various components. In particular, they may be derived directly from discretisation of the individual defining operators $\mathcal{L}_s$ and $\mathcal{L}_p$ for the various components.

The update (8), including only the terms involving the matrices $\mathbf{B}$ and $\mathbf{C}$ is a simulation of a set of isolated, unforced components. The column vector $\mathbf{f}_e^n$ is of size $N_e \times 1$, and consists of the external forcing signals, which could in principle be of any form, but in the current system are constrained to be sampled from the pluck/strike waveforms given in (4). The constant matrix $\mathbf{G}_e$ is $N \times N_e$, consisting of columns representing the spatial distributions of the various excitations, sampled over the constituent grids. Finally, $\mathbf{f}_c^n$ is an $N_c \times 1$ vector consisting of the connection forces, and, as in the case of the excitations, $\mathbf{G}_c$ is an $N \times N_c$ constant matrix, the columns of which contain the spatial connection distributions.

In the absence of the connection forces, update (8) is a complete recursion for a set of isolated externally-forced objects. The connection forces $\mathbf{f}_c^n$, however, are related nonlinearly to the yet-to-be-determined state $\mathbf{u}^{n+1}$. A semi-implicit choice of discretisation, as discussed in [8] is useful, in that it leads to numerical stablity conditions, and also to the simple expression:

$$\mathbf{A}^n\mathbf{f}_c^n = \mathbf{b}^n \tag{9}$$

where $\mathbf{A}^n = \mathbf{A}\left(\mathbf{u}^n, \mathbf{u}^{n-1}\right)$ and $\mathbf{b}^n = \mathbf{b}\left(\mathbf{u}^n, \mathbf{u}^{n-1}\right)$. Though nonlinear, it may be solved through an inversion of the $N_c \times N_c$ matrix $\mathbf{A}^n$ (which is positive definite by construction); indeed, if the connection spatial distribution functions are non-overlapping, then $\mathbf{A}^n$ is diagonal, and the solution to (9) may be computed using N_c divisions. Once $\mathbf{f}_c$ has been determined, it may then be used to advance the solution in (8).

3. INITIAL COMPUTATIONAL TESTING

The computational complexity of the individual objects, at an audio rate of 44.1 kHz, can be ranked as follows:

1. Connection: ≈ 20 floating-point calculations/time step.
2. Bar / String: $10^2 \sim 10^3$ floating-point calculations/time step.
3. Linear Plate: $10^3 \sim 10^5$ floating-point calculations/time step.

Such operation counts, for the stiff string and plate, follow from the required grid resolution, which is itself dependent on the sample rate. This section details initial testing of the most expensive element, the linear plate, outside of a real-time audio system. As a basic 'benchmark' we can assess the largest size of plate that can be computed within one second of wall clock time, at 44.1kHz. In the absence of any real-time audio overheads, this would be the outer limit of achievable computation; the amount of computation that can be performed in an actual audio plug-in is somewhat less than this, as is shown in Section 4.

As the main objective of this study is to evaluate what is achievable on recent consumer hardware, all testing is performed on an Intel Ivy Bridge Core i7 3770S processor. This has four physical cores, a base clock rate of 3.1GHz, turbo clock rate of 3.9GHz, and a maximum memory bandwidth of 25.6 GB/s. Preliminary testing on a newer Haswell Core i7 showed similar results, with a small improvement of around 5%. The testing performed here is restricted to double precision floating-point—see Section 6 for further discussion on this point. The purpose of this section is to analyse the options for implementing the plate object in C code, and the various optimisation strategies available to the programmer.

Updating the state of each point on the plate requires a weighted sum of thirteen neighbouring points from the previous time step

(See Figure 5), as well as five neighbouring points from two time steps past. The grid also includes a boundary layer, and halo layer of non-updated ghost points. A clamped boundary condition is used for this testing section.

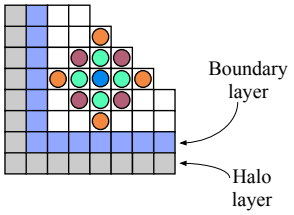


Figure 5: *13 point update stencil, boundary layer and halo.*

In terms of high level design, there are two principal methods for implementing such a scheme: in vector-matrix form, or as an 'unrolled' update equation applied to individual grid points.

3.1. Vector-matrix form

The vector-matrix update form is given by (8), accompanied by the connection force calculation (9). The matrices $\mathbf{B}$, and $\mathbf{C}$ are sparse and (nearly) block Toeplitz. Figure 6 shows the sparsity pattern of the matrix $\mathbf{B}$ corresponding to a single isolated plate. The vectors hold the 2D state data, which is decomposed linearly using either a row or column major approach. Implementing this form in C code is straightforward [9], requiring only a matrix by vector multiplication, and a function for vector addition, to be applied at each time step of the simulation. The CSR (compressed sparse row) format was used as the sparse matrix container.

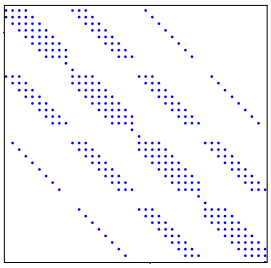


Figure 6: *Sparsity pattern of coefficient matrix* $\mathbf{B}$ *for a single isolated plate.*

3.2. Unrolled update equation

Whilst the vector-matrix form is highly concise (and useful for prototyping in systems such as Matlab), is it clearly somewhat wasteful as it requires storing and reading a large number of constant coefficients. Applying the update equation to individual grid points alleviates this, as factorized scalar coefficients are all that is required. Figure 7 shows a basic implementation of this, inside of the time domain loop. A small optimisation can be achieved by pre-defining the offset constants such as `2*wid`, which are shown here for clarity.

Note that whilst it is possible to use a non-factorized version equivalent to the matrix operations, it is not optimal here (there is no fused multiply-add instruction in Ivy Bridge). This basic implementation in a single thread is unlikely to make the most of the potential of the processor, even when using compiler optimisation. To do that, further manual optimisation is required, in the form of vector instructions and multi-threading.

```
// Time loop
for(n=0;n<Nf;n++){

  // Loop over state data in 2D, as we have boundaries
  for(Y=2;Y<(Ny-1);Y++){
    for(X=2;X<(Nx-1);X++){

    // Calc linear index, from row-major decomposition
    cp = Y*width+X;
    // Calculate update equation
    u[cp] = B1*(u1[cp-1]        + u1[cp+1]
              + u1[cp-width]    + u1[cp+width])
          + B2*(u1[cp-2]        + u1[cp+2]
              +u1[cp-2*width]   + u1[cp+2*width])
          + B3*(u1[cp+width-1]  + u1[cp+width+1]
              +u1[cp-width-1]   + u1[cp-width+1])
          + B4* u1[cp]
          + C1*(u2[cp-1]        + u2[cp+1]
              + u2[cp-width]    + u2[cp+width]
          + C2* u2[cp] );
    }
  }

  // Read output
  out[n] = u[out_location];

  // Swap pointers
  dummy_ptr = u2;
  u2 = u1;
  u1 = u;
  u = dummy_ptr;
}
```

Figure 7: Standard C code time loop and state update.

3.3. AVX vector instructions

AVX instructions make use of register vector units that are capable of computing arithmetic operations over multiple data simultaneously. For example, rather than computing a single multiplication with two input operands, an AVX instruction will perform a number of multiplications using vector operands, and giving a vector result. On Ivy Bridge and Haswell/Broadwell architectures these are 256 bit registers, and so can operate on four double precision data elements at a time (in the upcoming Skylake architecture these are extended to 512 bit registers). Clearly the ability to perform basic arithmetic operations on multiple data at the same time is well suited to finite difference schemes, where each updated element is independent in a given time step. Figure 8 shows the vector implementation of the update scheme, using nested intrinsic functions to compute the stages of the update.

Note that the loop over the state data is now one-dimensional, and is incremented by the vector size. Reducing down to a single `FOR` loop allows the SIMD vector operations to be performed in a continuous manner over the entire state (save for any additional overs at the end). This does require corrections at the boundaries, but these are at a minimal cost, especially when alternative boundary conditions are applied. Further manual unrolling of the loop (i.e. to a size of eight using two sets of updates applied consecutively) did not provide a further speedup.

```
// Time loop
for(n=0;n<Nf;n++){

  // Loop over state data
  for(cp=start;cp<vecend+1;cp+=vector_size){

    // Load up state data into separate vectors
    u1m2   = _mm256_load_pd(&u1[cp-2]);
    u1m1   = _mm256_load_pd(&u1[cp-1]);
    ...
    ...

    // Perform update equation in stages
    s1 = _mm256_mul_pd(B1v,_mm256_add_pd(_mm256_add_pd(
         u1m1, u1p1),_mm256_add_pd(u1mw,u1pw)));

    s2 = _mm256_mul_pd(B2v,_mm256_add_pd(_mm256_add_pd(
         u1m2, u1p2),_mm256_add_pd(u1m2w,u1p2w)));

    s3 = _mm256_mul_pd(B3v,_mm256_add_pd(_mm256_add_pd(
         u1pwp1,u1pwm1),_mm256_add_pd(u1mwp1,u1mwm1)));

    s4 = _mm256_mul_pd(B4v, u1cp);

    s5 = _mm256_mul_pd(C1v,_mm256_add_pd(_mm256_add_pd(
         u2m1, u2p1),_mm256_add_pd(u2mw, u2pw)));

    s6 = _mm256_mul_pd(C2v, u2cp);

    s7 = _mm256_add_pd(_mm256_add_pd(_mm256_add_pd(s1,
         s2), _mm256_add_pd(s3, s4)), _mm256_add_pd(s5,
         s6));

    // Store result
    _mm256_store_pd(&u[cp], s7);

  }

  // Deal with overs at the end of the state size, and
       correct boundaries that were over-written
  ...

  // Read output
  out[n] = u[out_location];

  // Swap pointers
  dummy_ptr = u2; u2 = u1; u1 = u; u = dummy_ptr;
}
```

Figure 8: AVX vectorization time loop.

3.4. Multi-threading

Whilst vector extensions provide a highly effective optimisation, the system is still only making use of a single core of the processor. Even the consumer-based i7 has four cores available, and so a further optimisation is to explore the use of multiple threads that can parallelize the operation over these cores. Again, the finite difference scheme is well suited for this, as the state can simply be partitioned into a suitable number of parts, with each thread operating over an independent section of data at each time step. This can be combined with AVX instructions as used above. Whilst frameworks such as OpenMP can be used to implement multi-threading using compiler directives, this section considers the manual use of POSIX threads (Pthreads).

There are, however, some design issues that need to be considered. The primary concern is how to issue threads that operate over spatial elements of the grid, but also take into account the updates over time. There are two possible approaches. First, one could create the threads at the beginning of each time step, perform the state update, and then destroy them. This would then be repeated each time step, but has the benefit of not requiring any additional thread barrier synchronisation. However, the overheads involved in the approach mean that it only becomes viable when large-scale arrays are used [10]. At the scale of the plates possible in real-time, this approach does not yield any performance benefits.

In order to hide the latencies in issuing threads, we need to create them not at each time step, but only once over many time steps. In the context of real-time audio, creating them at the start of an audio buffer (i.e. every 256 time steps) works well. Figure 9 shows the detail of the time loop.

```
// Time loop
for(int n=0;n<Nf;n+=buffer_size){

  // Create threads
  for (i=0; i<NUM_THREADS; i++) {
    td[i].n = n;
    td[i].u  = u;
    td[i].u1 = u1;
    td[i].u2 = u2;
    pthread_create(&threads[i], NULL, updateScheme, &td[
         i]);
  }

  // Destroy threads
  for (i=0; i<NUM_THREADS; i++) {
    pthread_join(threads[i], NULL);
  }
}
```

Figure 9: Time loop for multi-threading using POSIX.

Note that the time loop is now incremented by the buffer size, and the current time step integer is passed into the thread kernel in order to correctly read the output into an array. The coefficients are loaded into the `td` struct prior to the time loop. As the threads are now operating over time, a barrier is required inside the kernel to ensure that the threads are synchronised prior to performing the pointer swap. The outline of the kernel function is shown in Figure 12. Initial testing revealed that using two threads only provides a small increase over a single thread, eight threads actually perform worse (despite hyper-threading showing eight logical cores), and four is the optimal configuration here.

3.5. Summary

Testing was performed using a square plate at 44.1kHz, with the size varied such that one second of clock time was required to compute 44,100 time steps. Timing functions from `<sys/time.h>` were used, employing 'wall clock' functions that give accurate measurement even when using multi-threading. All codes were compiled using LLVM, with -O3 optimisation (-O0 was also tested as a comparison). Table 1 shows the results for each implementation.

The first result to note is the difference between the matrix form, which achieved 41 x 41, and the standard C code which achieved 80 x 80, nearly four times more efficient. Clearly the extra data movement is a cost, but also the compiler is more likely to aggressively optimise the unrolled version. However, the standard C code is still using only a fraction is the possible CPU performance. The AVX instructions provide a 2X speedup, clearly well below a linear 4X but there are overheads in loading the data points into the vector registers. From there, applying multiple threads achieves a further 35% increase in performance. Again, this is significantly below a linear increase over the single thread version, partly due to the decrease in clock frequency when running over

Code version (-O3 unless otherwise stated)	**Plate size** (grid points)	**Total state size** (grid points)
CSR matrix	41 x 41	1,681
C with -O0	44 x 44	1,936
C	80 x 80	6,400
C with AVX	112 x 112	12,544
C with AVX & 4 Pthreads	130 x 130	16,900

Table 1: *Maximum plate sizes for one second computation time, running for 44100 time steps.*

multiple cores, and also latencies involved in thread synchronisation at each time step. This is typical of multi-threaded codes, where best performance is achieved when parallelising over much larger arrays [10].

The algorithm is clearly memory bandwidth limited, and so it is useful to assess the memory throughput efficiency of the implementation. Taking one read and one write per grid point (assuming all other reads will be from cache), this can be calculated as 130 x130 grid points x 2 x 44100 time steps x 8 bytes = 11.9 GB/sec. This is half of the theoretical maximum, but there does not seem to be any obvious further avenues for optimisation. Note the importance of applying vectorization and multi-threading. Without them, the basic C code with -O3 only achieves 38% of the maximum performance.

4. TESTING WITHIN A REAL-TIME AUDIO UNIT

Audio Units are Apple's native plug-in architecture for enhancing digital audio hosts such as Logic Pro or Ableton. From a programming perspective, they are executable code within a standard API defined by the Core Audio system [11]. The plug-ins can be created either by using a development framework such as JUCE [12], or by directly sub-classing the Core Audio SDK. The latter approach is used here (the AU v2 API), to minimise any overhead involved with the use of additional structures.

The purpose of this section is to determine the maximum size of plate that can be computed from within an actual working plug-in. The basic unrolled code, as well as the AVX and Pthread codes were all tested, with the size of the plate varied to determine the largest size achievable just prior to audio dropouts. Within the real-time system, audio dropouts are caused by the buffer not being written completely, resulting in 'clicks' at the output stream. Testing was performed using the AU Lab host application, which gives visual warnings when buffer under-runs occur. Drop outs typically start at around 90% of CPU load. A buffer size of 512 samples was used, although the load appeared constant across sizes from 128 to 2048 samples. Table 2 shows the resulting plate sizes.

The multi-threaded and vectorized code still produces the best result, but at a reduced margin over the single thread AVX version. However, compared to the testing in the previous section, it achieves only 50% of the maximum plate size. As previous mentioned, this is largely an issue of data size, where smaller arrays achieve less performance benefits. Considering the single thread AVX code, this achieved 65% of the maximum achieved outside of the real-time system, with a size of 90 x 90 compared to 112 x 112. The non-vectorized code performs at 68%, so in general terms we can say that C code that runs in just less than 0.7 seconds

Code version (-O3 unless otherwise stated)	**Plate size** (grid points)	**Total state size** (grid points)
C	66 x 66	4,356
C with AVX	90 x 90	8,100
C with AVX & 4 Pthreads	92 x 92	8,464

Table 2: *Maximum plate size for an Audio Unit instrument plug-in, running at 44.1kHz.*

will be the maximum computable within the real-time plug-in for the single thread case.

5. IMPLEMENTING A COMPLETE SYSTEM

Having established the maximum performance for a single plate within an Audio Unit plug-in, this section examines the potential for real-time modular systems containing multiple object types. Creating a usable system that is capable of producing a wide sonic range requires the combination of multiple stiff strings, connections and plates. Clearly only one full-scale plate will be possible, but there is scope for a large number of additional 1D objects.

Very stiff strings (or bars) can be tuned a fundamental frequency over a range of around two octaves, from C3 to C5. This results in very small state arrays of around twenty elements each. Low stiffness strings require more state, varying from around 30 elements at C5 to around 250 elements for a low C2. Both can benefit from the use of AVX instructions, which result in 2X performance gains. At these sizes, a large number of such objects can be computed, depending on the size of the plate that is used.

Bars can also be used as resonators, when using a low stiffness value, and here the state size can increase to up to a thousand elements. A plate of dimensions comparable to a plate reverberation device (steel, 1.5m x 1.3m and of thickness 2mm) requires a state size of approximately 7000. Testing showed that this still leaves around 20% of the CPU to compute 1D objects and connections. This is sufficient to run a system such as that shown in Figure 10.

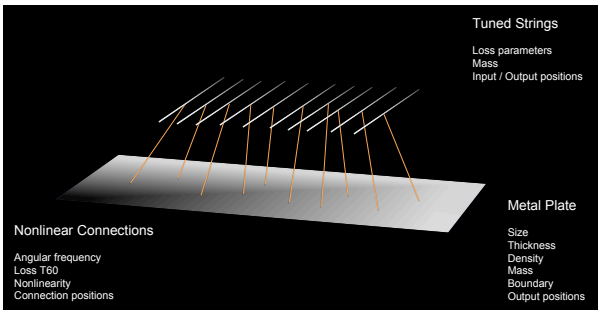


Figure 10: *Tuned string and plate system, with control parameters.*

There are a number of issues involved in actually implementing such a system in a real-time plug-in. The first is the question of parameter updating. Whilst some of the parameters such as the output position and angular frequency of the connections can be varied during real-time computation, others, such as the density of the plate cannot as the underlying grid representation will necessarily be altered. The parameters therefore need to be grouped

into 'state' and 'real-time' controls, and an appropriate system of applying state control changes is required.

The second major issue is how to deal with the large number of parameters that arise. A system consisting of, e.g., 20 stiff string objects, requires 200 parameter settings. This is problematic in terms of user interface design, and requires some level of grouping to be manageable in the first instance. The ability to work with hundreds of parameters may only be possible with a fully-realised visual interface, where individual objects could be manipulated using graphical representations.

There is also the question of the usable parameter space—understanding the combinations and limits of control parameters that lead to usable output. Experimentation in this regard is certainly easier inside a real-time environment. Despite these issues, initial prototype Audio Units have been created, such as systems of pitched bars connected to resonator bars, and tuned strings connected to a large-scale plate. These are playable from a standard MIDI keyboard, and make use of control change signals to vary the real-time parameters. Audio examples can be found at this webpage: http://www2.ph.ed.ac.uk/∼cwebb2/Site/Plugins.html

These prototypes were created using a two-stage process. First, individual C++ objects were created that define a stiff string, a plate, and a connection. These used common interface methods in order to facilitate the second stage, which is to create an instrument model. Here, a number of objects were instantiated using the primitive elements, and a further interface is constructed to allow the instrument to be easily tied into the Audio Unit SDK. Figure 11 shows the class diagram of the string and plate instrument used to create the plug-in.

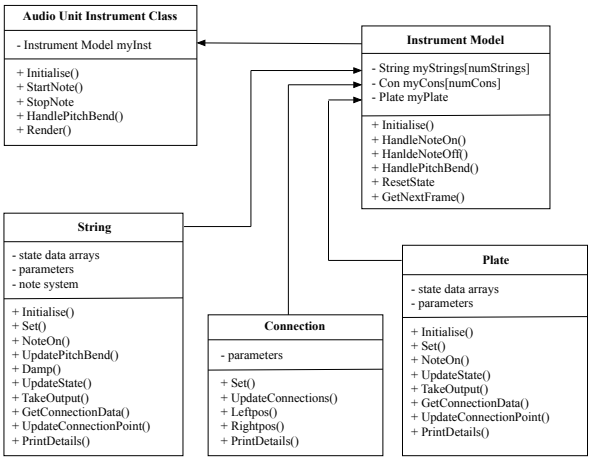


Figure 11: *Class diagram of the string and plate instrument plug-in, showing common interface elements.*

6. CONCLUSIONS AND FUTURE WORK

This paper has demonstrated the use of CPU optimisation techniques to compute coupled 2D and 1D finite difference schemes in real-time, as well as implementation techniques within real-time plug-ins such as Audio Units. Whilst AVX instructions provide a 2X speedup at double precision floating-point, the use of multi-threading is more complex. For the size of plates that can be computed in real-time, the use of multiple threads within a single object does not produce significant performance benefits. A more efficient approach may be to apply threads to separate objects, such as a thread for a plate, and another operating over the 1D objects and connections.

A further avenue for experimentation is the use of single precision floating-point. Although this may result in artefacts in the system due to the nonlinear elements, it would widen the possibilities for computation on consumer hardware. Vector extensions can operate over twice the number of single precision data, which could easily provide performance benefits on the CPU. Also, single precision would allow the use of consumer level GPUs. There have already been some experiments in this regard [13] [14], but with a simpler system consisting of a single 2D wave equation solver. The use of a desktop machine such as Apple's recent Mac Pro has interesting possibilities as it contains two GPUs, both capable of double precision calculations. This would require the use of the OpenCL language, and could also be used to explore multi-core CPU operation.

7. REFERENCES

[1] C. Cadoz, A. Luciani, and J.-L. Florens, "Cordis-anima: A modeling and simulation system for sound and image synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 19–29, 1993.

[2] J.-L. Florens and C. Cadoz, "The physical model: Modeling and simulating the instrument universe," in *Representations of Musical Signals*, G. DePoli, A. Picialli, and C. Roads, Eds., pp. 227–268. MIT Press, Cambridge, Massachusetts, 1991.

[3] D. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–46, 1993.

[4] F. Pedersini, A. Sarti, S. Tubaro, and R. Zattoni, "Towards the automatic synthesis of nonlinear wave digital models for musical acoustics," in *Proceedings of EUSIPCO-98, Ninth European Signal Processing Conference*, Rhodes, Greece, 1998, vol. 4, pp. 2361–2364.

[5] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation*, Wiley, 2009.

[6] S. Bilbao, B. Hamilton, A. Torin, C.J. Webb, P. Graham, A. Gray, K. Kavoussanakis, and J. Perry, "Large Scale Physical Modeling Sound Synthesis," in *Proceedings of the Stockholm Music Acoustics Conference*, Stockholm, Sweden, 2013, pp. 593–600.

[7] Apple Incorporated, "Audio Unit Programming Guide," *[Onine document][Cited: 7th June 2015.] https://developer.apple.com/library/mac/documentation/MusicAudio/Conceptual/AudioUnitProgrammingGuide*, July 2014.

[8] S. Bilbao, "A modular percussive synthesis environment," in *Proceedings of the 12th International Conference on Digital Audio Effects*, Como, Italy, 2009.

[9] G. Golub and C. Van Loan, *Matrix computations (3rd ed.)*, Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[10] C. Webb, *Parallel computation techniques for virtual acoustics and physical modelling synthesis*, Ph.D. thesis, University of Edinburgh, 2014.

[11] W. Pirkle, *Designing Software Synthesizer Plug-Ins in C++*, Focal Press, 2015.

[12] M. Robinson, *Getting started with JUCE*, Packt Publishing, 2013.

[13] B. Hsu and M. Sosnick, "Realtime GPU audio: Finite difference-based sound synthesis using graphics processors," *ACM Queue*, vol. 11, no. 4, May 2013.

[14] M. Sosnick amd B. Hsu, "Implementing a finite difference-based real-time sound synthesizer using GPUs," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, Oslo, Norway, 2011.

```
void *updateScheme(void *targ){

  // Calculate state array partition points for thread
  int psize  = 1 + ( (Nx+1)*(Ny+1)-1)/NUM_THREADS;
  ...

  // Load coefficients into vector objects
  __m256d B1v = _mm256_set1_pd(B1);
  ...

  // Loop over buffer size
  for(int n=0;n<buffer_size;n++){

    // Loop over state data
    for (cp = start;cp<vecend+1;cp+=vector_size){

      // Load up state data into separate vectors
      u1m2   = _mm256_load_pd(&u1[cp-2]);
      u1m1   = _mm256_load_pd(&u1[cp-1]);

      ...

      // Perform update equation in stages
      s1 = _mm256_mul_pd(B1v,_mm256_add_pd(_mm256_add_pd
           (u1m1, u1p1),_mm256_add_pd(u1mw,u1pw)));

      s2 = _mm256_mul_pd(B2v,_mm256_add_pd(_mm256_add_pd
           (u1m2, u1p2),_mm256_add_pd(u1m2w,u1p2w)));

      ...

      // Store result
      _mm256_store_pd(&u[cp], s7);

    }

    // Deal with overs
    ...

    // Read output
    if (tid==0) md->out[...] = u[md->out_location];

    // Barrier
    pthread_barrier_wait (&barrier);

    // Swap pointers
    dummy_ptr = u2; u2 = u1; u1 = u; u = dummy_ptr;

  }

}
```

Figure 12: POSIX thread kernel function.

TWO-POLARISATION FINITE DIFFERENCE MODEL OF BOWED STRINGS WITH NONLINEAR CONTACT AND FRICTION FORCES

Charlotte Desvages, *

Acoustics and Audio Group
The University of Edinburgh
Edinburgh, UK
s1260130@sms.ed.ac.uk

Stefan Bilbao

Acoustics and Audio Group
The University of Edinburgh
Edinburgh, UK
stefan.bilbao@ed.ac.uk

ABSTRACT

Recent bowed string sound synthesis has relied on physical modelling techniques; the achievable realism and flexibility of gestural control are appealing, and the heavier computational cost becomes less significant as technology improves. A bowed string is simulated in two polarisations by discretising the partial differential equations governing its behaviour, using the finite difference method; a globally energy balanced scheme is used, as a guarantee of numerical stability under highly nonlinear conditions. In one polarisation, a nonlinear contact model is used for the normal forces exerted by the dynamic bow hair, left hand fingers, and fingerboard. In the other polarisation, a force-velocity friction curve is used for the resulting tangential forces. The scheme update requires the solution of two nonlinear vector equations.Sound examples and video demonstrations are presented.

1. INTRODUCTION

Physical modelling synthesis for strings debuted in the 1970s, with time stepping methods to discretise the 1D wave equation [1, 2]. However, the very limited computational power at the time ruled out simulation at an audio sample rate in any reasonable amount of time. The next generation of models therefore focussed on algorithmic simplification, through physically plausible assumptions. The non-physical Karplus-Strong string synthesis algorithm [3] was followed by physical digital waveguide models [4]; Karjalainen et al. [5] review the use of these models for string synthesis. Their fast execution and realistic sound output found efficient applications in bowed string modelling, and are still widely used to this day [6, 7, 8, 9, 10]. Another class of physical models relies on the modal solutions of the string equation, and have been successfully adapted for bowed strings [11, 12].

However, the very assumptions that underlie the efficiency of these methods can lead to difficulties when extensions to more realistic settings are desired—the bowed string and its complex interaction with the environment being an excellent example. Time-stepping methods, and more specifically finite difference methods [13], though computationally costly, have regained appeal in musical sound synthesis [14] with the great computing power increase during the last two decades. String simulation in one dimension is particularly suited for these kind of methods [15, 16].

In this work, a linear bowed string is simulated in two polarisations. The model includes full distributed nonlinear contact and friction interactions between the string and the dynamic left hand fingers, dynamic bow, and fingerboard. A stable finite difference scheme for modelling distributed contact/collisions has recently been established [17, 18], that we can use in this stopped string-fingerboard setup [19, 20]. The friction force nonlinearity is modelled with a force/velocity friction curve for the bow [21]; tangential Coulomb friction also keeps the string captured between the fingers and fingerboard during note production. This time domain model allows for full control over the physical parameters of the system, as well as dynamic variations of the playing parameters; it is therefore able to reproduce most bowed string gestures.

In Section 2, the model equations for the bow/string system are presented, with an elaborate description of finger/string interaction in the case of stopped notes, and the string/fingerboard collision interaction. A globally energy balanced finite difference scheme is presented in Section 3. Finally, bowed string simulation results, with the reproduction of several typical gestures, are presented in Section 4. Some sound and video examples from the computed simulations are available online. [1]

2. MODEL DESCRIPTION

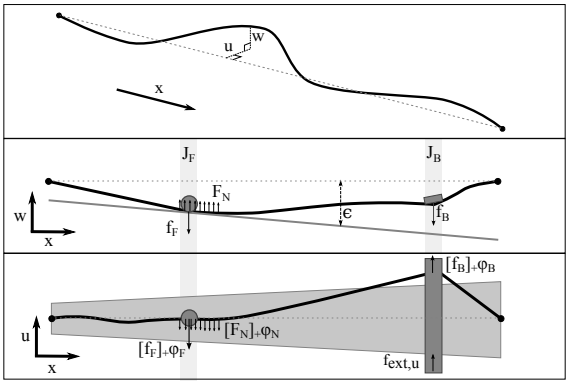


Figure 1: Choice of coordinates for the model. The string is simply supported at both ends. Fingerboard, bow, and left hand finger all interact with the strings in both the vertical and horizontal polarisations. Top: free string. Middle: vertical polarisation. Bottom: horizontal polarisation.

Consider a linear, stiff, and lossy string model in two polarisations. The string displacement in the *vertical* or *normal* polarisation is denoted by $w(x,t)$, while $u(x,t)$ is the string displacement in the

* This work was supported by the Edinburgh College of Art, the Audio Engineering Society, and the European Research Council under grant number StG-2011-279068-NESS.

[1] http://www.ness-music.eu/target-systems/more/bowed-string-instruments

horizontal or *tangential* polarisation. Both are defined for position $x \in \mathcal{D}_S = [0;L]$ and time $t \in \mathbb{R}^+$ (see Figure 1, top).

In this work, consider a single string model, excited by one bow and one finger, and in contact with a fingerboard. The model extends trivially to the case of multiple strings, fingers and bows.

2.1. Vertical polarisation

The partial differential equation governing the time evolution of $w(x,t)$ can be written as:

$$\mathcal{L}w = \mathcal{F}_N - J_F f_F - J_B f_B \tag{1}$$

$\mathcal{L}$ is the partial differential operator defined as [16]:

$$\mathcal{L} = \rho\partial_t^2 - T\partial_x^2 + EI_0\partial_x^4 + \lambda_1\rho\partial_t - \lambda_2\rho\partial_t\partial_x^2 \tag{2}$$

where ρ is the linear mass density of the string, in kg/m; T is the tension of the string, in N; EI_0 is the bending stiffness, where E is Young's modulus in Pa, and $I_0 = \frac{\pi r^4}{4}$ is the area moment of inertia of the circular cross-section of the string, with r the string radius in m; λ_1 (1/s) and λ_2 (m^2/s) are damping coefficients, that empirically account for frequency independent and dependent losses in the string, respectively. $\partial_\cdot^i$ is equivalent to $\frac{\partial^i}{\partial \cdot^i}$.

$\mathcal{L}$ is accompanied by a set of boundary conditions (four of them for the stiff string). We choose standard energy conserving conditions of the simply supported type, assuming an isolated string, with no interaction with the instrument body:

$$w(0,t) = w(L,t) = 0 \qquad \partial_t^2 w(0,t) = \partial_t^2 w(L,t) = 0 \tag{3}$$

The right hand side of Equation 1 contains terms modelling the contact force densities exerted by, respectively, the fingerboard or neck, the left hand finger, and the bow. Their expressions will be elaborated in the following sections.

2.1.1. Fingerboard

$\mathcal{F}_N$ is the contact force density exerted by the neck on the string, along its length (in N/m). Here, a Hunt and Crossley [22] collision model is used, as a smooth approximation to a rigid collision:

$$\mathcal{F}_N(\Delta_N) = \frac{\partial_t \Phi_N}{\partial_t \Delta_N} + \partial_t \Delta_N \Psi_N \tag{4}$$

$\Phi_N(\Delta_N)$ and $\Psi_N(\Delta_N)$ are functions of the *penetration* $\Delta_N(x,t)$, corresponding to the distance by which the edge of the colliding object (here, the fingerboard) would deform from its resting shape:

$$\Phi_N = \frac{K_N}{\alpha_N + 1}[\Delta_N]_+^{\alpha_N+1} \qquad \Psi_N = K_N\beta_N[\Delta_N]_+^{\alpha_N} \tag{5a}$$

$$\Delta_N(x,t) = \varepsilon(x) - w(x,t) \tag{5b}$$

where $K_N > 0$, and $\alpha_N > 1$ are related to the fingerboard stiffness, and $\beta_N > 0$ is a damping coefficient. K_N is chosen very large to approach an ideally rigid collision. $[\cdot]_+$ means $\max(\cdot, 0)$. $\varepsilon(x)$ is the position of the fingerboard with respect to the string at rest (i.e., the action of the instrument; see Figure 1, middle).

2.1.2. Finger and bow

The forces exerted by the finger and the bow onto the string are respectively denoted by $f_F(t)$ and $f_B(t)$. Their action on the string is localised as defined by the continuous distributions $J_F(x,t)$ and $J_B(x,t)$, possibly time-varying (one can use, e.g., a delta Dirac function to model a point wise interaction). $f_F(t)$ and $f_B(t)$ can be written, again, using the Hunt and Crossley model:

$$f_F(\Delta_F) = \frac{\dot{\Phi}_F}{\dot{\Delta}_F} + \dot{\Delta}_F\Psi_F \quad f_B(\Delta_B) = \frac{\dot{\Phi}_B}{\dot{\Delta}_B} + \dot{\Delta}_B\Psi_B \tag{6}$$

where the dot notation is used for total time differentiation ($\frac{d}{dt}$).

$\Phi_F(\Delta_F)$, $\Phi_B(\Delta_B)$, $\Psi_F(\Delta_F)$, and $\Psi_B(\Delta_B)$ are, as for the neck, functions of the penetration $\Delta_F(t)$ and $\Delta_B(t)$:

$$\Phi_F = \frac{K_F}{\alpha_F + 1}[\Delta_F]_+^{\alpha_F+1} \qquad \Psi_F = K_F\beta_F[\Delta_F]_+^{\alpha_F} \tag{7a}$$

$$\Phi_B = \frac{K_B}{\alpha_B + 1}[\Delta_B]_+^{\alpha_B+1} \qquad \Psi_B = K_B\beta_B[\Delta_B]_+^{\alpha_B} \tag{7b}$$

$$\Delta_F(t) = \int_{\mathcal{D}_S} J_F(x,t)w(x,t)dx - w_F(t) \tag{7c}$$

$$\Delta_B(t) = \int_{\mathcal{D}_S} J_B(x,t)w(x,t)dx - w_B(t) \tag{7d}$$

Here, $w_F(t)$ and $w_B(t)$ are respectively the vertical positions of the finger and bow at time t. Their behaviour is governed by:

$$M_F\ddot{w}_F = f_F + f_{\mathrm{ext}w,F} \tag{8a}$$

$$M_B\ddot{w}_B = f_B + f_{\mathrm{ext}w,B} \tag{8b}$$

where M_F, M_B are the finger and bow masses, respectively (in kg), and $f_{\mathrm{ext}w,F}(t)$, $f_{\mathrm{ext}w,B}(t)$ are the resulting external forces applied vertically on the finger and bow, respectively (in N).

2.2. Horizontal polarisation

The tangential displacement of the string $u(x,t)$ obeys:

$$\mathcal{L}u = -[\mathcal{F}_N]_+\varphi_N - J_F[f_F]_+\varphi_F - J_B[f_B]_+\varphi_B \tag{9}$$

where $\mathcal{L}$, $\mathcal{F}_N$, J_F, f_F, J_B, f_B are defined in Section 2.1. Figure 1, bottom, depicts the tangential forces at play. As for the vertical polarisation, the simply supported boundary conditions are:

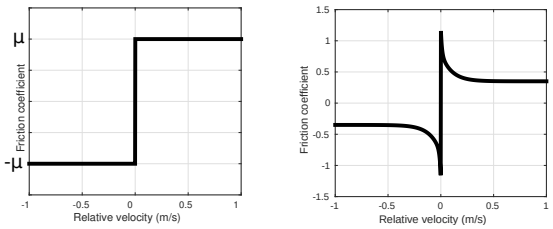
$$u(0,t) = u(L,t) = 0 \qquad \partial_t^2 u(0,t) = \partial_t^2 u(L,t) = 0 \tag{10}$$

φ_N, φ_F and φ_B are friction coefficients, depending on the relative velocity of the string with respect to each object. The resulting friction characteristic, or friction curve, differs for the three objects. Indeed, while φ_B has received a lot of experimental interest for rosin-coated bow hair [23], the friction characteristics of the fingerboard and the human fingers is not well known.

In all three cases, however, the friction coefficient is modulated by the normal force applied on the string (derived from the contact model in Section 2.1). As a result, the interactions in the vertical polarisation feed into the horizontal polarisation. It is important to note that this is the coupling point between the two directions of vibration; although definitely worth investigating, intrinsic and/or boundary coupling between the two polarisations is not included in the present model. We consider that the neck, finger and bow are not adhesive, therefore friction exists only for positive normal forces.

2.2.1. Fingerboard

The fingerboard friction coefficient is distributed along the length of the string. To the authors' knowledge, there is no experimental data allowing us to calibrate this friction curve. As the fingerboard (and, as detailed later in Section 2.2.2, the fingers) serves to capture the string to play notes, we can reasonably assume a Coulomb-like characteristic (illustrated in Figure 2a), where the static friction case occurs in most playing situations:



(a) Coulomb friction characteristic. (b) Friction characteristic for the bow, from Smith et al. [23].

Figure 2: Friction curves for (a) neck and finger, and ((b)) bow.

$$\begin{cases} |\varphi_N(v_{\mathrm{rel},N})| \leqslant \mu_N & \text{if } v_{\mathrm{rel},N} = 0 \text{ (static)} \\ \varphi_N(v_{\mathrm{rel},N}) = \mu_N \operatorname{sign}(v_{\mathrm{rel},N}) & \text{if } v_{\mathrm{rel},N} \neq 0 \text{ (kinetic)} \end{cases} \tag{11a}$$

$$v_{\mathrm{rel},N}(x,t) = \partial_t u \tag{11b}$$

2.2.2. *Fingers*

The fingers have the joint function, along with the fingerboard, of capturing the string to reduce its speaking length, to a crude approximation. Assuming a friction characteristic similar to that of the fingerboard leads to:

$$\begin{cases} |\varphi_F(v_{\mathrm{rel},F})| \leqslant \mu_F & \text{if } v_{\mathrm{rel},F} = 0 \text{ (static)} \\ \varphi_F(v_{\mathrm{rel},F}) = \mu_F \operatorname{sign}(v_{\mathrm{rel},F}) & \text{if } v_{\mathrm{rel},F} \neq 0 \text{ (kinetic)} \end{cases} \tag{12a}$$

$$v_{\mathrm{rel},F}(t) = \frac{d}{dt}\left(\int_{\mathcal{D}_S} J_F(x,t)u(x,t) - u_F(t)\right)dx \tag{12b}$$

$u_F(t)$ is the horizontal position of the fingertip, with respect to the resting string axis. We can hypothesise that the fingertip oscillates about the top finger joint, while simultaneously damping the horizontal vibrations of the string. We can therefore write the temporal evolution of $u_F(t)$ as:

$$M_F \ddot{u}_F = -K_F u_F - \lambda_F \dot{u}_F + [f_F]_+ \varphi_F \tag{13}$$

where λ_F is a damping coefficient. We opt for a linear damped oscillator model for the finger in the horizontal polarisation. Indeed, the choice of a more elaborate contact model such as the one used in the vertical polarisation seems unjustified;while impacts are dominant in the vertical polarisation, e.g. when hammering the string for changing notes, it is clear that collisions only have an auxiliary effect in the tangential polarisation.

2.2.3. *Bow*

The choice of a friction coefficient depending on relative velocity, $\varphi_B(v_{\mathrm{rel},B})$, is somewhat of a trade-off between computational simplification and physical realism. More elaborate models for the bowed string friction interaction, involving viscothermal effects in the rosin layer coating the bow hair, can be used [9, 10]; however, they require significantly more advanced implementations. The friction curve employed here for the bow is indeed deduced from experimental measurements in the steady sliding case (e.g., at constant velocity) [23]; it is illustrated in Figure 2b.

$$\varphi_B = \operatorname{sign}(v_{\mathrm{rel},B})\left(0.4e^{\frac{-|v_{\mathrm{rel},B}|}{0.01}} + 0.45e^{\frac{-|v_{\mathrm{rel},B}|}{0.1}} + 0.35\right) \tag{14a}$$

$$v_{\mathrm{rel},B}(t) = \frac{d}{dt}\left(\int_{\mathcal{D}_S} J_B(x,t)u(x,t) - u_B(t)\right)dx \tag{14b}$$

where $u_B(t)$ is the bow transverse displacement. The bow, as opposed to the finger, does not oscillate around an equilibrium position, but is pushed across the string:

$$M_B \ddot{u}_B = -\lambda_B \dot{u}_B + [f_B]_+ \varphi_B + f_{\mathrm{ext}u,B} \tag{15}$$

where λ_B is a coefficient quantifying the linear energy absorption by the bow hair in the horizontal direction, and $f_{\mathrm{ext}u,B}(t)$ is the force with which the player pushes the bow tangentially, in order to establish the desired bow velocity. Note the slight difference with the usual control parameter in most bowed string studies; instead of directly imposing a bow velocity $v_B(t)$, we use the force applied by the player on the bow, resulting in a bow velocity $\ddot{u}_B$.

2.3. Energy analysis

We can derive a power balance equation for both polarisations. The transfer of this equation to discrete time provides a tool to help ensure numerical stability.

Multiplying Equation 1 by $\partial_t w$ and integrating over the length of the string yields the following power balance (for energy-conserving boundary conditions, such as those given in 3):

$$\dot{H}_w = P_w - Q_w \tag{16}$$

The variation of the total kinetic and potential energy $H_w(t) = H_{w,s}(t) + H_{w,N}(t) + H_{w,F}(t) + H_{w,B}(t)$ is equal to the total power $P_w(t)$ withdrawn from or supplied to the system through external excitation, minus the power $Q_w(t) \geqslant 0$ escaping the system through damping. The system is therefore globally energy conserving. The energy is defined as:

$$H_w = H_{w,s} + H_{w,N} + H_{w,F} + H_{w,B} \tag{17a}$$

$$H_{w,s} = \int_{\mathcal{D}_S}\left[\frac{\rho}{2}(\partial_t w)^2 + \frac{T}{2}(\partial_x w)^2 + \frac{EI_0}{2}\left(\partial_x^2 w\right)^2\right]dx \tag{17b}$$

$$H_{w,N} = \int_{\mathcal{D}_S} \Phi_N dx \qquad H_{w,F,B} = \Phi_{F,B} + \frac{M_{F,B}}{2}\dot{w}_{F,B}^2 \tag{17c}$$

The power supplied through external excitation is:

$$P_w = \dot{w}_F f_{\mathrm{ext}w,F} + \dot{w}_B f_{\mathrm{ext}w,B} + \int_{\mathcal{D}_S}(f_F w \partial_t J_F + f_B w \partial_t J_B)dx \tag{18}$$

The power lost through damping within the string and through collision with the neck, finger and bow is given by:

$$Q_w = Q_{w,s} + Q_\Psi \tag{19a}$$

$$Q_{w,s} = \rho \int_{\mathcal{D}_S}\left[\lambda_1(\partial_t w)^2 + \lambda_2(\partial_t \partial_x w)^2\right]dx \tag{19b}$$

$$Q_\Psi = \int_{\mathcal{D}_S}(\partial_t \Delta_N)^2 \Psi_N dx + \dot{\Delta}_F^2 \Psi_F + \dot{\Delta}_B^2 \Psi_B \tag{19c}$$

In the absence of excitation, the energy H_w strictly decreases.

For the horizontal polarisation, multiplying Equation 10 by $\partial_t u$ and integrating over $\mathcal{D}_S$ yields the power balance:

$$\dot{H}_u = P_u - Q_u \tag{20}$$

Again, the variation of $H_u(t) = H_{u,s}(t) + H_{u,F}(t) + H_{u,B}(t)$ is equal to the total power $P_u(t)$ supplied to or withdrawn from the system in the horizontal polarisation through external excitation, minus power losses $Q_u(t) \geqslant 0$ from damping. The energy is defined as:

$$H_u = H_{u,s} + H_{u,F} + H_{u,B} \tag{21a}$$

$$H_{u,s} = \int_{\mathcal{D}_\mathcal{S}} \left[\frac{\rho}{2}(\partial_t u)^2 + \frac{T}{2}(\partial_x u)^2 + \frac{EI_0}{2}\left(\partial_x^2 u\right)^2 \right] dx \tag{21b}$$

$$H_{u,F} = \frac{M_F}{2}\dot{u}_F^2 + \frac{K_F}{2}u_F^2 \qquad H_{u,B} = \frac{M_B}{2}\dot{u}_B^2 \tag{21c}$$

The power supplied or withdrawn by external excitation is:

$$P_u = [f_F]_+ \varphi_F \int_{\mathcal{D}_\mathcal{S}} u \partial_t J_F dx + [f_B]_+ \varphi_B \int_{\mathcal{D}_\mathcal{S}} u \partial_t J_B dx + \dot{u}_B f_{\mathrm{ext}\,u,B} \tag{22}$$

The power lost through string damping and friction is:

$$Q_u = Q_{u,s} + Q_\varphi + Q_{u,F} + Q_{u,B} \tag{23a}$$

$$Q_{u,s} = \rho \int_{\mathcal{D}_\mathcal{S}} \left[\lambda_1 (\partial_t u)^2 + \lambda_2 (\partial_t \partial_x u)^2 \right] dx \tag{23b}$$

$$Q_\varphi = \int_{\mathcal{D}_\mathcal{S}} v_{\mathrm{rel},N} [\mathcal{F}_N]_+ \varphi_N dx + v_{\mathrm{rel},F} [f_F]_+ \varphi_F + v_{\mathrm{rel},B} [f_B]_+ \varphi_B \tag{23c}$$

$$Q_{u,F,B} = \lambda_{F,B} \dot{u}_{F,B}^2 \tag{23d}$$

Note that $Q_u \geqslant 0$ if $v_{\mathrm{rel}} \varphi(v_{\mathrm{rel}}) \geqslant 0$, which is true for the friction characteristics of the three objects.

The total power of the full system is therefore balanced by:

$$\dot{H} = P - Q \tag{24a}$$

$$H = H_u + H_w \qquad P = P_u + P_w \qquad Q = Q_u + Q_w \tag{24b}$$

3. NUMERICAL SCHEME

We can now discretise the equations of motion by approximating the partial derivation operators with their finite difference [13] counterparts. This method allows a full system simulation, and therefore great flexibility of control for the input parameters and gesture reproduction, at the cost of increased computational requirements. This method has seen a myriad of applications in physical modelling sound synthesis, and more generally musical acoustics simulations [1, 14]. In this section, we define the numerical scheme, detail the discrete energy balance, and describe the scheme update.

3.1. Grid functions and finite difference (FD) operators

All the varying quantities defined in Section 2 are now discretised into *grid functions*, defined at positions $x = lh, l \in \mathfrak{d}_\mathcal{S} = [0,\dots,N]$, and times $t = nk, n \in \mathbb{N}$. h is the *grid spacing*, in m; $k = 1/F_s$ is the *time step*, in s, with F_s the sample rate in Hz. For an arbitrary continuous function $g(x,t)$ defined for $x \in \mathcal{D}_\mathcal{S}$ and $t \in \mathbb{R}^+$, g_l^n is a grid function approximating $g(lh,nk)$.

Let us introduce the forward and backward unit time and space shift operators, applied to g_l^n:

$$e_{t-} g_l^n = g_l^{n-1} \qquad e_{t+} g_l^n = g_l^{n+1} \tag{25a}$$

$$e_{x-} g_l^n = g_{l-1}^n \qquad e_{x+} g_l^n = g_{l+1}^n \tag{25b}$$

The partial differentiation with respect to time and space can be approximated with a number of first order FD operators:

$$\delta_{t-} = \frac{1 - e_{t-}}{k} \qquad \delta_{t+} = \frac{e_{t+} - 1}{k} \qquad \delta_{t\cdot} = \frac{e_{t+} - e_{t-}}{2k} \tag{26a}$$

$$\delta_{x-} = \frac{1 - e_{x-}}{h} \qquad \delta_{x+} = \frac{e_{x+} - 1}{h} \tag{26b}$$

Higher order partial derivation operators are approximated with:

$$\partial_t^2 \approx \delta_{tt} = \delta_{t-}\delta_{t+} \qquad \partial_x^2 \approx \delta_{xx} = \delta_{x-}\delta_{x+} \tag{27a}$$

$$\partial_x^4 \approx \delta_{xxxx} = \delta_{xx}\delta_{xx} \tag{27b}$$

Finally, the averaging FD operators approximate identity:

$$\mu_{t-} = \frac{1 + e_{t-}}{2} \qquad \mu_{t+} = \frac{e_{t+} + 1}{2} \qquad \mu_{t\cdot} = \frac{e_{t+} + e_{t-}}{2} \tag{28}$$

Note that $\delta_{t-}\mu_{t+} = \delta_{t+}\mu_{t-} = \delta_{t\cdot}$.

3.2. Vector-matrix notation

A number of grid functions are defined over $\mathfrak{d}_\mathcal{S}$. We can therefore describe the discrete position of the whole string with vectors. The simply supported boundary conditions ensure that the two extreme values are 0 at all times:

$$w_0^n = w_N^n = 0 \qquad u_0^N = u_N^n = 0 \tag{29a}$$

$$\delta_{xx} w_0^n = \delta_{xx} w_N^n = 0 \qquad \delta_{xx} u_0^n = \delta_{xx} u_N^n = 0 \tag{29b}$$

We now only need to store the state of the string in a vector of size $(N-1)$, omitting the two extreme values:

$$\mathbf{w}^n = [w_1^n, \dots, w_{N-1}^n]^T \qquad \mathbf{u}^n = [u_1^n, \dots, u_{N-1}^n]^T \tag{30}$$

The action of spatial FD operators on the grid functions is then equivalent to a matrix-vector multiplication. For simply supported boundary conditions, the notation of spatial FD operators in matrix form naturally follows as:

$$\mathbf{D}_{x-} = \frac{1}{h} \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ & \ddots & \ddots & & \\ & & & -1 & 1 \\ & & & & -1 \end{bmatrix} \qquad \begin{aligned} \mathbf{D}_{x+} &= -\mathbf{D}_{x-}^T \\ \mathbf{D}_{xx} &= \mathbf{D}_{x+}\mathbf{D}_{x-} \\ \mathbf{D}_{xxxx} &= \mathbf{D}_{xx}\mathbf{D}_{xx} \end{aligned} \tag{31}$$

of size $N \times (N-1)$, $(N-1) \times N$, $(N-1) \times (N-1)$, and $(N-1) \times (N-1)$, respectively.

3.3. Finite difference scheme

3.3.1. Vertical polarisation

We can now discretise Equation 1 as follows:

$$\mathsf{L}\mathbf{w}^n = \mu_{t\cdot} \mathbf{J}_\mathbf{w}^n \mathbf{f}_\mathbf{w}^n \tag{32}$$

where L is a matrix form discretisation of the partial derivation operator $\mathcal{L}$ defined in Equation 2:

$$\mathsf{L} = \rho\delta_{tt} - T\mathbf{D}_{xx} + EI_0\mathbf{D}_{xxxx} + \lambda_1\rho\delta_{t\cdot} - \lambda_2\rho\delta_{t-}\mathbf{D}_{xx} \tag{33}$$

$\mathbf{J}_\mathbf{w}^n$ is the $(N-1) \times (N+1)$ distribution matrix, and $\mathbf{f}_\mathbf{w}^n$ is a column vector containing all the contact force information:

$$\mathbf{J}_\mathbf{w}^n = \left[\begin{array}{c|c|c} \mathbf{I_{N-1}} & -\mathbf{j}_F^n & -\mathbf{j}_B^n \end{array} \right] \tag{34a}$$

$$\mathbf{f}_\mathbf{w}^n = \left[\begin{array}{c|c|c} (\mathbf{f}_N^n)^T & f_F^n & f_B^n \end{array} \right]^T \tag{34b}$$

where $\mathbf{I_{N-1}}$ is the $(N-1)\times(N-1)$ identity matrix, and $\mathbf{j}_F^n$ and $\mathbf{j}_B^n$ are discrete spreading operators in column vector form, accounting for the continuous distributions described in Section 2.1.2. $\mathbf{f}_N^n$, f_F^n and f_B^n are the discrete counterparts of those defined in Section 2.1. Energy conserving schemes for lumped collisions have been used for similar problems (in particular, the collision of a piano hammer with the string [24]); one in particular, was recently studied [25], that we adapt for the damped, distributed case [17]. We now have:

$$\mathbf{f}_\mathbf{w}^n = \frac{\delta_t.\mathbf{\Phi}^n}{\delta_t.\mathbf{\Delta}^n} + (\delta_t.\mathbf{\Delta}^n)\odot\mathbf{\Psi}^n \tag{35}$$

where the division is pointwise, and $\odot$ is the pointwise product.

$\mathbf{\Phi}^n(\mathbf{\Delta}^n)$, $\mathbf{\Psi}^n(\mathbf{\Delta}^n)$ are function of the vector penetration $\mathbf{\Delta}^n$:

$$\mathbf{\Phi}^n(\mathbf{\Delta}^n) = \frac{\mathbf{K}}{\boldsymbol{\alpha}+1}\odot[\mathbf{\Delta}^n]_+^{\boldsymbol{\alpha}+1} \quad \mathbf{\Psi}^n(\mathbf{\Delta}^n) = \mathbf{K}\odot\boldsymbol{\beta}\odot[\mathbf{\Delta}^n]_+^{\boldsymbol{\alpha}} \tag{36}$$

where the exponentiation operation is also element-wise. $\mathbf{\Delta}^n$, $\mathbf{K}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$ are now in vector form:

$$\mathbf{K} = \begin{bmatrix} \vdots \\ K_N \\ \vdots \\ \hline K_F \\ \hline K_B \end{bmatrix} \quad \boldsymbol{\alpha} = \begin{bmatrix} \vdots \\ \alpha_N \\ \vdots \\ \hline \alpha_F \\ \hline \alpha_B \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} \vdots \\ \beta_N \\ \vdots \\ \hline \beta_F \\ \hline \beta_B \end{bmatrix} \tag{37a}$$

$$\mathbf{\Delta}^n = \begin{bmatrix} \vdots \\ \mathbf{\Delta}_N^n \\ \vdots \\ \hline \Delta_F^n \\ \hline \Delta_B^n \end{bmatrix} \quad \begin{aligned} \mathbf{\Delta}_N^n &= \varepsilon - \mathbf{w}^n \\ \Delta_F^n &= h\mathbf{j}_F^{n\,T}\mathbf{w}^n - w_F^n \\ \Delta_B^n &= h\mathbf{j}_B^{n\,T}\mathbf{w}^n - w_B^n \end{aligned} \tag{37b}$$

where $\varepsilon_l = \varepsilon(lh)$, and w_F^n and w_B^n are the respective vertical positions of the finger and bow, governed by:

$$\mathbf{M}_{FB}\delta_{tt}\mathbf{w}_{FB}^n = \mathbf{f}_{\mathbf{w}FB}^n + \mathbf{f}_{\mathbf{ext}w,FB}^n \qquad \mathbf{M}_{FB} = \begin{bmatrix} M_F & 0 \\ 0 & M_B \end{bmatrix} \tag{38a}$$

$$\mathbf{w}_{FB}^n = \begin{bmatrix} w_F^n \\ w_B^n \end{bmatrix} \qquad \mathbf{f}_{\mathbf{w}FB}^n = \begin{bmatrix} f_F^n \\ f_B^n \end{bmatrix} \qquad \mathbf{f}_{\mathbf{ext}w,FB}^n = \begin{bmatrix} f_{\mathrm{ext}w,F}^n \\ f_{\mathrm{ext}w,B}^n \end{bmatrix} \tag{38b}$$

3.3.2. Horizontal polarisation

Equation 9 is now discretised as:

$$\mathsf{L}\mathbf{u}^n = -\mu_t.\mathbf{J}_\mathbf{u}^n\mathbf{f}_\mathbf{u}^n \tag{39a}$$

$$\mathbf{J}_\mathbf{u}^n = \left[\ \mathbf{I_{N-1}}\ \middle|\ \mathbf{j}_F^n\ \middle|\ \mathbf{j}_B^n\ \right] \tag{39b}$$

L is defined in Equation 33. $\mathbf{f}_\mathbf{u}^n$ is a column vector containing the friction force information:

$$\mathbf{f}_\mathbf{u}^n = \begin{bmatrix} \vdots \\ [\mathbf{f}_N^n]_+\odot\varphi_N\left(\mathbf{v}_{\mathrm{rel},N}^n\right) \\ \vdots \\ \hline [f_F^n]_+\varphi_F\left(v_{\mathrm{rel},F}^n\right) \\ \hline [f_B^n]_+\varphi_B\left(v_{\mathrm{rel},B}^n\right) \end{bmatrix} \quad \begin{aligned} \mathbf{v}_{\mathrm{rel},N}^n &= \delta_t.\mathbf{u}^n \\ v_{\mathrm{rel},F}^n &= h\delta_t.\left(\mathbf{j}_F^{n\,T}\mathbf{u}^n\right) - \delta_t.u_F^n \\ v_{\mathrm{rel},B}^n &= h\delta_t.\left(\mathbf{j}_B^{n\,T}\mathbf{u}^n\right) - \delta_t.u_B^n \end{aligned} \tag{40}$$

where φ_N, φ_F and φ_B are defined in section 2.2. We can define a vector relative velocity:

$$\mathbf{v}_\mathrm{rel}^n = \left[\left(\mathbf{v}_{\mathrm{rel},N}^n\right)^T | v_{\mathrm{rel},F}^n | v_{\mathrm{rel},B}^n\right]^T \tag{41}$$

Finally, a matrix equation describes the evolution of the horizontal displacements u_F^n and u_B^n of the finger and bow, respectively:

$$\mathbf{M}_{FB}\delta_{tt}\mathbf{u}_{FB}^n = \mathbf{K}_{FB}\mu_t.\mathbf{u}_{FB}^n - \boldsymbol{\lambda}_{FB}\delta_t.\mathbf{u}_{FB}^n + \mathbf{f}_{\mathbf{u}FB}^n + \mathbf{f}_{\mathbf{ext}u,FB}^n \tag{42a}$$

$$\mathbf{K}_{FB} = \begin{bmatrix} K_F & 0 \\ 0 & 0 \end{bmatrix} \quad \boldsymbol{\lambda}_{FB} = \begin{bmatrix} \lambda_F & 0 \\ 0 & \lambda_B \end{bmatrix} \quad \mathbf{u}_{FB}^n = \begin{bmatrix} u_F^n \\ u_B^n \end{bmatrix} \tag{42b}$$

$$\mathbf{f}_{\mathbf{u}FB}^n = \begin{bmatrix} [f_F^n]_+\varphi_F\left(v_{\mathrm{rel},F}^n\right) \\ [f_B^n]_+\varphi_B\left(v_{\mathrm{rel},B}^n\right) \end{bmatrix} \quad \mathbf{f}_{\mathbf{ext}u,FB}^n = \begin{bmatrix} 0 \\ f_{\mathrm{ext}u,B}^n \end{bmatrix} \tag{42c}$$

3.4. Energy analysis

We can transfer the results of Section 2.3 to discrete time, and monitor the energy exchanges going on in the system at all times during the simulation. We derive an energy balance equation between the energy of the closed system (H^n) and the power brought in and out, by external excitation (P^n) and damping Q^n. Conservation of this *total* energy helps ensuring a stable algorithm.

3.4.1. Vertical polarisation

For the vertical polarisation, multiplying Equation 32 by $h(\delta_t.\mathbf{w}^n)^T$ and Equation 38a by $(\delta_t.\mathbf{w}_{FB}^n)^T$ gives the power balance:

$$\delta_t - \mathrm{H}_w^n = \mathrm{P}_w^n - \mathrm{Q}_w^n \tag{43}$$

The numerical energy H_w^n is defined as:

$$\mathrm{H}_w^n = \mathrm{H}_{w,s}^n + \mathrm{H}_\Phi^n \tag{44a}$$

$$\begin{aligned} \mathrm{H}_{w,s}^n = {} & \frac{\rho h}{2}|\delta_{t+}\mathbf{w}^n|^2 + \frac{Th}{2}(\mathbf{D}_{x-}\mathbf{w}^n)^T\mathbf{D}_{x-}\mathbf{w}^{n+1} \\ & + \frac{EI_0h}{2}(\mathbf{D}_{xx}\mathbf{w}^n)^T\mathbf{D}_{xx}\mathbf{w}^{n+1} - \frac{\lambda_2\rho kh}{4}|\delta_{t+}\mathbf{D}_{x-}\mathbf{w}^n|^2 \end{aligned} \tag{44b}$$

$$\mathrm{H}_\Phi^n = \mathbf{h}^T\mu_{t+}\mathbf{\Phi}^n + \frac{1}{2}(\mathbf{M}_{FB}\delta_{t+}\mathbf{w}_{FB}^n)^T\delta_{t+}\mathbf{w}_{FB}^n \tag{44c}$$

where $\mathbf{h} = [...h...|1|1]^T$.

The power P_w^n supplied or withdrawn through excitation is:

$$\mathrm{P}_w^n = (\delta_t.\mathbf{w}_{FB}^n)^T\mathbf{f}_{\mathrm{ext}w,FB}^n - h\Big((\mu_t.\mathbf{w}^n)^T(\delta_t.\mathbf{J}^n)\Big)\mathbf{f}_\mathbf{w}^n \tag{45}$$

The power $\mathrm{Q}_w^n \geqslant 0$ dissipated through damping is:

$$\mathrm{Q}_w^n = \mathrm{Q}_{w,s}^n + \mathrm{Q}_\Psi^n \tag{46a}$$

$$\mathrm{Q}_{w,s}^n = \lambda_1\rho h|\delta_t.\mathbf{w}^n|^2 + \lambda_2\rho h|\delta_t.\mathbf{D}_{x-}\mathbf{w}^n|^2 \tag{46b}$$

$$\mathrm{Q}_\Psi^n = (\mathbf{h}\odot\delta_t.\mathbf{\Delta}^n)^T((\delta_t.\mathbf{\Delta}^n)\odot\mathbf{\Psi}^n) \tag{46c}$$

In the absence of external excitation, the numerical energy H_w^n is strictly decreasing. The stability of this scheme then boils down to H_w^n being non-negative at all times. As $\mathrm{H}_\Phi^n \geqslant 0$ by construction, this is then equivalent to $\mathrm{H}_{w,s}^n \geqslant 0$, which is verified under the condition linking the time step k and grid spacing h [14]:

$$h \geqslant \sqrt{\frac{1}{2}\left(\frac{Tk^2}{\rho} + 2\lambda_2 k + \sqrt{\left(\frac{Tk^2}{\rho} + 2\lambda_2 k\right)^2 + 16k^2\frac{EI_0}{\rho}}\right)} \tag{47}$$

3.4.2. Horizontal polarisation

On the other hand, the product of Equation 36 by $h(\delta_{t\cdot}\mathbf{u}^n)^T$, and that of Equation 42a by $(\delta_{t\cdot}\mathbf{u}_{FB}^n)^T$, yields a numerical power balance for the horizontal polarisation:

$$\delta_{t-}\mathrm{H}_u^n = \mathrm{P}_u^n - \mathrm{Q}_u^n \tag{48}$$

where the numerical energy H_u^n is defined as:

$$\mathrm{H}_u^n = \mathrm{H}_{u,s}^n + \mathrm{H}_{u,FB}^n \tag{49a}$$

$$\mathrm{H}_{u,s}^n = \frac{\rho h}{2}|\delta_{t+}\mathbf{u}^n|^2 + \frac{Th}{2}(\mathbf{D}_{x-}\mathbf{u}^n)^T\mathbf{D}_{x-}\mathbf{u}^{n+1} + \frac{EI_0h}{2}(\mathbf{D}_{xx}\mathbf{u}^n)^T\mathbf{D}_{xx}\mathbf{u}^{n+1} - \frac{\lambda_2\rho k h}{4}|\delta_{t+}\mathbf{D}_{x-}\mathbf{u}^n|^2 \tag{49b}$$

$$\mathrm{H}_{u,FB}^n = \frac{1}{2}(\mathbf{M}_{FB}\delta_{t+}\mathbf{u}_{FB}^n)^T\delta_{t+}\mathbf{u}_{FB}^n + \frac{1}{2}\mu_{t+}\Big((\mathbf{K}_{FB}\mathbf{u}_{FB}^n)^T\mathbf{u}_{FB}^n\Big) \tag{49c}$$

The power P_u^n brought in or out by the excitation is:

$$\mathrm{P}_u^n = h\Big((\mu_{t\cdot}\mathbf{u}^n)^T(\delta_{t\cdot}\mathbf{J}^n)\Big)\mathbf{f}_{\mathbf{u}}^n + (\delta_{t\cdot}\mathbf{u}_{FB}^n)^T\mathbf{f}_{\mathbf{ext}u,FB}^n \tag{50}$$

The power $\mathrm{Q}_u^t \geqslant 0$ dissipated by friction and damping is:

$$\mathrm{Q}_u^n = \mathrm{Q}_{u,s}^n + \mathrm{Q}_\varphi^n + \mathrm{Q}_{u,FB}^n \tag{51a}$$

$$\mathrm{Q}_{u,s}^n = \lambda_1\rho h|\delta_{t\cdot}\mathbf{u}^n|^2 + \lambda_2\rho h|\delta_{t\cdot}\mathbf{D}_{x-}\mathbf{u}^n|^2 \tag{51b}$$

$$\mathrm{Q}_\varphi^n = (\mathbf{h}\odot\mathbf{v}_{\mathrm{rel}}^n)^T\mathbf{f}_{\mathbf{u}}^n \tag{51c}$$

$$\mathrm{Q}_{u,FB}^n = (\lambda_{FB}\delta_{t\cdot}\mathbf{u}_{FB})^T\delta_{t\cdot}\mathbf{u}_{FB} \tag{51d}$$

The stability condition 47 straightforwardly holds for this scheme; indeed, choosing to use the 2-point averaging operator in Equation 42a does not introduce any stricter bound on h, as the energy $\mathrm{H}_{u,FB}^n$ of the finger and bow is always strictly positive.

3.4.3. Total energy

The total numerical energy H^n of the system is balanced by:

$$\delta_{t-}\mathrm{H}^n = \mathrm{P}^n - \mathrm{Q}^n \tag{52a}$$

$$\mathrm{H}^n = \mathrm{H}_u^n + \mathrm{H}_w^n \qquad \mathrm{P}^n = \mathrm{P}_u^n + \mathrm{P}_w^n \qquad \mathrm{Q}^n = \mathrm{Q}_u^n + \mathrm{Q}_w^n \tag{52b}$$

We can therefore monitor the quantity E^n, that should remain constant (to machine accuracy) throughout the simulation:

$$\mathrm{E}^n = \mathrm{H}^n - k\sum_{i=0}^{n}\Big(\mathrm{P}^i - \mathrm{Q}^i\Big) = \mathrm{H}^0 \tag{53}$$

3.5. Scheme update

3.5.1. Vertical polarisation

Expanding the operators in Equations 32 and 38a, and combining 35 and 36, leads to a two-step recursion algorithm in vector-matrix form, to be updated at each time step n:

$$\mathbf{w}^{n+1} = \mathbf{B}\mathbf{w}^n + \mathbf{C}\mathbf{w}^{n-1} + A\mu_{t\cdot}\mathbf{J}_{\mathbf{w}}^n\mathbf{f}_{\mathbf{w}}^n \tag{54a}$$

$$\mathbf{w}_{FB}^{n+1} = 2\mathbf{w}_{FB}^n - \mathbf{w}_{FB}^{n-1} + k^2\mathbf{M}_{FB}^{-1}\big(\mathbf{f}_{\mathbf{w}FB}^n + \mathbf{f}_{\mathbf{ext}w,FB}^n\big) \tag{54b}$$

$$A = \frac{2k^2}{\rho(2+\lambda_1 k)} \tag{54c}$$

$$\mathbf{B} = \frac{2}{2+\lambda_1 k}\left(2 + \left(\frac{Tk^2}{\rho} + \lambda_2 k\right)\mathbf{D}_{xx} - \frac{EI_0k^2}{\rho}\mathbf{D}_{xxxx}\right) \tag{54d}$$

$$\mathbf{C} = \frac{2}{2+\lambda_1 k}\left(\frac{\lambda_1 k}{2} - 1 - \lambda_2 k\mathbf{D}_{xx}\right) \tag{54e}$$

However, the nonlinearity of the contact model doesn't allow for a simple explicit update. Combining Equations 54a and 54b, and rewriting in terms of $\mathbf{\Delta}^n$, leads to a nonlinear equation in matrix form, in terms of the unknown vector $\mathbf{r}^n = \mathbf{\Delta}^{n+1} - \mathbf{\Delta}^{n-1}$:

$$\mathbf{\Lambda}_1^n\mathbf{r}^n + \mathbf{\Lambda_2}^n\mathbf{f}_\Phi^n + \mathbf{b}_{\mathbf{w}}^n = 0 \tag{55}$$

where the matrices $\mathbf{\Lambda}_1^n$, $\mathbf{\Lambda}_2^n$, and the vectors $\mathbf{f}_\Phi^n$, $\mathbf{b}^n$ are given by:

$$\mathbf{\Lambda}_2^n = A\mathrm{diag}(\overline{\mathbf{h}})\big(\mathbf{J}_{\mathbf{w}}^{n+1}\big)^T\mu_{t\cdot}\mathbf{J}_{\mathbf{w}}^n + k^2\mathbf{M}_{\mathrm{inv}} \tag{56a}$$

$$\mathbf{\Lambda}_1^n = \mathbf{I_{N+1}} + \frac{1}{2k}\mathbf{\Lambda}_2^n\mathrm{diag}(\mathbf{\Psi}^n) \tag{56b}$$

$$\mathbf{f}_\Phi^n = \frac{\delta_{t\cdot}\mathbf{\Phi}^n}{\delta_{t\cdot}\mathbf{\Delta}^n} = \frac{\mathbf{\Phi}(\mathbf{r}^n + \mathbf{\Delta}^{n-1}) - \mathbf{\Phi}(\mathbf{\Delta}^{n-1})}{\mathbf{r}^n} \tag{56c}$$

$$\mathbf{b}_{\mathbf{w}}^n = \Big[\mathbf{0_{N-1}}|2\big(\mathbf{w}_{FB}^n - \mathbf{w}_{FB}^{n-1}\big)^T + k^2\big(\mathbf{M}_{FB}^{-1}\mathbf{f}_{\mathbf{ext},\mathrm{FB}}^n\big)^T\Big]^T + \mathrm{diag}\big(\overline{\mathbf{h}}\big)\Big(\big(\mathbf{J}_{\mathbf{w}}^{n+1}\big)^T\big(\mathbf{B}\mathbf{w}^n + \mathbf{C}\mathbf{w}^{n-1}\big) - \big(\mathbf{J}_{\mathbf{w}}^{n-1}\big)^T\mathbf{w}^{n-1}\Big) \tag{56d}$$

where $\mathbf{M}_{\mathrm{inv}}$ is a $(N+1)\times(N+1)$ matrix with $\mathbf{M}_{FB}^{-1}$ at its bottom-right corner, and all zeros elsewhere; $\mathbf{0_{N-1}}$ is an all-zero row vector of length $(N-1)$; and $\overline{\mathbf{h}} = [...1...|h|h]^T$.

Equation 55 is resolved with an iterative nonlinear system solver.

3.5.2. Horizontal polarisation

Similarly to the vertical polarisation, a two-step recursion is derived from Schemes 39a and 42a:

$$\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1} - A\mu_{t\cdot}\mathbf{J}_{\mathbf{u}}^n\mathbf{f}_{\mathbf{u}}^n \tag{57a}$$

$$\mathbf{u}_{FB}^{n+1} = \mathbf{B}_{FB}\mathbf{u}_{FB}^n + \mathbf{C}_{FB}^{n-1} + k^2\mathbf{M}_{FB}^{-1}\mathbf{A}_{FB}\big(\mathbf{f}_{\mathbf{u}FB}^n + \mathbf{f}_{\mathbf{ext}u,FB}^n\big) \tag{57b}$$

$$\mathbf{A}_{FB} = 2\big(2\mathbf{M}_{FB} + k^2\mathbf{K}_{FB} + k\mathbf{\lambda}_{FB}\big)^{-1}\mathbf{M}_{FB} \tag{57c}$$

$$\mathbf{B}_{FB} = 2\mathbf{A}_{FB} \tag{57d}$$

$$\mathbf{C}_{FB} = \frac{1}{2}\mathbf{A}_{FB}\big(-2\mathbf{M}_{FB} - k^2\mathbf{K}_{FB} + k\mathbf{\lambda}_{FB}\big)\mathbf{M}_{FB}^{-1} \tag{57e}$$

A, $\mathbf{B}$, and $\mathbf{C}$ are defined in 54. We can write Equations 57a and 57b in terms of $\mathbf{v}_{\mathrm{rel}}^n$:

$$\mathbf{v}_{\mathrm{rel}}^n + \mathbf{\Lambda}_3^n\mathbf{f}_{\mathbf{u}}^n + \mathbf{b}_{\mathbf{u}}^n = 0 \tag{58}$$

where the matrix $\mathbf{\Lambda}_3^n$, and the vector $\mathbf{b}_{\mathbf{u}}^n$ are defined as:

$$\mathbf{\Lambda}_3^n = \frac{1}{2k}\left(A\text{diag}\left(\overline{\mathbf{h}}\right)\left(\mathbf{J}_\mathbf{u}^{n+1}\right)^T \mu_t . \mathbf{J}_\mathbf{u}^n + \mathbf{A}_{\text{obj}}\right) \tag{59a}$$

$$\mathbf{b}_\mathbf{u}^n = \frac{1}{2k}\text{diag}\left(\overline{\mathbf{h}}\right)\left(\left(\mathbf{J}_\mathbf{u}^{n-1}\right)^T \mathbf{u}^{n-1} - \left(\mathbf{J}_\mathbf{u}^{n+1}\right)^T \left(\mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1}\right)\right) + \left[\mathbf{0_{N-1}} | \left(\mathbf{b}_{\mathbf{u}FB}^n\right)^T\right]^T \tag{59b}$$

$$\mathbf{b}_{\mathbf{u}FB}^n = \frac{1}{2k}\left(\mathbf{B}_{FB}\mathbf{u}_{FB}^n + (\mathbf{C}_{FB} - \mathbf{I_2})\mathbf{u}_{FB}^{n-1} + \mathbf{A}_{FB}\mathbf{f}_{\mathbf{ext}u,FB}^n\right) \tag{59c}$$

where $\mathbf{A}_{\text{obj}}$ is a $(N+1)\times(N+1)$ matrix with $\mathbf{A}_{FB}$ at its bottom-right corner and zeros elsewhere; $\mathbf{I_2}$ is the 2×2 identity matrix.

4. SIMULATION RESULTS

4.1. Control parameters

Simulations are run at audio sample rate (F_s = 44.1 kHz). The user controls the physical parameters of the string and all three objects. A table of measured string parameters on violins, violas and cellos (from Percival [26]) is readily available as a preset. The gestural control is achieved with breakpoint functions for the bow position, force applied normally and tangentially, and the finger position and normal force. The output waveform is read as the displacement of the last mobile point of the string before the bridge termination.

A video demonstrating a typical gesture, generated from simulated data from this model, is available on the companion website [1].

4.2. Bowed string motion

As the bow is driven by an external force, and not an imposed velocity, the amplitude and shape of the force signal to send into the bow is at first less intuitive to gauge. However, while a full parameter exploration study is definitely worth considering (with regards to playability and transient quality; see e.g. [27]), minimal trial and error allowed us to successfully reproduce the standard, periodic Helmholtz motion of the bowed string, as well as other typical oscillation states under realistic bowing conditions. Schelleng [28] described theoretical bow force limits, for a given bow position and velocity, beyond which the player presses the bow either too strongly for the returning Helmholtz corner to detach it from the string (*raucous motion*), or too lightly for the string to stick to it for a whole nominal period (*multiple slipping*). Figure 3 shows the typical sawtooth waveform associated with the Helmholtz motion, the split sawtooth associated with multiple slipping, and the rough, aperiodic waveform resulting from raucous motion of the string.

4.3. Gesture reproduction

The inclusion of the left hand finger and neck, as well as the dynamics of the bow, allow to simulate a broad range of typical bowed string gestures. The bow can move along and bounce against the string; the fingers sliding along the fingerboard or oscillating around a central position create glissando and vibrato sounds. "Plucking" the string with a half raised cosine function in both polarisations leads to pizzicato sounds, and even slap double bass, if the string is plucked hard enough to bounce and rub against the fingerboard.

4.4. Energy balance

To demonstrate the balanced numerical energy of the system, we monitor the variations of the quantity E^n defined in Equation 53 along a bowed string simulation, where the bow and finger positions, forces, and the bow tangential force are all time-varying. We normalise E^n with respect to the mean energy $\bar{\mathrm{H}}^n$, averaged over the duration of the simulation. As seen in Figure 4 (bottom), E^n is invariant until the 10th significant digit. The finite error tolerance for the nonlinear system solvers, as well as the accumulation of round-off error, seem to prevent reaching true floating point accuracy.

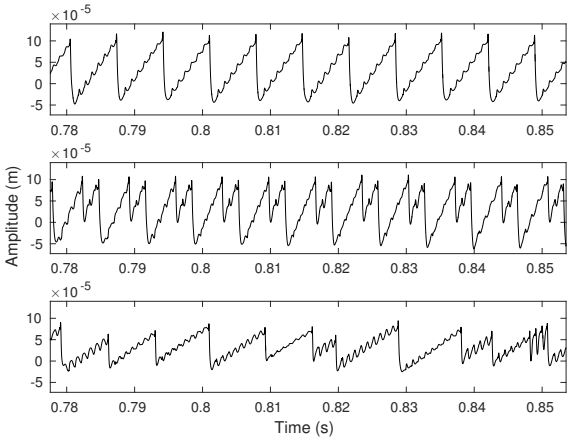


Figure 3: Different simulated waveforms on a cello D string, with fixed bow position $x_B = 0.851L$ m, bow force $f_{\text{ext}w,B} = -2$ N, and bow tangential force $f_{\text{ext}u,B} = 4.4$ N (Helmholtz motion, top), $f_{\text{ext}u,B} = 6.8$ N (multiple slipping, middle), and $f_{\text{ext}u,B} = 2.5$ N (raucous motion, bottom).

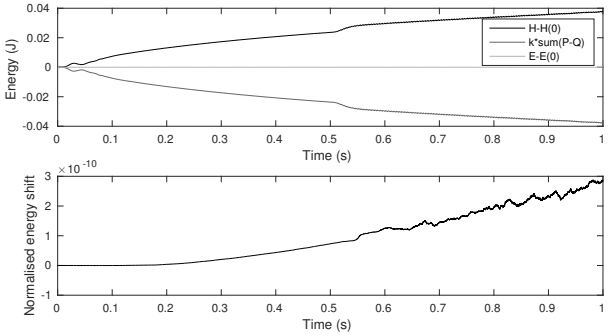


Figure 4: Numerical energy balance for the whole system. Top: in both polarisations, the energy is balanced at all times by the cumulative supplied and withdrawn power. Bottom: the total energy is conserved to the 10th significant digit, when normalised with respect to the mean energy. The apparent trend is due to accumulated round-off error.

5. CONCLUSIONS

This work introduces a novel two polarisation bowed string physical model, including nonlinear damped contact and friction interactions with one bow, one stopping finger, and the distributed fingerboard. An energy-balanced finite difference scheme was presented, resulting in a two-step time recursion. The scheme implementation takes great advantage not only of the structure of Equations 55 and 58, but also of the shape of the nonlinear force term, to optimise computations. In particular, and for this choice of friction curves, the string stopping part of the friction interaction (finger and neck) can, in most realistic playing cases, be decoupled from the highly nonlinear bow part, and solved separately. On the bow side, the use of Friedlander's construction [29] ensures a well-behaved root finding, even trivial during the *sticking* phases of each cycle. Friedlander's

hypothesis, confirmed later experimentally for this type of friction model [23], allows the deterministic resolution of the likely case where the (decoupled) bow part of Equation 58 has not one graphical solution, but three; as a result, a hysteretic cycle arises, leading to pitch flattening. This effect has indeed later been found to be due to the naturally hysteretic thermal behaviour of the melting rosin, indeed well approximated by the simpler friction curve models.

The inclusion of lumped and distributed interactions with the player and fingerboard allows for simulating full articulated gestures in a relatively instinctive and concrete way, without having to rely on somewhat abstract hypotheses — an eloquent example being the finger model, that accounts for several important phenomena that would be difficult (impossible in fact, for some) to model with a simple absorbing string termination. Here, the simple action of pushing a finger down onto the string results in damped dynamic behaviour in both polarisations, variations of the string's speaking length, possible slipping of the string while captured, while the portion of the string between the nut and finger is still realistically oscillating, and responding to the excitation.

However, an important aspect of gestural control in bowed string playing resides in real-time adjustments of playing parameters during note production. The musician relies on immediate feedback from his instrument, adapting its playing accordingly. Our model, even with the aforementioned possible optimisations, does not run in real-time, making gesture design rather difficult. An interesting study could make use of recorded data from sensors during various gestures, feeding them as time series into the model, rather than our current breakpoint functions. This would help calibrate the model, on the string side as well as for the gestural functions [11, 30].

The adaptation of this work to the more realistic case of multiple fingers (and, why not, multiple bows) is trivial, as well as the design of a multiple string environment. The mutual coupling of such strings is the obvious next step, moving towards the design of a full instrument, where strings communicate with a flexible body and with each other through a bridge. The simulated body will eventually take a great part in both the virtual instrument's playability, introducing vibrations feeding back into the strings, and the realism of the synthetic sound; to address the latter, and get a glimpse at the potential of a full instrument model, we have convolved a dry output signal from this string model with the impulse response of a cello body, a principle that is still used to this day for high quality sound synthesis [31]. The resulting sound example can be found online, amongst other relevant samples obtained from the model [1].

6. REFERENCES

[1] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part 1," *J. Audio Eng. Soc.*, vol. 19, pp. 462–470, June 1971.

[2] R. A. Bacon and J. M. Bowsher, "A discrete model of a struck string," *Acta Acust. united Ac.*, vol. 41, no. 1, pp. 21–27, 1978.

[3] K. Karplus and A. Strong, "Digital synthesis of plucked-string and drum timbres," *Comput. Music J.*, vol. 7, no. 2, pp. 43–55, 1983.

[4] J. O. Smith III, "A new approach to digital reverberation using closed waveguide networks," in *Proc. Int. Computer Music Conf.*, (Vancouver, Canada), pp. 47–53, 1985.

[5] M. Karjalainen, V. Välimäki, and T. Tolonen, "Plucked-string models: from the Karplus-Strong algorithm to digital waveguides and beyond," *Comput. Music J.*, vol. 22, no. 3, pp. 17–32, 1998.

[6] J. Woodhouse, "Physical modeling of bowed strings," *Comput. Music J.*, vol. 16, no. 4, pp. 43–56, 1992.

[7] T. Takala, J. Hiipakka, M. Laurson, and V. Välimäki, "An expressive synthesis model for bowed string instruments," in *Proc. Int. Computer Music Conf.*, (Berlin, Germany), 2000.

[8] S. Serafin, F. Avanzini, D. Ing, and D. Rocchesso, "Bowed string simulation using an elasto-plastic friction model," in *Proc. Stockholm Mus. Acoust. Conf.*, (Stockholm, Sweden), pp. 1–4, 2003.

[9] J. Woodhouse, "Bowed string simulation using a thermal friction model," *Acta Acust. united Ac.*, vol. 89, no. 2, pp. 355–368, 2003.

[10] E. Maestre, C. Spa, and J. O. Smith III, "A bowed string physical model including finite-width thermal friction and hair dynamics," in *Proc. Int. Computer Music Conf.*, (Athens, Greece), 2014.

[11] M. Demoucron, *On the control of virtual violins—Physical modelling and control of bowed string instruments*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2008.

[12] V. Debut, X. Delaune, and J. Antunes, "Identification of the nonlinear excitation force acting on a bowed string using the dynamical responses at remote locations," *Int. J. Mech. Sci.*, vol. 52, no. 11, pp. 1419–1436, 2010.

[13] J. C. Strikwerda, *Finite difference schemes and partial differential equations*. Siam, 2004.

[14] S. Bilbao, *Numerical sound synthesis*. Chichester, UK: John Wiley & Sons, Ltd, Oct. 2009.

[15] A. Chaigne and A. Askenfelt, "Numerical simulations of piano strings. I. A physical model for a struck string using finite difference methods," *J. Acoust. Soc. Am.*, vol. 95, no. 2, pp. 1112–1118, 1994.

[16] J. Bensa, S. Bilbao, R. Kronland-Martinet, and J. O. Smith III, "The simulation of piano string vibration: From physical models to finite difference schemes and digital waveguides," *J. Acoust. Soc. Am.*, vol. 114, no. 2, pp. 1095–1107, 2003.

[17] S. Bilbao, A. Torin, and V. Chatziioannou, "Numerical modeling of collisions in musical instruments," *Acta Acust. united Ac.*, vol. 101, pp. 155–173, Jan. 2015.

[18] V. Chatziioannou and M. Van Walstijn, "Energy conserving schemes for the simulation of musical instrument contact dynamics," *J. Sound Vib.*, vol. 339, pp. 262–279, 2015.

[19] C. Desvages and S. Bilbao, "Physical modeling of nonlinear player-string interactions in bowed string sound synthesis using finite difference methods," in *Proc. Int. Symp. Mus. Acoust.*, (Le Mans, France), 2014.

[20] S. Bilbao and A. Torin, "Numerical simulation of string/barrier collisions: the fretboard," in *Proc. Int. Conf. Digital Audio Effects*, (Erlangen, Germany), 2014.

[21] M. E. McIntyre and J. Woodhouse, "On the fundamentals of bowed-string dynamics," *Acta Acust. united Ac.*, vol. 43, no. 2, pp. 93–108, 1979.

[22] K. H. Hunt and F. R. E. Crossley, "Coefficient of restitution interpreted as damping in vibroimpact," *J. Appl. Mech.*, vol. 42, no. 2, pp. 440–445, 1975.

[23] J. H. Smith and J. Woodhouse, "The tribology of rosin," *J. Mech. Phys. Solids*, vol. 48, pp. 1633–1681, 2000.

[24] J. Chabassier, *Modeling and numerical simulation of a piano.* Phd thesis, Ecole Polytechnique X, Mar. 2012.

[25] V. Chatziioannou and M. van Walstijn, "An energy conserving finite difference scheme for simulation of collisions," in *Proc. Sound Music Computing Conf.*, (Stockholm, Sweden), pp. 584–591, 2013.

[26] G. K. Percival, *Physical modelling meets machine learning: performing music with a virtual string ensemble*. PhD thesis, University of Glasgow, 2013.

[27] R. T. Schumacher and J. Woodhouse, "The transient behaviour of models of bowed-string motion.," *Chaos (Woodbury, N.Y.)*, vol. 5, no. 3, pp. 509–523, 1995.

[28] J. C. Schelleng, "The bowed string and the player," *J. Acoust. Soc. Am.*, vol. 53, no. 1, pp. 26–41, 1973.

[29] F. G. Friedlander, "On the oscillations of a bowed string," *Math. Proc. Cambridge Phil. Soc.*, vol. 49, pp. 516–530, Oct. 1953.

[30] E. Maestre, "Analysis/synthesis of bowing control applied to violin sound rendering via physical models," in *Proc. Meet. Acoust.*, vol. 19, (Montreal, Canada), p. 035016, 2013.

[31] A. Pérez Carrillo, J. Bonada, J. Patynen, and V. Välimäki, "Method for measuring violin sound radiation based on bowed glissandi and its application to sound synthesis.," *J. Acoust. Soc. Am.*, vol. 130, pp. 1020–9, Aug. 2011.

HARMONIZING EFFECT USING SHORT-TIME TIME-REVERSAL

Hyung-Suk Kim
CCRMA,
Stanford University
Stanford, CA, USA
hskim08@ccrma.stanford.edu

Julius O. Smith
CCRMA,
Stanford University
Stanford, CA, USA
jos@ccrma.stanford.edu

ABSTRACT

A prior study of short-time time-reversal showed sideband modulation occurs for short time durations, creating overtones for single sinusoid signals. In this paper, we examine the overtones created by short-time time-reversal and the tonal relation between the overtones and the input signal. We present three methods of using short-time time-reversal for harmonizing audio signals. Then modifications to the previous short-time time-reversal needed to implement the proposed methods are described.

1. INTRODUCTION

In a previous paper [1], the general analysis of short-time time-reversal (STTR) was covered. It was shown that for very short window lengths, less than 30 ms, STTR has a sideband modulation effect, a form of spectral aliasing, that creates overtones for single sinusoidal signals and the possibility of exploiting this property for use as a harmonizing effect was briefly mentioned. In this paper, we will examine the overtones created by STTR on sinusoidal signals and propose methods for using STTR as a harmonizing effect.

Harmonizing effects, an application of pitch-shifting, is a widely used audio effect [2] popularized by the Eventide Harmonizer series. Previous implementations of harmonizing effects include delay-line modulation [3, 4, 5] and phase-vocoders [6, 7].

STTR can be categorized as a delay-line modulation. However, it is not a direct application of pitch shifting. The overtones occur naturally from a form of sideband modulation that arises by STTR. Because of this, a harmonizing effect using STTR can be implemented very efficiently. For 50% overlap-add STTR, only a single delay line per channel and two read pointers are needed [1].

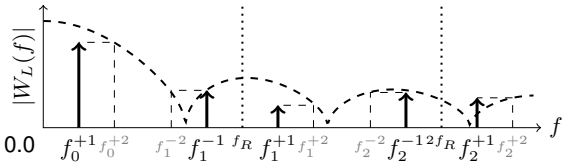
2. SHORT-TIME TIME-REVERSAL

STTR is a linear time-variant filter where the audio signal is windowed and the signal under the window is time reversed in place. In this section, we review the STTR equations and specify the parameters that will be used in the following sections.

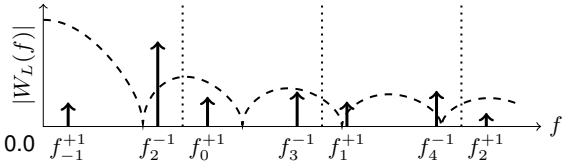
2.1. The STTR Equations

In [1], the equations for STTR in both the time-domain and the frequency-domain were derived. For input signal $x(t)$, the output of STTR $y(t)$ and its frequency response $Y(f)$ is as below, where $w_L(t)$ is the windowing function of length L used for time reversal and R is the step size used for overlap-add.

$$y(t) = \sum_{m=-\infty}^{\infty} x(-t + 2mR) w_L(t - mR) \tag{1}$$



(a) $f_0 < \frac{f_R}{2}$



(b) $f_0 > f_R$

Figure 1: A visualization of equation (4), the STTR output for a single sinusoid. The dotted vertical lines mark multiples of f_R. 1a shows the simple case where $f_0 < \frac{f_R}{2}$. The amplitude of the impulse at $f_k^{\pm 1}$ is found by sampling the window function $|W_L(f)|$ at $f_k^{\pm 2}$. 1b shows the generic case where $f_0 > f_R$. In such cases, the fundamental frequency, $f_0 = f_0^{+1}$, might not have the greatest amplitude, which can be used to harmonize the original signal.

$$Y(f) = \frac{1}{R} \sum_{k=-\infty}^{\infty} X\left(f - \frac{k}{R}\right) W_L\left(2f - \frac{k}{R}\right) \tag{2}$$

As an added constraint to preserve signal power, $w_L(t)$ and R must satisfy constant overlap-add.

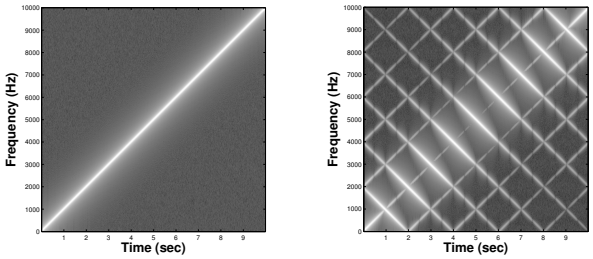
$$\sum_{m=-\infty}^{\infty} w_L(t - mR) = 1 \tag{3}$$

2.2. STTR for Single Sinusoid Input

For a single sinusoid input $x(t) = \cos(2\pi f_0 t + \phi)$, the STTR output is

$$Y(f) = \frac{1}{2R} \sum_{k=-\infty}^{\infty} \Big\{ e^{i\phi} W_L(f_k^{+2}) \delta(f - f_k^{+1}) + e^{-i\phi} W_L(f_k^{-2}) \delta(f - f_k^{-1}) \Big\} \tag{4}$$

where $f_R = \frac{1}{R}$ and $f_k^{\pm a} = k f_R \pm a f_0$. f_R is called the frame rate.



(a) Input signal (b) Window length 1 ms

Figure 2: Spectrogram plots for a 10 second linear sine sweep from 0 Hz to 10 kHz (2a) and the STTR output with a Hann window of length 1 ms and 50% overlap-add (2b). The frame rate f_R is 2 kHz.

We see that STTR creates overtones at frequencies $f_k^{\pm 1}$, f_0 above and below multiples of f_R. Note that the fundamental frequency of the input is $f_0 = f_0^{+1}$. The amplitude for each overtone is proportional to $W_L(f_k^{\pm 2})$, the window spectrum sampled at $2f_0$ above and below multiples of f_R. This is illustrated in Figure 1a.

Figure 1b illustrates a general case where the overtone with the greatest amplitude is not the fundamental frequency f_0. This is further visible in Figure 2, STTR output for a linear sine sweep. The main diagonal is the fundamental frequency (Figure 2a), however the main diagonal of the output (Figure 2b) does not always have the greatest magnitude. This gives rise to the possibility of using STTR to harmonize an input signal.

2.3. Fixed Overlap-add Ratio

To simplify implementation, we use a fixed overlap-add ratio. In effect, the window function becomes dependent of the step size R. For a fixed overlap-add ratio $\alpha = R/L$, the window length is $L = R/\alpha = \frac{1}{\alpha f_R}$. The window spectrum $W_L(f) = W_{\frac{R}{\alpha}}(f)$ stretches in the frequency domain relative to f_R.

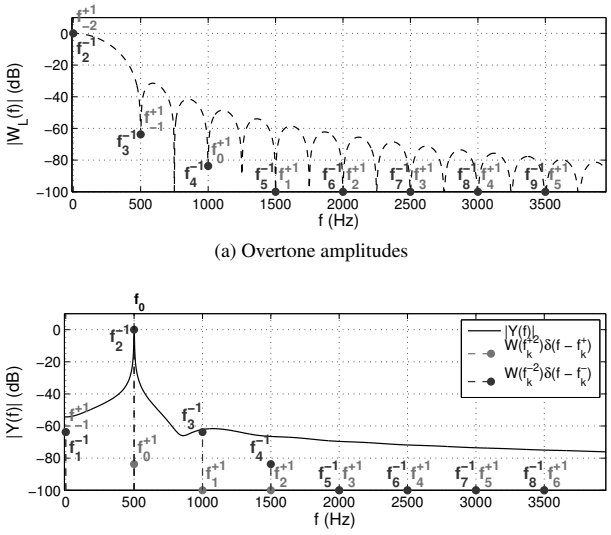
An interesting example is when the window function is of the generalized Hamming window family. In this case, the window spectrum is zero at frequencies $f = k \times 1/L = k \times \alpha f_R$, where k is an integer other than $-1, 0, 1$. For 50% overlap-add, the window spectrum will be zero at multiples of $f_R/2$ except for $f = 0, \pm f_R/2$. This means that for octaves of $f_R/2$, $f_0 = nf_R/2$, where n is a natural number, no overtones will occur.

3. OVERTONE ANALYSIS

In this section, we take a look at examples to understand the overtones created by STTR, then generalize the results to obtain an overtone map. For the examples, we will use 50% overlap-add ($\alpha = 0.5$) with a Hann window.

3.1. Example 1: $f_0 = f_R$

We first look at the simple case where the frame rate is equal to the input frequency. For $f_0 = f_R$, $W_{2R}(f) = 0$ at $f = k \times f_R/2$ for integer k other than $-1, 0, 1$. Using this fact, equation



(a) Overtone amplitudes

(b) Ideal overtones with STTR output spectrum

Figure 3: Plots illustrating the STTR overtones for $f_0 = f_R = 500$ Hz. Figure 3a shows the window spectrum and the points $W_L(f_k^{+2})$. The points are labeled with the corresponding overtone frequency f_k^{+1}. Figure 3b shows the resulting overtones (dotted) given by equation (4) together with the spectrum of the STTR output, $Y(f)$ (solid). Since the window function is a Hann window, most of the overtone amplitudes fall on the nulls.

(4) simplifies to,

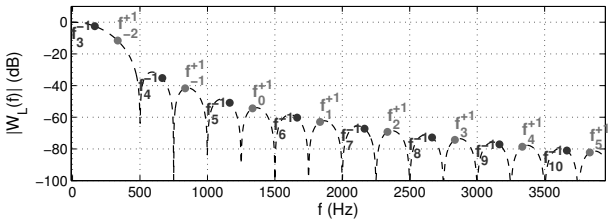
$$\begin{aligned} Y(f) =& \frac{1}{2R}\Big\{e^{i\phi}W_{2R}(f_{-2}^{+2})\delta(f - f_{-2}^{+1}) \\ &+ e^{-i\phi}W_{2R}(f_2^{-2})\delta(f - f_2^{-1})\Big\} \\ =& \frac{1}{2}\Big\{e^{i\phi}\delta(f + f_0) + e^{-i\phi}\delta(f - f_0)\Big\} \\ =& \overline{X(f)} \end{aligned} \tag{5}$$

The output in this case is the conjugate of the input, $\overline{X(f)}$, in the frequency domain, which is the time reversed version of the input signal in the time domain. This makes sense since we are reversing two full periods of $x(t)$ which in effect does nothing to the fundamental frequency. However, from an STTR perspective, it is an overtone at $f_2^{-1} = 2f_R - f_0$, not the original input $f_0 = f_0^{+1}$, that represents the signal (Figure 3).

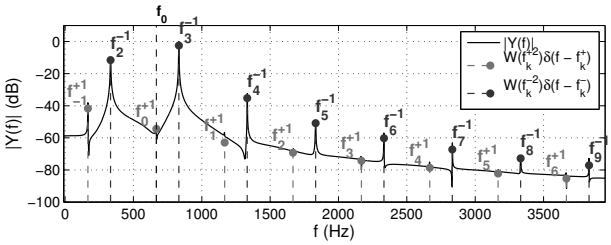
3.2. Example 2: 5 Semitone Difference

We next look at an example where the input frequency is a perfect 4th (5 semitones) above the frame rate, i.e. $f_0 = 2^{5/12} \times f_R$. For this case, most of the overtones are not zero (Figure 4) and thus equation (4) will not simplify like the previous case.

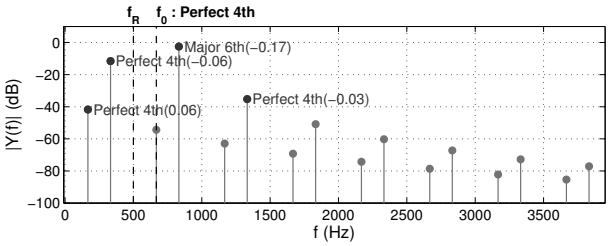
Figure 4b shows the annotated peaks along with the spectrum of the STTR output. We see that f_0 is not the most prominent frequency. To make tonal sense of the overtones, the peaks are annotated with the closest musical interval relative to f_R in Figure 4c. The most prominent overtone is approximately a major 6th



(a) Overtone amplitudes



(b) Ideal overtones with STTR output spectrum



(c) Overtones mapped to musical intervals

Figure 4: Plots illustrating the STTR overtones for $f_0 = 2^{5/12} \times f_R$ with a Hann window where $f_R = 500\,\text{Hz}$. $W(f_k^{\pm 2})$ is mostly non-zero in this case (Figure 4a), and as a result there are many overtones in the output signal (Figure 4b). Figure 4c shows the peaks greater than $-45\,\text{dB}$ annotated with the musical interval with f_R as the key. The greatest frequency is neither f_0 nor f_R.

Semitones	Interval	Overtones			
-12	**P1**	**P1 (0 +0.00)**			
-11	m2	m7 (-2 -0.06)	TT (18 -0.33)	**P1 (0 +0.00)**	
-10	**M2**	m6 (-4 -0.26)	**M3 (16 +0.30)**	**P1 (0 +0.00)**	
-9	m3	**P4 (-7 +0.37)**	m3 (15 -0.11)	**P1 (0 +0.00)**	**P1 (24 +0.20)**
-8	**M3**	m3 (-9 -0.21)	m2 (13 +0.45)	**P1 (0 +0.00)**	**M7 (23 -0.06)**
-7	**P4**	**P1 (-12 -0.06)**	**P1 (12 -0.03)**	**P1 (0 +0.00)**	m7 (22 -0.34)
-6	TT	**M6 (-15 -0.26)**	m7 (10 +0.45)	**P1 (0 +0.00)**	
-5	**P5**	**P4 (-19 +0.06)**	**M6 (9 -0.13)**		
-4	m6	**P5 (7 +0.25)**	m2 (-23 -0.33)	TT (18 -0.30)	
-3	**M6**	TT (6 -0.44)	**P5 (-29 +0.18)**	**M3 (16 +0.33)**	**P1 (0 +0.00)**
-2	m7	**M3 (4 -0.21)**	**P1 (-36 -0.36)**	m3 (15 -0.08)	**P1 (0 +0.00)**
-1	**M7**	**M2 (2 -0.05)**	**M7 (-49 +0.14)**	m2 (13 +0.48)	
0	**P1**	**P1 (0 +0.00)**			
1	m2	m7 (-2 -0.06)	m7 (10 +0.48)	m7 (-50 +0.14)	
2	**M2**	m6 (-4 -0.26)	**M6 (9 -0.09)**	m7 (-38 -0.36)	**M3 (16 +0.30)**
3	m3	**P4 (-7 +0.37)**	**P5 (7 +0.28)**	**M3 (-32 +0.18)**	m3 (15 -0.11)
4	**M3**	TT (6 -0.41)	m3 (-9 -0.21)		
5	**P4**	**M3 (4 -0.17)**	**P1 (-12 -0.06)**	**P1 (12 -0.03)**	**P1 (-24 +0.06)**
6	TT	**M2 (2 -0.02)**	**M6 (-15 -0.26)**	m7 (10 +0.45)	m3 (-21 -0.26)
7	**P5**	**P1 (0 +0.04)**			
8	m6	m7 (-2 -0.02)	**P5 (7 +0.25)**	m2 (-23 -0.33)	m2 (13 +0.25)
9	**M6**	m6 (-4 -0.22)	TT (6 -0.44)	**P5 (-29 +0.18)**	**P1 (12 -0.24)**
10	m7	**M3 (4 -0.21)**	**P4 (-7 +0.42)**	m7 (10 +0.24)	
11	**M7**	**M2 (2 -0.05)**	m3 (-9 -0.16)	**M6 (9 -0.34)**	**M7 (-49 +0.14)**
12	**P1**	**P1 (0 +0.00)**			

Figure 5: Visualization of an STTR overtone table. The STTR settings are 50% overlap-add using a Hann window. In this instance, the tonic is equal to f_R. The rows span 1 octave above and below the tonic. The table only shows overtones greater than -45 dB with respect to the input signal. The overtones are annotated with the closest musical interval with the actual semitone difference and cent offsets in parenthesis. The text is scaled linearly by the overtone amplitude. Overtones that fall on a major scale with f_R as the tonic are shown in bold.[1]

(9 semitones) above the frame rate and a major 3rd (4 semitones) above the input frequency. Other prominent overtones are octaves of the input frequency. Playing a major scale in the key of f_R, the STTR output of a perfect 4th will approximately be in harmony. From the perspective of the input signal, setting the frame rate to be 5 semitones lower than the input frequency will result in a major 3rd harmony.

3.3. STTR Overtone Table

Using the method in the example above, we can map overtones greater than a reasonable threshold to the closest musical interval and find harmonizing notes for each semitone difference between f_0 and f_R .

Figure 5 shows an example STTR overtone table with overtones on a major scale in bold. The first column lists the semitone difference relative to f_R. The second column shows the abbreviated musical interval with respect to the tonic. The abbreviation indicates the type of interval (Major, minor, Perfect) and the semitone difference from the key. TT denotes a tritone.

For this case, playing the notes 0, 5, 7, 12 semitones above f_R will keep the harmonies on a major scale. Not surprisingly, the aforementioned intervals are known as the perfect intervals, intervals that occur as harmonics.

One point to note is that the overtones may vary over octaves, though overall the overtones between octaves are similar. For example, in Figure 5 the overtones of a major third -8 semitones below f_R are different from those of a major third 4 semitones above f_R.

For more general use, we can pre-compute the prominent overtone frequencies and save the ratio between the overtone frequency and the frame rate, then use this as a lookup table for harmonizing.

This is an example of using STTR to harmonize an input signal. In the next section, different methods of using STTR as a harmonizing effect are covered.

[1] An interactive version of the STTR overtone table can be found at `https://ccrma.stanford.edu/~hskim08/sttr/harmonize/`. Source code, compiled audio plug-ins and sound examples are also available at the URL.

4. HARMONIZING METHODS

From equation (4), we can express the frequencies and amplitudes of the overtones as

$$f_k^{\pm 1} = f_R(k \pm \beta) = f_0(k/\beta \pm 1). \quad (6)$$

$$W_L(f_k^{\pm 2}) = W_L(f_R(k \pm 2\beta)) = W_L(f_0(k/\beta \pm 2)) \quad (7)$$

where $\beta = f_0/f_R$. The overtones can be expressed relative to the input frequency f_0 or the frame rate f_R.

Since f_0 will be determined by the input signal, to use STTR as a harmonizing effect we have three choices, a) fix f_R and choose f_0 respectively, b) fix β by changing f_R with respect to f_0 or c) change both f_R and β according to some rule.

4.1. Fixed f_R

In Section 3.3, we covered an example of harmonizing a signal with the tonic f_{tonic} to be equal to the frame rate f_R. An extension of this approach is to separate the tonic from the frame rate by specifying an offset factor n_{offset} such that $f_{\text{tonic}} = 2^{n_{\text{offset}}/12} f_R$.

Figure 6 is an overtone table with $n_{\text{offset}} = 5$. With this setting, most of the notes on a major scale will loosely sound harmonized with a few out-of-scale overtones.

Semitones	Interval	Overtones			
-7	**P1**	**P1 (-12 -0.06)**	**P1 (12 -0.03)**	**P1 (0 +0.00)**	m7 (22 -0.34)
-6	m2	**M6 (-15 -0.26)**	m7 (10 +0.45)	**P1 (0 +0.00)**	
-5	**M2**	**P4 (-19 +0.06)**	**M6 (9 -0.13)**		
-4	m3	**P5 (7 +0.25)**	m2 (-23 -0.33)	TT (18 -0.30)	
-3	**M3**	TT (6 -0.44)	**P5 (-29 +0.18)**	**M3 (16 +0.33)**	**P1 (0 +0.00)**
-2	**P4**	**M3 (4 -0.21)**	**P1 (-36 -0.36)**	m3 (15 -0.08)	**P1 (0 +0.00)**
-1	TT	**M2 (2 -0.05)**	**M7 (-49 +0.14)**	m2 (13 +0.48)	
0	**P5**	**P1 (0 +0.00)**			
1	m6	m7 (-2 -0.06)	m7 (10 +0.48)	m7 (-50 +0.14)	
2	**M6**	m6 (-4 -0.26)	**M6 (9 -0.09)**	m7 (-38 -0.36)	**M3 (16 +0.30)**
3	m7	**P4 (-7 +0.37)**	**P5 (7 +0.28)**	**M3 (-32 +0.18)**	m3 (15 -0.11)
4	**M7**	TT (6 -0.41)	m3 (-9 -0.21)		
5	**P1**	**M3 (4 -0.17)**	**P1 (-12 -0.06)**	**P1 (12 -0.03)**	**P1 (-24 +0.06)**
6	m2	**M2 (2 -0.02)**	**M6 (-15 -0.26)**	m7 (10 +0.45)	m3 (-21 -0.26)
7	**M2**	**P1 (0 +0.04)**			
8	m3	m7 (-2 -0.02)	**P5 (7 +0.25)**	m2 (-23 -0.33)	m2 (13 +0.25)
9	**M3**	m6 (-4 -0.22)	TT (6 -0.44)	**P5 (-29 +0.18)**	**P1 (12 -0.24)**
10	**P4**	**M3 (4 -0.21)**	**P4 (-7 +0.42)**	m7 (10 +0.24)	
11	TT	**M2 (2 -0.05)**	m3 (-9 -0.16)	**M6 (9 -0.34)**	**M7 (-49 +0.14)**
12	**P5**	**P1 (0 +0.00)**			
13	m6	m7 (-2 -0.06)	**P4 (5 +0.32)**	**M6 (-15 -0.19)**	m7 (10 +0.48)
14	**M6**	m6 (-4 -0.26)	**M3 (4 -0.45)**		
15	m7	**M2 (2 -0.31)**	**P4 (-7 +0.37)**	**P5 (7 +0.28)**	m2 (-23 -0.23)
16	**M7**	**P1 (0 -0.27)**	TT (6 -0.41)	m3 (-9 -0.21)	
17	**P1**	m7 (-2 -0.35)	**M3 (4 -0.17)**	**P1 (-12 -0.06)**	m6 (8 +0.37)

Figure 6: Visualization of an STTR overtone table (50% overlap-add, Hann window) with f_{tonic} 5 semitones above f_R. The first column shows the semitone difference of f_0 and f_R while the second column shows the musical interval of f_0 relative to f_{tonic}.[2]

The advantage of this approach is that it requires no additional computation apart from that of standard STTR. It also presents a characteristic harmonizing scheme determined by the window function. However, finding a setting that works for all notes on a desired musical scale becomes a nontrivial search problem.

4.2. Fixed β

Another approach would be to keep the ratio between f_0 and f_R constant. This will keep the overtone intervals constant relative to f_0. This can be viewed as a form of pitch shifting, especially in the case where there is a single dominant overtone. However for most cases, there are many overtones so it will be a harmonizing effect.

f_R now becomes a function of f_0 by $f_R(f_0) = f_0/\beta$. This approach requires a fundamental frequency (F0) estimator. F0 estimation is a well studied area with many proposed solutions [8, 9, 10, 11].

We also need to decide what value of β to use, or what set of overtones to use. In Figure 7, the overtones are mapped to the musical interval relative to f_0. We can use this table to select β and in turn choose the overtones to use.

Semitones	Overtones			
-12	**P1 (0 +0.00)**			
-11	m7 (-2 -0.06)	TT (18 -0.33)	**P1 (0 +0.00)**	
-10	m6 (-4 -0.26)	**M3 (16 +0.30)**	**P1 (0 +0.00)**	
-9	**P4 (-7 +0.37)**	m3 (15 -0.11)	**P1 (0 +0.00)**	**P1 (24 +0.20)**
-8	m3 (-9 -0.21)	m2 (13 +0.45)	**P1 (0 +0.00)**	**M7 (23 -0.06)**
-7	**P1 (-12 -0.06)**	**P1 (12 -0.03)**	**P1 (0 +0.00)**	m7 (22 -0.34)
-6	**M6 (-15 -0.26)**	m7 (10 +0.45)	**P1 (0 +0.00)**	
-5	**P4 (-19 +0.06)**	**M6 (9 -0.13)**		
-4	**P5 (7 +0.25)**	m2 (-23 -0.33)	TT (18 -0.30)	
-3	TT (6 -0.44)	**P5 (-29 +0.18)**	**M3 (16 +0.33)**	**P1 (0 +0.00)**
-2	**M3 (4 -0.21)**	**P1 (-36 -0.36)**	m3 (15 -0.08)	**P1 (0 +0.00)**
-1	**M2 (2 -0.05)**	**M7 (-49 +0.14)**	m2 (13 +0.48)	
0	**P1 (0 +0.00)**			
1	m7 (-2 -0.06)	m7 (10 +0.48)	m7 (-50 +0.14)	
2	m6 (-4 -0.26)	**M6 (9 -0.09)**	m7 (-38 -0.36)	**M3 (16 +0.30)**
3	**P4 (-7 +0.37)**	**P5 (7 +0.28)**	**M3 (-32 +0.18)**	m3 (15 -0.11)
4	TT (6 -0.41)	m3 (-9 -0.21)		
5	**M3 (4 -0.17)**	**P1 (-12 -0.06)**	**P1 (12 -0.03)**	**P1 (-24 +0.06)**
6	**M2 (2 -0.02)**	**M6 (-15 -0.26)**	m7 (10 +0.45)	m3 (-21 -0.26)
7	**P1 (0 +0.04)**			
8	m7 (-2 -0.02)	**P5 (7 +0.25)**	m2 (-23 -0.33)	m2 (13 +0.25)
9	m6 (-4 -0.22)	TT (6 -0.44)	**P5 (-29 +0.18)**	**P1 (12 -0.24)**
10	**M3 (4 -0.21)**	**P4 (-7 +0.42)**	m7 (10 +0.24)	
11	**M2 (2 -0.05)**	m3 (-9 -0.16)	**M6 (9 -0.34)**	**M7 (-49 +0.14)**
12	**P1 (0 +0.00)**			

Figure 7: Visualization of an STTR overtone table (50% overlap-add, Hann window) with the overtones annotated with musical intervals relative to f_0. The "Semitones" column in this table can be translated to β, the ratio between f_0 and f_R.[3]

[2] https://ccrma.stanford.edu/~hskim08/sttr/harmonize/vis.html?offset=5.

[3] https://ccrma.stanford.edu/~hskim08/sttr/harmonize/vis.html?relative=1.

4.3. Variable f_R and β

One possibility of solving the limitations of the two previous examples, would be a hybrid approach where for a given f_0 we would find a value β that keeps the overtones within a given key with tonic f_{tonic}. In this case, β becomes a function of f_0. Thus $f_R(f_0) = f_0/\beta(f_0, f_{\text{tonic}})$. This becomes an extension of the search problem briefly mentioned in Section 4.1. Here we need to find a function or mapping $\beta(f_0, f_{\text{tonic}})$ such that the resulting overtones mostly fall on the target scale. This is an open-ended design problem that requires further study.

5. IMPLEMENTATION

The harmonizing methods covered in the previous section work by setting the frame rate f_R. It does not affect the STTR implementation itself. Only a front-end for controlling f_R needs to be added. For STTR, the real-time implementation presented in the previous paper [1] can be used. For STTR with 50% overlap-add, it was shown that the implementation only requires one delay line per channel and two reader pointers that are shared between the channels.[4]

5.1. Fixed f_R

To implement the STTR harmonizing effect with fixed f_R, the frame rate needs to be restricted to a reasonable range. The STTR implementation in [1] allowed for step sizes from 0.5 ms to 0.5 s. For user convenience, we changed the *window length* parameter, which is in milliseconds, to a MIDI note number ranging from 48 to 72, one octave below and above middle C (MIDI number 60). This corresponds to a step size of 1.9111 ms to 7.6445 ms. We add a *Fine Tune* parameter to allow a user to adjust f_R by $\pm$ 50 cents.

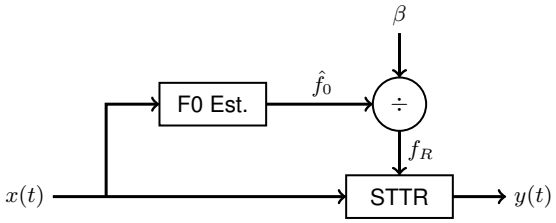
Figure 8: Audio plug-in for fixed f_R harmonizing effect. f_R parameter is labeled *Key*.

5.2. Fixed β

For the fixed β implementation, an F0 estimator is required to control the frame rate. For our implementation, we use the YIN algorithm [9]. We expose β, the semitone offset of between f_0 and f_R, as a user parameter. We add a *Fine Tune* parameter to adjust β by $\pm$ 50 cents.

While the YIN algorithm works well for single note input, it does not work well with signals with multiple fundamental frequencies. The decay from a previous note will often cause momentarily erroneous F0 estimates. While using a multiple F0 estimator [10, 11] can fix this problem, we are left with the problem of deciding the harmonizing strategy for multiple fundamental frequencies.

[4]Sound examples using the implementations explained can be found at `https://ccrma.stanford.edu/~hskim08/sttr/harmonize/examples.html`.



(a) Fixed β schematic

(b) Audio plug-in for fixed β harmonizing effect

Figure 9: Schematic and audio plug-in implementation of fixed β harmonizing effect. In Figure 9b, β parameter is labeled *Harmonize*. The F0 estimation is shown below the controls.

Each time the estimated f_0 changes, f_R and, in turn, the window length L needs to be updated. This change is accommodated by adjusting the positions of the windows and readers with respect to the writer position (Figure 10). This keeps the average lag between the input buffer readers and the output buffer writer to one window length. As a side effect, when the window length changes, there is a momentary pitch shift caused by speed change of the readers. To reduce discontinuity artifacts, L is gradually changed with a set slew rate.

6. CONCLUSION

For short window lengths, STTR creates overtones through sideband modulation. This property allows STTR to be used as a harmonizing effect without an explicit pitch shifting implementation. Thus, STTR can be used as an efficient method of implementing a harmonizing effect. We investigated the tonal relation of the overtones with respect to the frame rate and the frequency of a single sinusoidal input. These tonal relations can be saved as an STTR overtone table which then can be used as a guide when using STTR as a harmonizing effect. Three methods were proposed for harmonizing a signal using STTR of which two were covered in detail. The first method, fixed f_R, directly used an STTR overtone table for finding intervals that work for a given musical scale. The second method, fixed $\beta = f_0/f_R$, kept the overtones at a constant interval relative to the input signal. Implementation of the two methods requires little modification to the general STTR implementation. A front-end that controls the frame rate needs to be added which in the first case is trivial and in the latter case requires an F0 estimator.

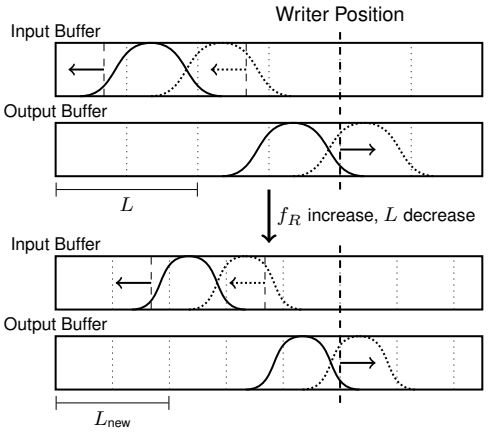


Figure 10: Illustration of STTR adjusting to an increase in the frame rate f_R. The reader and window positions of the input buffer are adjusted relative to the writer position of the output buffer, keeping an average lag of one window length. The arrows in the input/output buffers depict the reader/writer positions and directions respectively.

The analysis in this paper was focused on single sinusoids signals. The next step would be to analyze the effects of signals with harmonics. From our observations for STTR with 50% overlap-add using a Hann window, octaves generally had similar overtone relations to the input signal, though not strictly identical. However this cannot be said for the partials.

The requirements for constant overlap-add windows for the 50% overlap-add case was previously covered, outlining the possibility of designing windows [1]. By designing window functions tailored to STTR harmonizing, we could further control the overtones.

In Section 4.3, a variable β harmonizing scheme was briefly covered. Together with window design, this opens possibilities of extending STTR harmonizing such that it can harmonize input signals to a given scale without the limitations of the two implemented methods.

7. REFERENCES

[1] H. S. Kim and J. O. Smith, "Short-time time-reversal on audio signals," in *Proc. of 17th Int. Conf. on Digital Audio Effects (DAFx-14)*, Erlangen, Germany, September 2014.

[2] P. Dutilleux, G. De Poli, and U. Zölzer, "Time-segment processing," *DAFX: Digital Audio Effects*, pp. 201–236, 2002.

[3] A. Agnello, "Method and apparatus for producing two complementary pitch signals without glitch," Patent US 4 369 336, January 18, 1983.

[4] K. Bogdanowicz and R. Belcher, "Using multiple processors for real-time audio effects," in *Audio Engineering Society Conference: 7th Int. Conf.: Audio in Digital Times*. Audio Engineering Society, 1989.

[5] S. Disch and U. Zölzer, "Modulation and delay line based digital audio effects," in *2nd Workshop on Digital Audio Effects DAFx*. Citeseer, 1999.

[6] J. Laroche and M. Dolson, "New phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 1999, pp. 91–94.

[7] A. Röbel and X. Rodet, "Efficient spectral envelope estimation and its application to pitch shifting and envelope preservation," in *Proc. of 8th Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, September 2005.

[8] A. von dem Knesebeck and U. Zölzer, "Comparison of pitch trackers for real-time guitar effects," in *Proc. of 13th Int. Conf. on Digital Audio Effects (DAFx-10)*, Graz, Austria, September 2010.

[9] A. De Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[10] A. de Cheveigné, "Multiple f0 estimation," *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*, pp. 45–72, 2006.

[11] A. P. Klapuri, "Multiple fundamental frequency estimation based on harmonicity and spectral smoothness," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 6, pp. 804–816, 2003.

BARBERPOLE PHASING AND FLANGING ILLUSIONS

Fabián Esqueda, Vesa Välimäki*
Aalto University
Dept. of Signal Processing and Acoustics
Espoo, Finland
fabian.esqueda@aalto.fi
vesa.valimaki@aalto.fi

Julian Parker
Native Instruments GmbH
Berlin, Germany
julian.parker@native-instruments.de

ABSTRACT

Various ways to implement infinitely rising or falling spectral notches, also known as the barberpole phaser and flanging illusions, are described and studied. The first method is inspired by the Shepard-Risset illusion, and is based on a series of several cascaded notch filters moving in frequency one octave apart from each other. The second method, called a synchronized dual flanger, realizes the desired effect in an innovative and economic way using two cascaded time-varying comb filters and cross-fading between them. The third method is based on the use of single-sideband modulation, also known as frequency shifting. The proposed techniques effectively reproduce the illusion of endlessly moving spectral notches, particularly at slow modulation speeds and for input signals with a rich frequency spectrum. These effects can be programmed in real time and implemented as part of a digital audio processing system.

1. INTRODUCTION

Shepard introduced in the 1960s the infinitely ascending chromatic scale, which was produced with additive synthesis [1, 2]. Risset expanded this idea by designing a continuously rising and falling sweep [3, 4]. The spectrum of two instances of the Shepard tone are shown in Figure 1. It is seen that the sinusoidal components are equally spaced in the logarithmic frequency scale, as each component is one octave higher than the previous one. A bell-shaped spectral envelope function takes care of the fade-in and fade-out of harmonic components. In addition to Shepard-Risset tones, other auditory illusions have been discovered, including binaural paradoxes [5] and rhythms which appear to be gaining speed in a neverending manner [4, 6]. This paper discusses impossible-sounding phasing and flanging effects inspired by the Shepard-Risset tones.

Flanging is a delay-based audio effect which generates a series of sweeping notches on the spectrum of a signal [7, 8, 9, 10]. Historically, analog flanging was achieved by mixing the output of two tape machines and varying their speed by applying pressure on the flanges—hence the effect's name [7, 9, 10]. Adding a signal with a delayed version of itself results in a comb filtering effect, introducing periodic notches in the output spectrum. As the length of the delay changes over time, the number of notches and their position also changes, producing the effect's characteristic swooshing or "jet aircraft" sound [7, 9]. Wanderley and Depalle [11, 12] have found that a flanging effect is also produced when a musician moves in front of the microphone while playing, as the time delay between the direct sound and its reflection from the floor is varied.

Phasing was introduced in effect pedals as a simulation of the flanging effect, which originally required the use of open-reel tape machines [8]. Phasing is implemented by processing the input signal with a series of first- or second-order allpass filters and then adding this processed signal to the original [13, 14, 15]. Each allpass filter then generates one spectral notch. When the allpass filter parameters are slowly modulated, the notches move up and down in frequency, as in the flanging effect. The number and distribution of the notches are the main differences between phasing and flanging, which can sometimes sound quite similar.

Bode developed a barberpole phaser in which the spectral notches move endlessly in one direction in frequency [16]. The name 'barberpole' stems from a rotating cylindrical sign, usually white with a red stripe going around it, which have been traditionally used in front of barber shops in England and in the US. As the pole rotates, a visual illusion of the red stripe climbing up endlessly along the pole is observed, although the pattern is actually

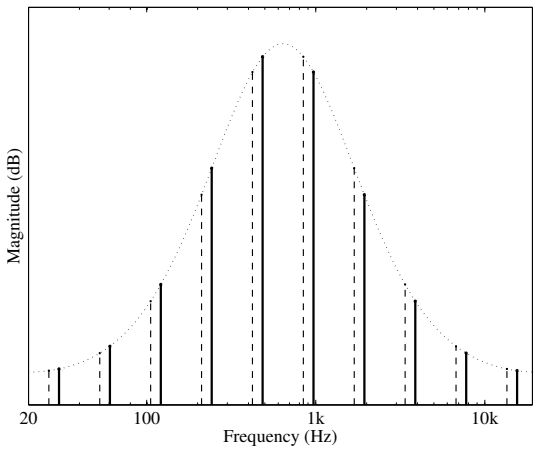


Figure 1: *Spectrum of a Shepard tone with 10 harmonics (solid lines), spaced one octave apart, and a raised-cosine spectral envelope (dotted line) [1]. The dashed lines show the harmonics a short time earlier.*

* The work of Fabián Esqueda is supported by the CIMO Centre for International Mobility and the Aalto ELEC Doctoral School.

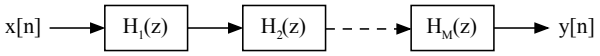


Figure 2: *Block diagram for the proposed network of M time-varying notch filters $H_m(z)$ for $m = 1, 2, 3, ..., M$. The center frequencies of these filters are situated at one-octave intervals.*

stationary. Barberpole phasing and flanging effects are currently available in some audio software, but it is not known to us how they are implemented [17, 18].

Related recent work has focused on virtual analog models of vintage flanger and phaser circuits, such as the digital modeling of the nonlinear behavior caused by the use of operational transconductance amplifiers [19] and bucket-brigade devices [20]. Eichas et al. [21] have presented a detailed virtual analog model of a famous phaser. Furthermore, some research has focused on understanding how flanging, phasing, and other audio effects processing is recognized by humans [22] or by the computer [23].

In this paper we investigate ways to implement barberpole phasing and flanging effects. The inspiration for this comes from inverting the Shepard-Risset tone, i.e. replacing the spectral peaks with notches, to create a new impossible audio effect. In the end, we found that there are at least three different principles to obtain this effect. This paper is organized as follows. Section 2 discusses the basic cascaded notch filter technique to simulate the barberpole phasing effect. Section 3 introduces a novel flanging method using a pair of delay lines. Section 4 describes a third method, derived from that of Bode, which has its roots in single-sideband modulation. Finally, Section 5 provides some concluding remarks.

2. CASCADED TIME-VARYING NOTCH FILTERS

The illusion of endless rising or falling sweeping notches, similar to that of the phasing effect, can be achieved using a network of cascaded time-varying notch filters (see Figure 2). To do so, we follow the design of the Shepard tone and place the center frequencies of the filters at one-octave intervals. This design choice translates into notches uniformly distributed along the logarithmic frequency axis. The amount of attenuation caused by each filter is determined using an inverted version of the raised-cosine envelope originally proposed by Shepard [1] (see Figure 1).

Considering the case of a rising notch sweep, as the center frequencies of the filters move up the spectrum, notches approaching the Nyquist limit (f_N) will gradually disappear. Similarly, notches coming from the low end of the spectrum will increase in depth as they reach the middle frequencies. Since the one-octave interval between notches is preserved at every time step, the center frequencies of the filters will eventually reach twice their initial value. At this point, we say the system has completed one full cycle. If the filter parameters are continuously reset one time step before the system completes a cycle, the illusion of an infinite filter sweep is generated. Therefore, to implement the illusion we only need to derive the center frequencies and their respective gain values for a single cycle. These parameters can then be stored in a table and read indefinitely during implementation.

To compute the necessary initial parameters, we begin by defining a system of M cascaded notch filters and denote the repetition rate of the effect, in Hz, by ρ. The total number of center frequencies (denoted by K) each filter will go through before it completes a full cycle is the same for every filter and is determined by

$$K = \lfloor F_s/\rho \rfloor, \tag{1}$$

where F_s is the sampling rate of the system. The k^{th} center frequency for the m^{th} filter can then be computed from

$$f_c(m,k) = f_0 2^{[K(m-1)+k-1]/K} \tag{2}$$

for $k = 1, 2, 3, ..., K$ and $m = 1, 2, 3, ..., M$. The parameter f_0 is the center frequency of the first filter at the beginning of a cycle. Next, the k^{th} center gain of the m^{th} filter is defined as

$$L_c(m,k) = L_{\min} + \frac{(L_{\max} - L_{\min})(1 - \cos[\theta(m,k)])}{2}, \tag{3}$$

where $L_{\min}$ and $L_{\max}$ are the minimum and maximum attenuation levels in dB, respectively, and $L_{\max} < L_{\min} < 0$. The function θ is defined as

$$\theta(m,k) = 2\pi \frac{(m-1)K + k - 1}{MK}. \tag{4}$$

In summary, we must implement M notches that sweep uniformly throughout K frequencies, each with its own attenuation level. In order to achieve this amount of control for each filter, we can use a *parametric equalizer* (EQ) filter structure [24]. This type of second-order IIR filter is commonly used in graphic equalizers, since it allows users to increase (boost) or reduce (cut) the gain of a specific frequency band.

The z-domain transfer function for the cutting case of the parametric equalizer filter is given by

$$H(z) = \frac{\left(\frac{1+G\beta}{1+\beta}\right) - 2\left(\frac{\cos(\frac{2\pi f_c}{F_s})}{1+\beta}\right) z^{-1} + \left(\frac{1-G\beta}{1+\beta}\right) z^{-2}}{1 - 2\left(\frac{\cos(\frac{2\pi f_c}{F_s})}{1+\beta}\right) z^{-1} + \left(\frac{1-\beta}{1+\beta}\right) z^{-2}}, \tag{5}$$

where G is the scalar gain at the center frequency f_c (i.e. $10^{L_c/20}$) and β is defined as

$$\beta = \sqrt{\frac{G_B^2 - 1}{G^2 - G_B^2}} \tan\left(\frac{\Delta\omega}{2}\right), \tag{6}$$

where $\Delta\omega$ is the width of the filter at gain level G_B [24]. Figure 3 illustrates the relationship between these three parameters for a filter with arbitrary center frequency f Hz.

Now, the definition of $\Delta\omega$ is rather ambiguous in this case. It is generally taken to be the width of the filter 3 dB below the reference level (i.e. $G_B^2 = 1/2$). In our case, since notches near DC and f_N may not reach this level of attenuation, this particular definition is inadequate. For this reason, we instead define the filter bandwidths in terms of their Q factor

$$Q = \frac{2\pi f_c}{\Delta\omega F_s} \Leftrightarrow \Delta\omega = \frac{2\pi f_c}{QF_s}. \tag{7}$$

Maintaining a constant Q for every filter, rather than a constant $\Delta\omega$, will ensure the notches are equally wide on the logarithmic axis. Otherwise, notches near DC would be much wider than those near f_N. Finally, we define G_B^2 to be arithmetic mean between a reference gain of 1 and G, yielding

$$G_B^2 = \frac{1 + G^2}{2}. \tag{8}$$

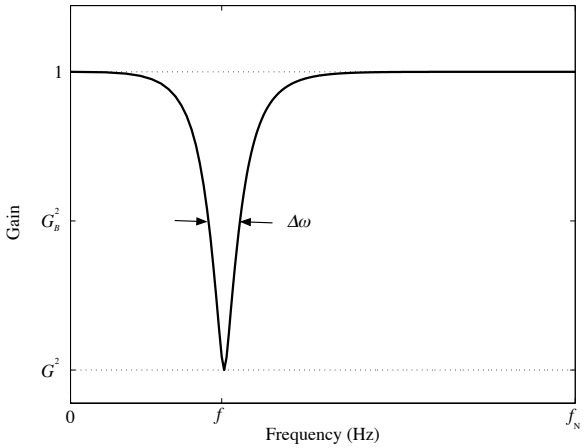


Figure 3: *General form of the magnitude response of the parametric EQ filter in cut mode.*

Figure 4 shows the magnitude response of the proposed system at its initial state. This implementation was realized with parameters $M = 10$, $\rho = 0.1$ Hz, $f_0 = 20$ Hz, $Q = 15$, $L_{\max} = -20$ dB and $L_{\min} = -3$ dB. A sampling rate $F_s = 44.1$ kHz was used for this and the rest of the examples in this paper. As we can see from the spectrum, the envelope of the notches resembles an inverted version of its Figure 1 counterpart. As expected, keeping the value of Q constant produces a fairly uniform notch distribution.

Figure 5 shows the spectrogram of a 30-second simulation of the barberpole illusion with the same parameters as in Figure 4 and white noise as the input signal. For this spectrogram and all those presented in this paper, a 1024-sample Chebyshev window with 100 dB of sidelobe attenuation, along with 512 samples of overlap were used. To increase image resolution at low frequencies, this particular signal was oversampled by a factor 10. The points in time where each cycle begins are marked with the three markers on top of the figure. Overall, in Figure 5 we can appreciate how, as

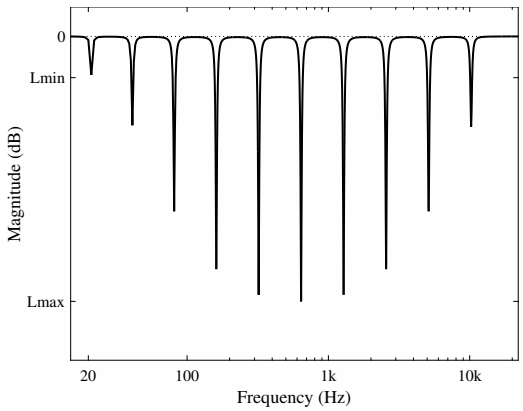


Figure 4: *Magnitude response of a network of 10 cascaded parametric notch filters. The gain (attenuation) at each center frequency is determined by the raised-cosine envelope.*

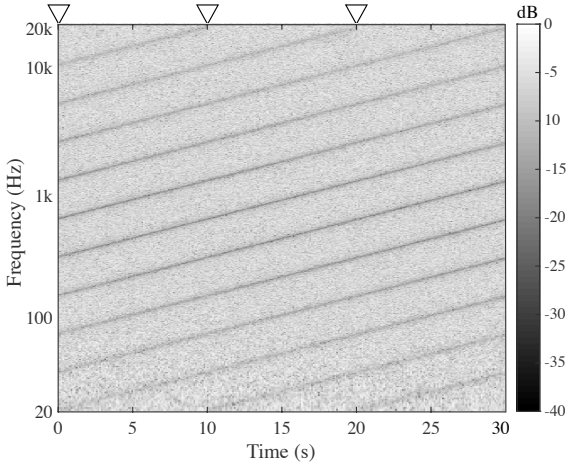


Figure 5: *Spectrogram of a barberpole phaser illusion implemented using white noise filtered with $M = 10$ cascaded time-varying notch filters and $\rho = 0.1$. Starting points of the cycles are indicated with triangles.*

the notches increase in frequency and approach the Nyquist limit, new ones begin to appear. Additionally, notches around the lower and upper ends of the spectrogram are clearly less dark, which translates into less attenuation. This system implementation requires f_0 and M to be chosen appropriately in order to ensure the last filter in the chain ends as close to the Nyquist limit as possible.

When implementing this illusion using a software routine, one additional consideration must be made. Once any given filter m reaches its K^{th} center frequency, it must reset to $k = 1$ at its next step. However, at this first time step of the new cycle the EQ filter requires the state variables of the previous filter $m - 1$ to correctly compute the output [25]. Therefore, the two final outputs and values of the delay state variables of each filter, which occur when $k = K - 1$ and $k = K$, must be passed on to the next filter. Failing to do so will introduce transients at the output of the network which will translate to audible clicks at the end of every cycle [25]. This would reveal the cyclic nature of the system and break the illusion.

Overall, this implementation of the barberpole phaser illusion works best for input signals with a dense and nearly flat magnitude spectrum, e.g. pink noise or noisy drum loops. Additionally, its parameters must be tuned differently for each input type. In terms of suitable values for ρ, numerous tests revealed that the illusion works best for values below 0.3 Hz. At higher rates, the cyclic nature of the design is also revealed and the illusion fails. This issue is also inherent to both the Shepard scale and the Shepard-Risset glissando which only work at slow playback rates.

Sound examples for this section can be found online at the accompanying website[1].

[1] `http://research.spa.aalto.fi/publications/papers/dafx15-barberpole`

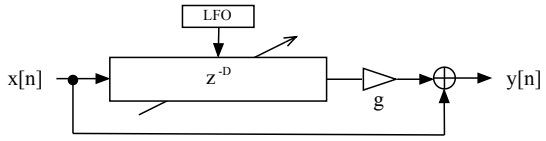


Figure 6: *Block diagram of a basic digital flanger effect.*

3. SYNCHRONIZED DUAL FLANGER

In the barberpole phasing approach discussed in the previous section, the number of notches remains constant throughout every cycle. This behavior is different to that of flanging, where the number of notches, along with their position, varies over time.

A barberpole version of the flanger effect can be implemented following the typical structure of a flanger in the digital domain, shown in Figure 6. In this system, the input is delayed by D samples, scaled by a gain factor g that controls the depth of the effect, and combined with the original signal. The length of the delay line is modulated by a sinusoidal or triangular low-frequency oscillator (LFO). Typical LFO rate values range between 0-3 Hz, while the maximum delay introduced by the delay line can be of up to 15 ms [9]. As the length of the delay line oscillates over time, the number of notches introduced and their position along the spectrum changes.

To implement the barberpole illusion, we first need to ensure the displacement of the notches is unidirectional. One possible solution is to use a sawtooth waveform for the LFO, which would ensure the length of the delay line is reset after every cycle. A trivial sawtooth LFO $s(n)$ can be synthesized for this purpose using a modulo counter [26]

$$s(n) = (D_{\max} - D_{\min})[(n\Delta) \bmod 1] + D_{\min}, \qquad (9)$$

where n is the time step index, $D_{\min}$ and $D_{\max}$ are the minimum and maximum delay lengths (in samples), respectively, and $\Delta = \rho/F_s$ is the phase increment. As before, ρ is the rate of the effect. Since this waveform resembles an ascending ramp, the length of the delay line will gradually increase until it reaches $D_{\max}$. In the frequency domain this represents an increasing density of notches that move towards the lower end of the spectrum as they distribute themselves uniformly along the linear frequency axis. This effect is perceived as a descending filter sweep, contrary to the ascending nature of the LFO. To implement the effect in the opposite direction we just need to flip $s(n)$ horizontally, i.e. $s'(n) = (D_{\max} + D_{\min}) - s(n)$.

The abrupt transition from the maximum to the minimum delay lengths and vice versa can be described as a "hard reset" of the delay lines. This reset will generate a sudden change in the frequency content of the output signal that does not support the illusion of an infinitely ascending/descending sweep. To fix this issue the design can be extended to incorporate a second delay line at the output of the first one and use cross-fading to switch between them. The second delay line will have the same characteristics as the first one but its instantaneous length will be controlled by a 90° shifted version of the modulating LFO.

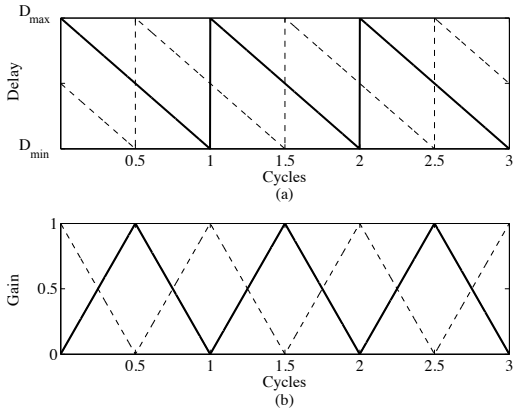


Figure 8: *Waveforms for three cycles of (a) delay-line length controls LFO 1 (solid line) and LFO 3 (dashed line) and (b) gain controls g_1 (solid line) and g_2 (dashed line) of the synchronized dual flanger for the ascending flanger case, showing the 90° phase offset between pairs of oscillators. Cf. Figure 7.*

Figure 7 shows the block diagram of the proposed system. Two new LFOs (LFO 2 and LFO 4) have been added in order to modulate the gain blocks at the output of each delay line. The basic concept behind this design is to avoid the hard resets by switching between delay lines as they approach their maximum/minimum delay length. Triangular LFOs with a rate of ρ Hz can be used, for example, to implement a linear cross-fade. These new oscillators must be synchronized with the delay line modulators. Therefore, LFO 4 should also have a 90° phase shift in relation to LFO 2. Figure 8 shows the waveforms for three cycles of the four LFOs suggested for the design (ascending flanging case).

In order to make the notches travel smoothly in frequency the design can be expanded to incorporate fractional delay filters [27, 28], since modulation of the delay lengths will most likely require fractional delay lengths. Naïve implementations usually resort to rounding off these values, making the notches sweep in a step-like manner and introducing the well-known "zipper noise".

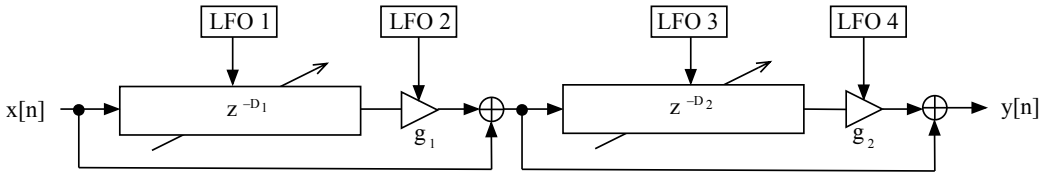


Figure 7: *Block diagram of the proposed synchronized dual flanger system.*

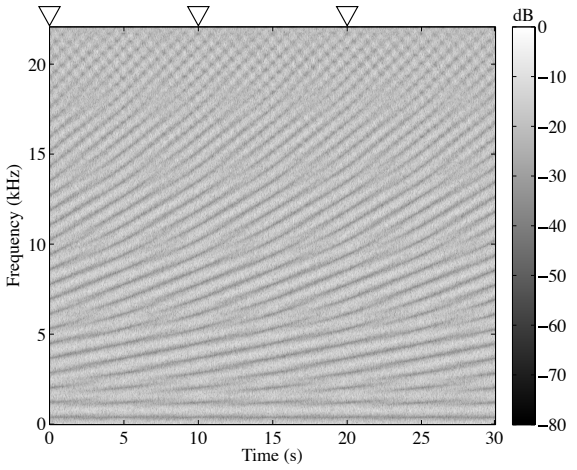


Figure 9: *Spectrogram of white noise processed by the system described by Figure 7 and parameters* $\rho = 0.1\,Hz$, $D_{\max} = 66$ *and* $D_{\min} = 44$. *The triangular markers atop the spectrogram indicate the points where the first delay line begins its cycle.*

Figure 9 shows the spectrogram for a 30-second simulation of the effect using white noise as an input. For this example, the system parameters were set as $\rho = 0.1\,\text{Hz}$, $D_{\max} = 66$ and $D_{\min} = 44$. Additionally, a third-order Lagrangian fractional delay filter was used to accommodate fractional delay lengths. As we can see from the spectrogram, the two flangers cross-fade over time and generate the illusion of a continuous sweep. The blurred portions on the spectrum represent the points were the two flangers meet. At high frequencies, the spectrogram shows the notch sweep is not as uniform as at low frequencies. This can be attributed to the lowpass response of the fractional delay interpolator and can be minimized using a higher order filter. Ideally, $D_{\min}$ should be higher than half the value of $D_{\max}$. Otherwise, it becomes difficult to hide the hard resets and the illusion is broken. As before, this effect also works best at low rates.

Overall, this implementation can be heard on virtually any input signal. This can be attributed to the larger number of notches that can be easily achieved with considerably fewer operations per sample compared to the approach discussed in Section 2. However, the effect is still clearly more dramatic on signals with a relatively dense spectrum e.g. distorted guitars or drum loops. A real-time implementation of this effect can be easily achieved using circular buffers.

4. SINGLE-SIDEBAND MODULATION

A commonly known historical technique of producing barberpole-like phasing or flanging effects is to employ single-sideband (SSB) modulation, also known as frequency shifting. This technique was first described by Harald Bode [16]. The modulated signal is mixed with the un-modulated signal, producing notches in the combined spectrum which move at a rate dictated by the amount of frequency-shift applied. Feedback may be applied around this structure to strengthen the effect.

This effect initially seems counter-intuitive, as it contradicts our usual experience of combining two identical waveforms with

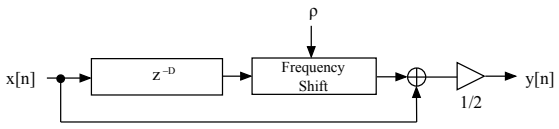


Figure 10: *Block diagram showing generalized SSB-modulation based barberpole effect, capable of producing linearly spaced moving notches.*

differing frequencies. Given an ideal frequency shifter operating on a sinusoid, with the frequency shifted sinusoid being mixed equally with the original sinusoid, we can write an expression describing the combination as a single sinusoid with time-varying amplitude and phase-shift:

$$\sin(\omega t) + \sin(\omega t + \omega_o t) = A(t)\sin[\omega t + \phi(t)], \qquad (10)$$

where ω gives the angular frequency of the sinusoid, ω_o gives the amount of frequency shift, and t denotes time. $A(t)$ and $\phi(t)$ give the amplitude and phase of the combined waveform. With some manipulation and application of trigonometric identities, we can write an expression for the amplitude as:

$$A(t) = \sqrt{2 + 2\cos(\omega_o t)}. \qquad (11)$$

This expression seems to confirm our initial intuition, that combining two signals separated by a constant frequency offset should produce frequency independent beating. This is not consistent with the effect described by Bode and others. However, if a time delay between the shifted and un-shifted signals is added, the result is quite different. Given a time delay of τ, we have

$$\sin(\omega t) + \sin[(\omega + \omega_o)(t - \tau)] = A(t)\sin[\omega t + \phi(t)]. \qquad (12)$$

Solving for A again, the following is produced:

$$A(t) = \sqrt{2 + 2\cos(\omega_o t - \omega_o \tau - \omega \tau)}. \qquad (13)$$

The addition of the time delay means that the phase of the amplitude variation with time for a particular frequency is offset depending on the frequency. If we assume linearity of the frequency shifter and delay, we can extend this result to an arbitrary input signal. Equation (13) then describes a set of notches in the frequency response of system, spread linearly over the frequency range and moving constantly in either the positive or negative frequency direction over time. The number of notches is related to the time delay, τ, and the rate and direction of movement is set by the frequency shift amount, ω_o.

The system described by Bode does not contain an explicit delay element. However, it relies on a 'dome filter'—a parallel system of two chains of allpass filters with a relative phase delay difference of $\frac{\pi}{2}$. The allpass filter chains themselves introduce some (frequency-dependent) delay to the signal, therefore no additional delay element is needed to produce the effect.

Given the knowledge gained so far, we can design a generalised SSB-modulation based barberpole flanging effect with the structure described in Figure 10, where the rate (in Hz) and direction are controlled by $\rho = w_o f_s / 2\pi$. The length of the delay line $D = \tau f_s$ is related to the number of notches M by $M = D/2$.

In our generalised system, the design of the frequency shifter block is assumed to be based on the common method of quadrature

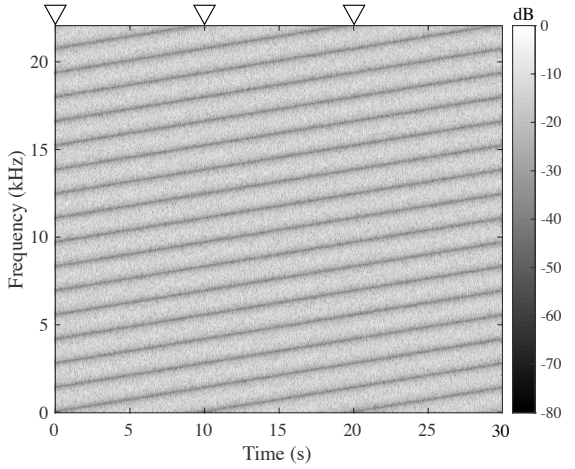


Figure 11: *Spectrogram of white noise processed by the structure given in Figure 10. The delay is 32 samples, and the amount of frequency shift is 0.1 Hz. The markers on top indicate when the system starts a new cycle.*

modulation of the real and imaginary parts of the analytic signal [29]. The analytic signal is produced via applying a Hilbert transform, which can be implemented in a variety of ways—as an FIR [30, 31] or IIR filter [32], as a digital version of the 'dome-filter' approach [33] taken by Bode, or via the use of an FFT. In this work we use the Matlab implementation of the Hilbert transform, which is based on the FFT.

The output of the system of Figure 10 when used to process an input of white noise is shown in Figure 11. The structure produces an interesting barberpole phasing effect, but does not convincingly produce the illusion of circularity given by the Shepard-Risset glissando. For this, octave distribution of the notches is necessary.

4.1. Warping the distribution of notches

Referring again to (13), we can see that to vary the spacing of the notches it is necessary to make the time delay, τ dependent on the frequency, ω. This can be achieved by replacing the delay line in the structure given in Figure 10 with a spectral delay [34], which can be implemented using a chain of first-order allpass filters, given by

$$A^D(z) = \left(\frac{a + z^{-1}}{1 + az^{-1}} \right)^D, \tag{14}$$

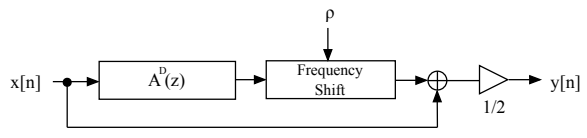


Figure 12: *Block diagram showing SSB-modulation based barberpole effect using a spectral delay filter, capable of producing warped distribution of notches.*

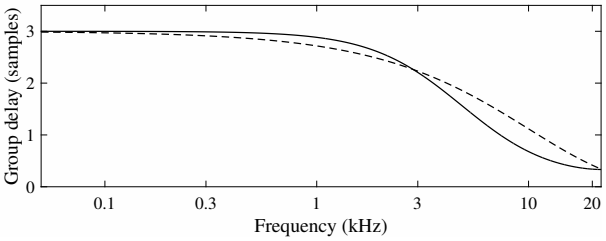


Figure 13: *Desired group delay* $3 \cdot 2^{-\omega}$ *of the spectral delay filter (dashed line) and its approximation using a first-order allpass filter with* $a = -0.5$ *(solid line).*

where a is the allpass filter coefficient and D is the number of stages. Spectral delay filters have been previously used for example for spring reverb emulation [35, 36] and for modeling the Leslie effect in Hammond organ synthesis [37].

The block diagram for the expanded configuration is shown in Figure 12. As in the linearly distributed case, the number of notches produced by this structure is given by $M = D/2$. The group delay of a first-order allpass filter is given by [34]

$$\tau_g(\omega) = \frac{1 - a^2}{1 + 2a\cos\omega + a^2}. \tag{15}$$

In order to achieve the ideal octave spacing of notches specified by the illusion, $\tau_g(\omega) \propto 2^{-\omega}$ would be required for each allpass section. This can be approximated by setting a to a moderate negative value, for example $a = -0.5$, as shown in Figure 13. A better fit could be produced by optimizing the individual coefficients of the first-order allpass filters, or by applying a more generalized method of fitting the desired group-delay curve [38].

The output of this system when used to process an input of white noise is shown in Figure 14. Note that there are still 16 notches, as in Figure 11. Compared to the version of the system with linearly spaced notches, this system produces a much more convincing Sherpard-Risset glissando effect. Some cyclic-sounding behaviour is audible in the very low frequencies, which can be explained by the flattening of the group-delay curve in this region (see Figure 13).

5. CONCLUSIONS

This paper has discussed three different ways to implement infinite phaser and flanging effects, which are called barberpole effects. These effects can be implemented in real time and incorporated as parts of an audio processing environment.

The first proposed method can be interpreted as an inverted version of the Shepard-Risset auditory illusion. Instead of producing an infinitely sweeping tone, it produces infinitely one-way sweeping notches using time-varying notch filters. This processing technique can be used as a digital audio effect for rich-sounding audio material, such as noisy sounds, drums loops, or distorted guitars, to replace the traditional back-and-forth-going phaser.

The second method is a novel dual flanger structure, which is based on a pair of cascaded feedforward comb filters with synchronized modulation controls. Sawtooth waveforms are used to control the length of the delay lines, making the displacement of notches unidirectional. The LFOs used to modulate the second delay line and its gain are in a 90° phase shift in relation to those of

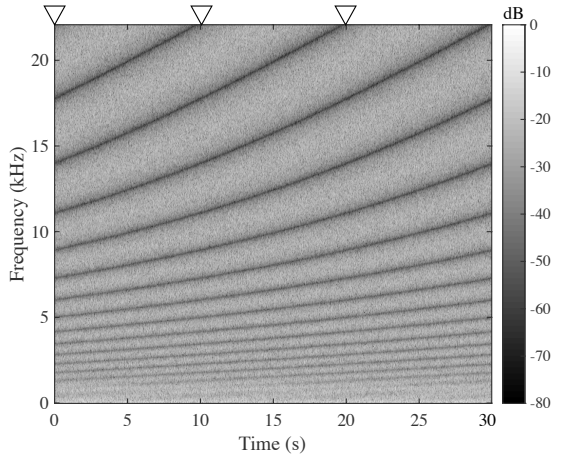


Figure 14: *Spectrogram of white noise processed by the structure given in Figure 12. The number of cascaded allpass filters is* $D = 32$*, the allpass coefficient* $a = -0.5$*, and the amount of frequency shift is 0.1 Hz.*

the first delay line. This results in a smooth cross-fading between delay lines, minimizing the audibility of any abrupt changes in delay lengths. This extension can be called a barberpole flanging effect.

The third method is a digital version of the first barberpole-like phasing effects described by Harald Bode, with a modification proposed to make the notches more closely fit the distribution of tones in a Shepard-Risset illusion. As with the other methods, the effect sounds best with low modulation speeds and with rich spectral content as the input.

In general terms, there are very subtle differences between the proposed techniques. The three of them effectively recreate the illusion of endlessly moving notches. The cyclic nature of the effect is perhaps best hidden in the first implementation, where the notches on the low and high ends of the spectrum are not as audible as those in the middle region. However, this technique is severely restricted to signals with a quasi-flat spectrum. The other proposed techniques are far more versatile, they are not as restricted to a certain type of input. Additionally, their implementation is considerably more economic. For instance, the dual flanger approach can implement a high number of notches without increased complexity. Overall, the biggest limitation of the dual flanger and SSB-modulation methods comes from the fact that, if not properly tuned, their cyclic nature can be easily given away. Audio examples related to all three techniques described in this paper are available at the accompanying website `http://research.spa.aalto.fi/publications/papers/dafx15-barberpole`.

6. ACKNOWLEDGMENTS

The authors would like to thank Dr. Stefan Bilbao and Dr. José Antonio Belloch for their helpful comments during the elaboration of this paper.

7. REFERENCES

[1] R. N. Shepard, "Circularity in judgments of relative pitch," *J. Acoust. Soc. Amer.*, vol. 36, no. 12, pp. 2346–2353, Dec. 1964.

[2] R. N. Shepard, "Demonstrations of circular components of pitch," *J. Audio Eng. Soc.*, vol. 31, no. 9, pp. 641–649, Sept. 1983.

[3] J.-C. Risset, "Pitch control and pitch paradoxes demonstrated with computer-synthesized sounds (abstract)," in *77th Meeting of the Acoustical Society of America*, 1969.

[4] J.-C. Risset, *Current Directions in Computer Music Research*, chapter 'Paradoxical sounds', pp. 149–158, M. V. Mathews and J. R. Pierce (eds.), The MIT Press, Cambridge, MA, 1989.

[5] D. Deutsch, "Auditory illusions, handedness, and the spatial environment," *J. Audio Eng. Soc.*, vol. 31, no. 9, pp. 607–618, Sept. 1983.

[6] D. Stowell, "Scheduling and composing with Risset eternal accelerando rhythms," in *Proc. Int. Computer Music Conf.*, Huddersfield, UK, July 2011.

[7] B. Bartlett, "A scientific explanation of phasing (flanging)," *J. Audio Eng. Soc.*, vol. 18, no. 6, pp. 674–675, Dec. 1970.

[8] W. M. Hartmann, "Flanging and phasers," *J. Audio Eng. Soc.*, vol. 26, no. 6, pp. 439–443, Jun. 1978.

[9] J. O. Smith, *Physical Audio Signal Processing*, W3K Publishing, 2010, online book. See subsection 'Flanging'.

[10] J. D. Reiss and A. P. McPherson, *Audio Effects: Theory, Implementation and Application*, CRC Press, 2015, See Ch. 2 'Delay Line Effects'.

[11] M. M. Wanderley and P. Depalle, "Gesturally controlled digital audio effects," in *Proc. COST-G6 Conf. Digital Audio Effects (DAFx-01)*, Limerick, Ireland, Dec. 2001, pp. 165–169.

[12] M. M. Wanderley and P. Depalle, "Gestural control of sound synthesis," *Proceedings of the IEEE*, vol. 92, no. 4, pp. 632–644, Apr. 2004.

[13] M. L. Beigel, "A digital "phase shifter" for musical applications, using the Bell Labs (Alles-Fischer) digital filter module," *J. Audio Eng. Soc.*, vol. 27, no. 9, pp. 673–676, Sept. 1979.

[14] J. O. Smith, "An allpass approach to digital phasing and flanging," in *Proc. Int. Computer Music Conf.*, Paris, France, Oct. 1984, pp. 103–109.

[15] V. Välimäki, S. Bilbao, J. O. Smith, J. S. Abel, J. Pakarinen, and D. Berners, "Virtual analog effects," in *DAFX: Digital Audio Effects*, U. Zölzer, Ed., pp. 473–522. Wiley, Chichester, UK, second edition, 2011.

[16] H. Bode, "History of electronic sound modification," *J. Audio Eng. Soc.*, vol. 32, no. 10, pp. 730–739, Oct. 1984.

[17] O. Larkin, "Endless Series," Software and sound examples available at http://www.olilarkin.co.uk/index.php?p=eseries, accessed June 4, 2015.

[18] C. Budde, "Barberpole Flanger," VST effect plug-in for Windows available at http://www.kvraudio.com/product/barberpole_flanger_by_christian_budde, accessed June 4, 2015.

[19] A. Huovilainen, "Enhanced digital models for analog modulation effects," in *Proc. Int. Conf. Digital Audio Effects (DAFx-05)*, Madrid, Spain, Sept. 2005, pp. 155–160.

[20] C. Raffel and J. Smith, "Practical modeling of bucket-brigade device circuits," in *Proc. Int. Conf. Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 2010.

[21] F. Eichas, M. Fink, M. Holters, and U. Zölzer, "Physical modeling of the MXR Phase 90 guitar effect pedal," in *Proc. Int. Conf. Digital Audio Effects (DAFx-14)*, Erlangen, Germany, Sept. 2014, pp. 153–166.

[22] T. Wilmering, G. Fazekas, and M. B. Sandler, "Audio effect classication based on auditory perceptual attributes," in *Proc. AES 135th Convention*, New York, USA, Oct. 2013.

[23] M. Stein, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic detection of audio effects in guitar and bass recordings," in *Proc. AES 128th Convention*, London, UK, May 2010.

[24] S. J. Orfanidis, *Introduction to Signal Processing*, chapter 'IIR Digital Filter Design', Prentice Hall International Editions, Englewood Cliffs, NJ, USA, 1996.

[25] V. Välimäki and T. I. Laakso, "Suppression of transients in variable recursive digital filters with a novel and efficient cancellation method," *IEEE Trans. Signal Process.*, vol. 46, no. 12, pp. 3408–3414, Dec. 1998.

[26] V. Välimäki, "Discrete-time synthesis of the sawtooth waveform with reduced aliasing," *IEEE Signal Process. Lett.*, vol. 12, no. 3, pp. 214–217, Mar. 2005.

[27] H.-M. Lehtonen, V. Välimäki, and T. I. Laakso, "Canceling and selecting partials from musical tones using fractional-delay filters," *Comp. Music J.*, vol. 32, no. 2, pp. 43–56, Summer 2008.

[28] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay—Tools for fractional delay filter design," *IEEE Signal Process. Mag.*, vol. 13, no. 1, pp. 30–60, Jan. 1996.

[29] D.E. Norgaard, "The phase-shift method of single-sideband signal generation," *Proc. of the IRE*, vol. 44, no. 12, pp. 1718–1735, Dec 1956.

[30] L. R. Rabiner and R. W. Schafer, "On the behavior of minimax FIR digital Hilbert transformers," *Bell Syst. Tech. J.*, vol. 53, no. 2, pp. 363–390, 1974.

[31] S. Disch and U. Zölzer, "Modulation and delay line based digital audio effects," in *Proc. Second COST G-6 Workshop on Digital Audio Effects (DAFx-99)*, Trondheim, Norway, Dec. 1999, pp. 5–8.

[32] S. Wardle, "A Hilbert-transformer frequency shifter for audio," in *Proc. First COST G-6 Workshop on Digital Audio Effects (DAFx-98)*, Barcelona, Spain, Nov. 1998, pp. 25–29.

[33] R. Ansari, "IIR discrete-time Hilbert transformers," *IEEE Trans. Acoustics, Speech, Signal Process.*, vol. 35, no. 8, pp. 1116–1119, Aug. 1987.

[34] V. Välimäki, J. S. Abel, and J. O. Smith, "Spectral delay filters," *J. Audio Eng. Soc.*, vol. 57, no. 7–8, pp. 521–531, Jul. 2009.

[35] V. Välimäki, J. Parker, and J. S. Abel, "Parametric spring reverberation effect," *J. Audio Eng. Soc.*, vol. 58, no. 7–8, pp. 547–562, Jul. 2010.

[36] J. Parker, "Efficient dispersion generation structures for spring reverb emulation," *EURASIP J. Advances in Signal Processing*, vol. 2011, no. 646134, pp. 521–531, Mar. 2011.

[37] J. Pekonen, T. Pihlajamäki, and V. Välimäki, "Computationally efficient Hammond organ synthesis," in *Proc. Int. Conf. Digital Audio Effects (DAFx-11)*, Paris, France, Sept. 2011, pp. 19–22.

[38] J. S. Abel and J. O. Smith, "Robust design of very high-order allpass dispersion filters," in *Proc. 9th Int. Conf. Digital Audio Effects (DAFx-06)*, Montreal, Canada, Sept. 2006, pp. 13–18.

DISTORTION AND PITCH PROCESSING USING A MODAL REVERBERATOR ARCHITECTURE

Jonathan S. Abel, Kurt James Werner

Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University, Stanford, CA 94305 USA
{abel,kwerner}@ccrma.stanford.edu

ABSTRACT

A reverberator based on a room response modal analysis is adapted to produce distortion, pitch and time manipulation effects, as well as gated and iterated reverberation. The so-called "modal reverberator" is a parallel collection of resonant filters, with resonance frequencies and dampings tuned to the modal frequencies and decay times of the space or object being simulated. Here, the resonant filters are implemented as cascades of heterodyning, smoothing, and modulation steps, forming a type of analysis/synthesis architecture. By applying memoryless nonlinearities to the modulating sinusoids, distortion effects are produced, including distortion without intermodulation products. By using different frequencies for the heterodyning and associated modulation operations, pitch manipulation effects are generated, including pitch shifting and spectral "inversion." By resampling the smoothing filter output, the signal time axis is stretched without introducing pitch changes. As these effects are integrated into a reverberator architecture, reverberation controls such as decay time can be used produce novel effects having some of the sonic characteristics of reverberation.

1. INTRODUCTION

The "modal reverberator" algorithm [1, 2] proposed a unification of perspectives on room reverberation analysis with the goals of synthetic reverberation. While room reverberation has long been analyzed from the viewpoint of modal analysis [3, pg. 172, ff.], [4, pg. 576, ff.], artificial reverberation is typically synthesized using structures such as delay networks or convolution which attempt to reproduce time domain features of the room response [5]. By contrast, the modal reverberator implements room modes directly as the sum of parallel resonant filters.

The parallel architecture of the modal reverberator provides explicit, interactive control over the parameters of each mode, allowing accurate modeling of acoustic spaces, as well as movement within them and morphing among them. Here, we extend this structure to allow manipulation of the mode responses in ways that lead to alternative implementations of audio effects such as distortion and pitch shifting, and to novel effects which integrate reverberation into nonlinear processes.

Schroeder and Logan's seminal 1961 article " 'Colorless' Artificial Reverberation" [6] introduced the comb and allpass filters as powerful building blocks for designing artificial reverberators and launched a flurry of research and commercial activity in the field. Their perspective led directly to artificial reverberation algorithms using digital waveguide and physical modeling approaches [7, 8], multi-feedback unitary systems [9], and the Feedback Delay Network (FDN) framework [9–12]. This family of methods, based on networks of delay lines and filters, is still considered state of the art today [5].

Although Schroeder and Logan pioneered the use of delay lines in mimicking a room's response, Schroeder was an experienced researcher of the modal properties of rooms [13, 14] and they situated their work in the frequency domain. Their article [6] opens: "A room can be characterized by its normal modes of vibration." This perspective is standard in room acoustics [4, 15] and musical instrument modeling [16], and in fact modal thinking about room acoustics has its roots in antiquity. Blesser traces a history of interactions between space, music, and culture [17], including a history of *acoustical vases*, Helmholtz resonators set up in amphitheaters to change their reverberant properties—the tradition was well-established when Vitruvias reported on it in 30 B.C. [18] and continued into medieval times [19].

In artificial reverberation research, ironically, the modal perspective has never quite risen above a theoretical framing device; mode responses are not implemented directly. Blesser gives a thorough review of eigentone statistics [17] before concluding: "The eigentone model is much less useful than the random echo model for many reasons."

In fact, in FDNs and related frameworks, audible modes can be considered undesirable artifacts. Moorer reports on "annoying ringing" in Schroeder-style cascaded allpass reverberators [20]. Griesenger [21] describes Stautner's approach (which is widely used) to controlling "unpleasant resonant behavior": time-varying delays and mild feedback [10]. Jot writes of "unnatural resonances" which sound "metallic" [11, 22]. In the discourse of FDN limitations, late-field resonances are considered to be artifacts much like limit cycles [20, 23].

In the development of artificial reverberation algorithms, many techniques created out of practical necessity are repurposed and exaggerated to artistic effect. For instance, consider that early work on feedback systems used frequency shifting to avoid howling in public address systems [24, 25]. Artificial reverberation algorithms such as Sean Costello's ValhallaShimmer [26] and Vesa Norilo's "Vectored Time Variant Comb Filter" [27] have repurposed this practical tool as an artistic effect. Consider also the concept of modulating delay lines in FDNs to break up patterns of regular echos, a technique well represented in the literature [10, 12, 21] and found in commercial products such as the Lexicon 224 [28, 29]. Now, algorithms may use more extreme versions of delay-line modulation to achieve versions of, e.g., chorus and flangers within feedback loops [27]. Embedded "Slinky" paths in FDNs can be considered an exaggeration of natural allpass characteristics along a propagation path [30].

In this work, we propose extensions to the modal reverberator algorithm [1, 2] that parallel this development, emphasizing the al-

gorithm's potential as a platform for new musical effects. Like extended FDNs, a "modal effects processor" framework yields classes of novel musical effects. In §2 the modal reverberator is reviewed. In §3, classes of modal effects are presented, including gating and envelope processing (§3.1), time stretching (§3.2), pitch manipulation (§3.3), and distortion processing (§3.4).

2. MODAL REVERBERATOR

Acoustic spaces and vibrating objects have long been analyzed in terms of their normal modes [4, 16]. The impulse response $h(t)$ between a pair of points in the system may be expressed as the linear combination of mode responses,

$$h(t) = \sum_{m=1}^{M} h_m(t), \tag{1}$$

where the system has M modes, with the mth mode response denoted by $h_m(t)$, t being the discrete time sample index. The system output $y(t)$ in response to an input $x(t)$, the convolution

$$y(t) = h(t) * x(t), \tag{2}$$

is then seen to be the sum of mode outputs

$$y(t) = \sum_{m=1}^{M} y_m(t), \quad y_m(t) = h_m(t) * x(t), \tag{3}$$

where the mth mode output $y_m(t)$ is the mth mode response convolved with the input. The modal reverberator simply implements this parallel combination of mode responses (3), as shown in Fig. 1.

In general, mode responses $h_m(t)$ are complex exponentials, each characterized by a mode frequency ω_m, mode damping α_m and complex mode amplitude γ_m,

$$h_m(t) = \gamma_m \exp\{(j\omega_m - \alpha_m)t\}. \tag{4}$$

The choice of complex, rather than real, mode responses is made here for clarity of presentation and to suggest the implementation structures described below. A real response would be formed by a combinations of conjugate responses; here a proportional result is formed by taking the real part of each complex mode response. A stereo effect can by obtained by taking the imaginary part as a second channel.

The mode frequencies and dampings are properties of the room or object. They describe, respectively, the mode oscillation frequencies and decay times. The mode amplitudes are determined by the sound source and listener positions (driver and pick-up positions for an electro-mechanical device), according to the mode spatial patterns.

Note that even for short reverberation times of just a few hundred milliseconds, the mode responses $h_m(t)$ are very resonant, and last many thousands of samples at typical audio sampling rates. In implementing the mode filters, therefore, numerically stable methods must be used. One such method is the phasor filter [31, 32], in which each mode filter is implemented as a complex first-order update,

$$y_m(t) = \gamma_m x(t) + e^{(j\omega_m - \alpha_m)} y_m(t-1). \tag{5}$$

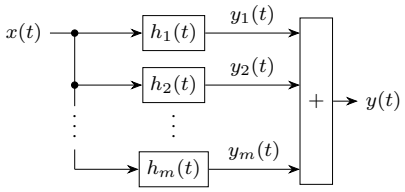


Figure 1: *Modal Reverberator Architecture. The modal reverberator is the parallel combination of resonant filters matched to the modes of the system.*

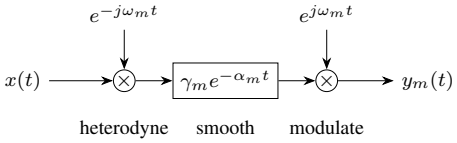


Figure 2: *Mode Response Implementation. The mode response may be implemented as a cascade of heterodyning, smoothing and modulation operations.*

Another approach is to rearrange the mode response convolution,

$$y_m(t) = \sum_{\tau} \gamma_m e^{(j\omega_m - \alpha_m)(t-\tau)} x(\tau) \tag{6}$$

$$= e^{j\omega_m t} \sum_{\tau} \gamma_m e^{-\alpha_m (t-\tau)} \left[e^{-j\omega_m \tau} x(\tau) \right]. \tag{7}$$

Here, the mode filtering is implemented by heterodyning the input signal to dc to form a baseband response, smoothing this baseband response by convolution with an exponential, and modulating the result back to the original mode frequency,

$$y_m(t) = e^{j\omega_m t} \cdot \left(\gamma_m e^{-\alpha_m t} * \left[e^{-j\omega_m t} x(t) \right] \right). \tag{8}$$

This process is shown in Fig. 2. The heterodyning and modulation steps implement the mode frequency, and the smoothing filter generates the mode envelope, in this case an exponential decay.

Using this architecture, rooms and objects may be simulated by tuning the filter resonant frequencies and dampings to the corresponding room or object mode frequencies and decay times. The parallel structure allows the mode parameters to be separately adjusted, while the updates (5) or (8) provide interactive parameter control with no computational latency.

Three design approaches are suggested in [2] for setting the mode parameters: behavioral, analytical, and perceptual. The behavioral approach fits mode parameters to system measurements. The analytic approach derives mode parameters from the physics of the system. The perceptual approach selects mode parameters according to desired wet response equalization and, say, early and late decay times. An example design is shown in Fig. 3, in which a modal reverberator architecture is fit to the measured impulse response of a classroom. For this design 1605 modes were used, and the modal system response was a close perceptual match to that of the measurement.

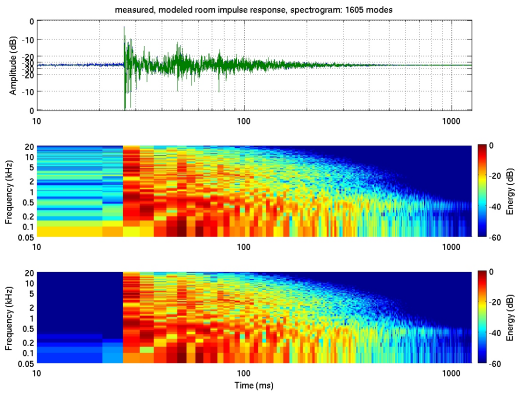
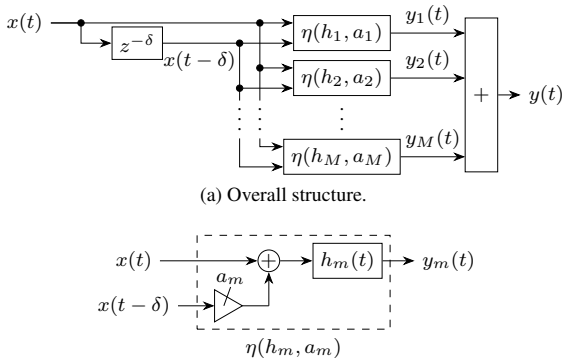


Figure 3: *Example Modal Reverberator Design. Measured and modeled room impulse responses for a roughly 10 m×12 m×5 m classroom, Room K217 at The Knoll, Stanford University, are overlaid (top), and the corresponding spectrograms shown (middle, modeled; bottom, measured).*



(a) Overall structure.

(b) Detail of the mth band.

Figure 4: *Modal Gated Reverberator Signal Flow Architecture, including (a) overall structure and (b) detail of an individual band.*

3. MODAL EFFECTS PROCESSOR

The modal reverberator lends itself to effects processing through its parallel architecture and dense, narrowband mode responses. In the following, we present four categories of effects, all provided by manipulating different blocks of the mode response processor shown in Fig. 2:

- gating and envelope processing,
- time stretching,
- pitch manipulation, and
- distortion processing.

Reverberation envelope effects such as gating and iterated convolution may be achieved by manipulating the smoothing filter response. Time stretching without pitch shifting is possible by resampling the smoothing filter output. Pitch manipulation effects such as pitch shifting and spectral "inversion" are available by using different sinusoid frequencies for the heterodyning and modulation steps. Finally, distortion effects may be generated by distorting or substituting for the modulation sinusoid waveform.

Since these effects are integrated into a reverberation architecture, their sonics are different than their standard counterparts when the mode decay times are longer than a few hundred milliseconds. The result is a unique effect with sonic qualities of both the standard effect and reverberation.

3.1. Reverberation Envelope Effects

We first describe reverberation envelope effects.

Gated reverberation is a reverberation effect in which the reverberation response onset is rapidly forced to zero after a short period of time, say 250 ms. The effect was popularized by Phil Collins in the 1980s (e.g. [33]). One approach, used by the AMS RMX-16, one of the first digital reverberators and popular in the 1980's [34], implements a system impulse response which decays very rapidly after a given point in time.

In the modal architecture discussed here, the individual mode responses are decaying exponentials, and may be "switched off" after a specified delay using a truncated IIR (TIIR) technique [35]. A first-order mode filter impulse response may be truncated after a given delay by subtracting from its input an appropriately scaled, delayed version of the input. Stated mathematically, the mth mode filter impulse response

$$h_m(t) = e^{(j\omega_m - \alpha_m)t} \tag{9}$$

can be made zero starting at a delay δ by replacing the input $x(t)$ with

$$x(t) - e^{(j\omega_m - \alpha_m)\delta} x(t-\delta). \tag{10}$$

This is implemented in the signal flow architecture shown in Fig. 4. Note that the input signal delay is implemented just once, outside the modal reverberator structure, and the scaling, $\exp\{(j\omega_m - \alpha_m)\delta\}$, which varies from mode to mode, is implemented locally (Fig. 4b).

As an example, consider the dry and processed guitar track snippets shown in Fig. 5. Here, the reverberator employed 2048 modes, with decay times ranging from 2500 ms in the mid frequencies to 200 ms in the high frequencies. The gate time δ was set to 290 ms. Transients in the guitar track reverberated for only a short time before truncation.

We suggest two variations: An interesting "gated cathedral" sound results when a long reverberation time is used, and the response isn't fully truncated. This is accomplished by slightly reducing the magnitude of the scale factor, e.g., to $0.95\exp\{(j\omega_m - \alpha_m)\delta\}$. Another artistic effect forms groups of modes, with different groups having different gate times. An example impulse response of such a system is shown in Fig. 6.

This TIIR approach may be used with higher order mode response filters. For instance, repeated pole filters having N poles and impulse response onsets roughly proportional to t^{N-1} can be truncated to generate a "reverse reverberation" effect. To implement such filters, the structure of Fig. 4 can be augmented with additional delay lines having lengths equal to integer multiples of δ, up to $N\delta$. Finally, reverse reverberation can also be implemented using the TIIR approach with a growing exponential mode envelope.

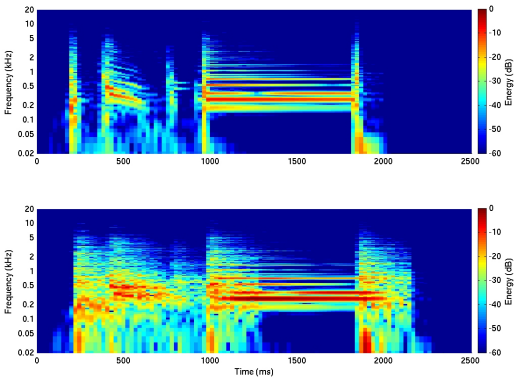


Figure 5: *Modal Gated Reverberator Processed Guitar Track. Dry (top) and processed (bottom) guitar track snippets. Note the truncation of reverberated transients.*

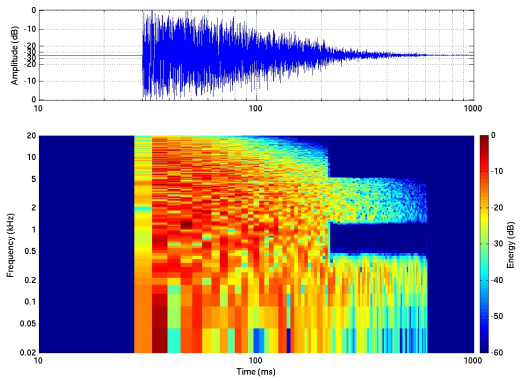


Figure 6: *Modal Gated Reverberator Impulse Response. The impulse response (top) and corresponding spectrogram (bottom) are shown for a modal gated reverberator with a gate time of 180 ms for certain frequency bands, and 580 ms for others. Logarithmic time axes are used.*

Another reverberation envelope effect is iterated reverberation, the repeated application of a reverberant impulse response, inspired by Alvin Lucier's piece "I am sitting in a room" [36]. Since the mode responses are orthogonal, the order-N iterated convolution of the system response $h(t)$,

$$h^{*N}(t) = \underbrace{h(t) * h(t) * \cdots * h(t)}_{N \text{ responses}}, \tag{11}$$

is the sum of the mode response iterated convolutions,

$$h^{*N}(t) = \sum_{m=1}^{M} h_m^{*N}(t). \tag{12}$$

The mode response iterated convolutions may be implemented using the approach of (8), in which the heterodyning and modulation operations are left unchanged and the mode envelope filter is cascaded with itself (i.e., iterated) N times. Doing so produces a mode envelope proportional to $t^{N-1}\exp\{-\alpha_m t\}$, which provides a delayed onset of late field energy, peaking at a time $(N-1)/\alpha_m$.

Additional reverberation envelopes include delayed onset and two-stage decays, as would be appropriate for modeling coupled spaces such as a box in an opera hall, and as described in [37, 38]. These can be implemented in the context of the modal reverberator structure by design of the mode envelope filter in a manner similar to that described in [38]. Alternatively, a two-stage decay may be implemented by having some modes take on a large amplitude and decay quickly while other modes have a smaller amplitude and decay slowly.

3.2. Time Stretching Effects

In the modal reverberator structure of Fig. 1 and Fig. 2, a reverberated signal is constructed from its mode responses, each of which is generated by applying its mode envelope to a sinusoid at its mode frequency. If the mode decay times are short (less than, say, a couple hundred milliseconds) and the mode energies are inversely proportional to the frequency density of nearby modes, then the processed audio will sound much like the input. In this case, if all of the mode envelopes are resampled to a stretched time axis, then the resulting signal will be a time-stretched version of the input.

As an example, Fig. 7 shows spectrograms of the dry guitar track of Fig. 4 time stretched in this way by factors of 0.25, 1, and 4. Note that the spectra are similar under the appropriate time axis scalings. While the timbers of the time-stretched signals match well, there are differences between the dry input and the slightly reverberated signal generated without resampling the mode envelope. Mainly, transients are less crisp.

One other artifact appears, a subtle beating or tremolo heard during sustained notes. Using a second-order smoothing filter, say the first-order filter repeated, effectively eliminates the problem. The examples of Fig. 7 were generated with such a second-order mode envelope. Finally, as the smoothing filter output has relatively low bandwidth, linear interpolation is expected to be sufficient for resampling, and was used to generate the time stretched signals of Fig. 7.

The time stretching can vary with time so as to compress or expand the time axis of different sections of the signal by different amounts. In addition, the parallel structure makes it simple to vary the time axis modification over frequency, for instance, having low frequencies time expanded and high frequencies time compressed.

3.3. Pitch Manipulation Effects

Recall that the mode response can be thought of as the cascade of heterodyning, smoothing, and modulation operations, as shown in Fig. 1. Now consider replacing each modulation frequency ω_m with one shifted by σ half steps,

$$\nu_m = 2^{\sigma/12}\omega_m, \tag{13}$$

as seen in Fig. 8. Doing so will shift the pitch of the output relative to the input. This is illustrated in Fig. 9, in which a guitar track is shifted by -2, 0, and 2 octaves. Note that the shifted spectrograms

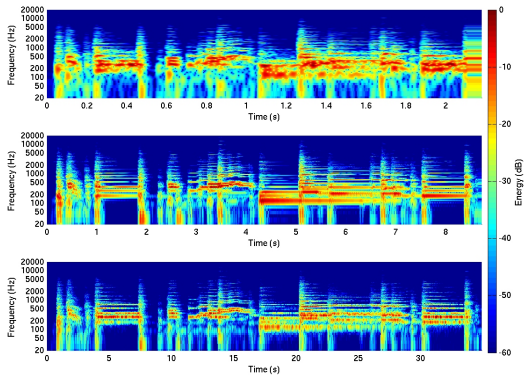


Figure 7: *Time Stretched Guitar Track. Spectrograms for a guitar track, processed using a 2400-mode processor with randomly selected, exponentially distributed mode frequencies in the range 20 Hz to 20 kHz and having 200 ms decay times, is shown with the mode envelope filter output sampled at rates of* 4 *(top),* 1 *(middle), and* $1/4$ *(bottom) of the system sampling rate of 48 kHz. Note the different time axes.*

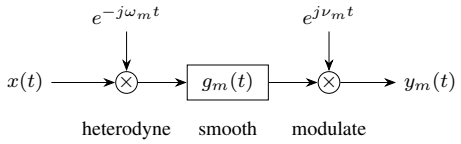


Figure 8: *Pitch Shifting Mode Response Implementation.*

are well matched, save the constant frequency resolution of the spectrogram.

As in the case of the time stretching effect above, we suggest using higher order mode envelope filters to eliminate tremolo-type artifacts. The signals shown in Fig. 9 were prepared with three iterations of the first-order response (i.e., four-pole filters) (4), though one iteration works well. There is a trade-off involving the mode decay time: With a short decay time, the mode envelope filter has a wide bandwidth, resulting in beating between adjacent modes. With a long decay time, the beating is eliminated, but the output will take on a reverberant quality, which might be unwanted.

In the example of Fig. 9, we used 200 ms decay times, and 2400 exponentially distributed mode frequencies. We heard little if any difference between deterministically and randomly generated mode frequencies. We used random mode phases $\angle\gamma_m$, uniformly distributed on the interval $[0, 2\pi)$, so that the system response to a pulse would lack structure, and therefore reduce temporal artifacts. Finally, note that modes shifted up or down in frequency outside the audio band need not be computed.

Since the modulation is computed on a sample-by-sample basis, the pitch shift may be changed on a sample-by-sample basis. Since the modes are independent, different pitch shifts may be applied to different modes. Fig. 10 shows a pitch shift which drifts upward over time, and for input frequencies above about 200 Hz, develops a vibrato having an increasing rate. Also shown in Fig. 10

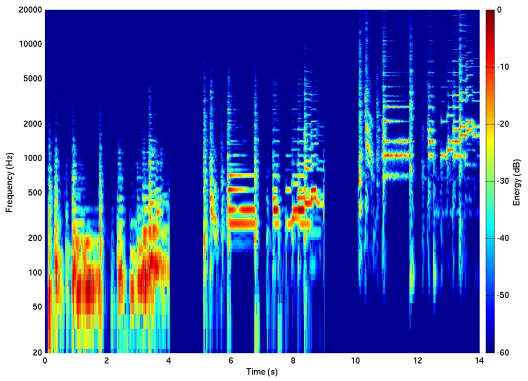


Figure 9: *Pitch Shift Example. Spectrograms are shown for a guitar track processed with a modal reverberator having 2400 modes and a 200 ms decay time, and pitch shifted by factors of* 4 *(top),* 1 *(middle), and* $1/4$ *(bottom).*

is a signal processed using the same pitch trajectory, but with a 1.5 s-long reverberation in the mid frequencies. In this case the mode dampings were kept constant, and the reverberation time as a function of frequency $T_{60}(\omega)$ moves with the pitch shift. To keep the reverberation time independent of the pitch shift, the mode dampings could be set according to the instantaneous pitch shifted frequencies, $T_{60}(2^{\sigma_m(t)/12}\omega_m)$.

Other pitch effects may be produced by shifting different mode frequencies by different amounts. A number of strategies for generating the modulation frequencies have been tried, including permuting the mode frequencies, generating random frequencies, and controlling the distance from the heterodyne frequencies to a quantized set of frequencies.

Another choice is a spectral "inversion," formed by inverting the mode frequencies about a center frequency, ω_c, and applying a frequency shift,

$$\nu_m = 2^{\sigma/12}\frac{\omega_c^2}{\omega_m}. \tag{14}$$

The effect creates different harmonic relationships among the partials present, as seen in the example of Fig. 11.

3.4. Distortion Effects

The modal reverberator architecture can be integrated with a distortion process by distorting each mode response before mixing to form the output, as shown in Fig. 12. Alternatively, groups of modes may be mixed and distorted together, as shown in Fig. 13.

With the modes individually distorted, or with the mode frequencies in a given group having harmonic relationships, the distortion produced will be free of dissonances from intermodulation products. This distortion has a different sonic character than is typical, producing a distorted sound while maintaining the harmonic structure of the dry track. There are a number of pop songs, including "Something About You" by Boston [39] and "God Save the Queen" by Queen [40], which use this type of effect with guitar, and presumably achieved by multitracking, building up chord sequences from single-note lines of music. (Such an effect could

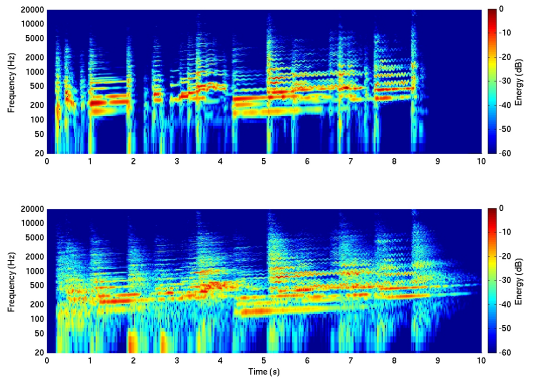


Figure 10: *Pitch Drift Example. The spectrogram of the guitar track of Fig. 9 processed according to a drifting, modulating pitch shift is shown (top), along with a version processed with the same pitch shift trajectory, but with a reverberation time of about 1.5 s (bottom). Note that since the mode dampings are left constant, the reverberation time drifts in frequency with the pitch shift.*

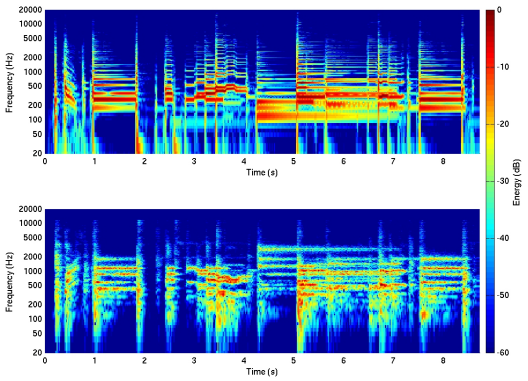


Figure 11: *Spectral Inversion Example. Spectrograms are shown for a dry guitar track (top) and a processed version (bottom), in which mode heterodyning and modulation frequencies are inverses of each other about 440 Hz.*

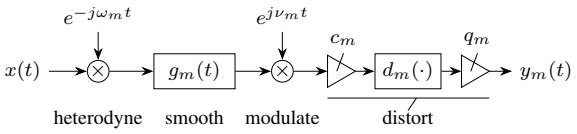


Figure 12: *Distortion Effect on One Mode.*

also be achieved using a hexaphonic pick-up, and having a separate distortion process on each string's output.) Modal distortion can be considered a further exaggeration of this method—rather than distorting individual notes, each mode response produced by a signal may be distorted independently.

Though it may be computationally costly to distort modes individually or in narrow-bandwidth groups, the need for upsampling to avoid aliasing may be eliminated, thus reducing the cost. To do so, consider a distortion function $d_m(w)$ expressed as a power series in w. On a per-mode or per-mode-group basis, $d_m(w)$ can be designed to produce output without aliasing by implementing only those power series terms which would produce harmonic distortion below the Nyquist limit. For instance, for modes having frequencies above half the Nyquist limit, only the constant and linear terms would be included; for modes having frequencies between half and one third the Nyquist limit, only the constant, linear, and quadratic terms would be included; and so on. (For a discussion of distortion and aliasing, see [41].)

An example distortion process output is shown in Fig. 14, with the guitar track used previously distorted in the presence of both short and longer reverberation times.

Another approach is to substitute periodic waveforms, say sawtooth or wavetable-based waveforms, for the sinusoidal modulators $\exp\{j\nu_m t\}$. Unlike the distortion from memoryless nonlinearities, this approach provides distortion that is independent of signal amplitude.

4. SUMMARY

In this work, we described extensions to the modal reverberator algorithm that produce audio effects in four categories: reverberation envelope control, time stretching, pitch manipulation, and distortion processing. This was achieved by implementing the resonant mode filters as cascades of heterodyning, smoothing, and modulation steps, and manipulating aspects of the smoothing and modulation operations.

The modal processor signal flow architecture is in some ways similar to that of a phase vocoder [42, 43], having a parallel bank of analysis, processing and synthesis operations. A key difference, however, is that the modal processor's filterbank center frequencies and bandwidths (i.e., dampings) are often chosen according to a desired acoustic space, rather than as the filters associated with a given Fourier transform length and window. This allows modal processor effects such as pitch shifting and distortion to incorporate reverberation, while providing relatively artifact-free effects when the mode decay times are short.

The parallel nature of the modal architecture allows different effects or no effect to be applied on a per mode or per mode group basis. The sample-by-sample processing allows continuous control of all effects parameters without latency and with no blocking artifacts.

Acknowledgement

The authors would like to thank Eoin Callery of the Department of Music at Stanford University for discussions and feedback on using the modal effects processors described above for music composition and production.

5. REFERENCES

[1] J. S. Abel, S. Coffin, and K. S. Spratt, "A modal architecture

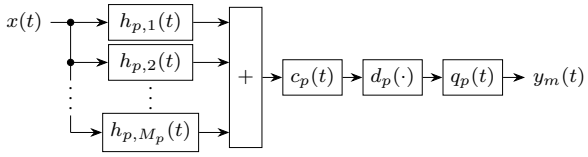


Figure 13: *Distortion Effect on a Group of Modes.*

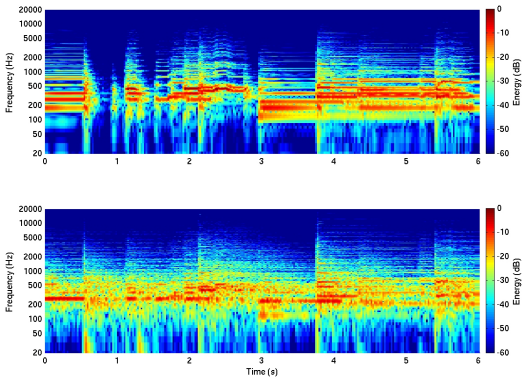


Figure 14: *Distortion Example. Spectrograms are shown for the guitar track used previously, processed with a modal distortion having decay times of 200 ms (top) and about 2 s (bottom).*

for artificial reverberation," *The Journal of the Acoustical Society of America*, vol. 134, no. 5, pp. 4220, 2013.

[2] J. S. Abel, S. Coffin, and K. Spratt, "A modal architecture for artificial reverberation with application to room acoustics modeling," in *Audio Engineering Society Convention*, Los Angeles, CA, October 9–12 2014, vol. 137.

[3] A. H. Benade, *Fundamentals of Musical Acoustics*, Oxford University Press, 1976.

[4] P. M. Morse and K. U. Ingard, *Theoretical Acoustics*, Princeton University Press, 1987.

[5] V. Välimäki, J. D. Parker, L. Savioja, J. O. Smith III, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1421–1448, July 2012.

[6] M. R. Schroeder and B. F. Logan, "'Colorless' artificial reverberation," *Journal of the Audio Engineering Society*, vol. 9, no. 3, pp. 192–197, July 1961.

[7] J. O. Smith III, "A new approach to digital reverberation using closed waveguide networks," in *International Computer Music Conference*, Burnaby, B.C., Canada, August 22 1985, pp. 47–53, Also available as [8].

[8] J. O. Smith III, "A new approach to digital reverberation using closed waveguide networks," CCRMA Technical Report STAN-M-31, CCRMA, Stanford University, Stanford, CA, 1985.

[9] M. A. Gerzon, "Unitary (energy-preserving) multichannel networks with feedback," *Electronics Letters*, vol. 12, no. 11, pp. 278–279, May 1976.

[10] J. Stautner and M. Puckette, "Designing multi-channel reverberators," *Computer Music Journal*, vol. 6, no. 1, pp. 52–65, Spring 1982.

[11] J.-M. Jot and A. Chaigne, "Digital delay networks for designing artificial reverberators," in *Audio Engineering Society Convention*, Paris, France, February 19–22 1991, vol. 90.

[12] J. Frenette, "Reducing artificial reverberation algorithm requirements using time-variant feedback delay networks," M. S. thesis, University of Miami, Coral Gable, FL, December 2000 2000.

[13] M. Schröder, "Eigenfrequenzstatistik und Anregungsstatistik in Räumen," *Acta Acustica united with Acustica*, vol. 4, no. Supplement 1, pp. 456–468, 1954.

[14] M. Schröder, "Die statistischen Parameter der Frequenzkurven von großen Räumen," *Acta Acustica united with Acustica*, vol. 4, pp. 594–600, 1954.

[15] P. M. Morse and R. H. Bolt, "Sound waves in rooms," *Reviews of Modern Physics*, vol. 16, no. 2, pp. 69, April 1944.

[16] N. H. Fletcher and T. D. Rossing, *Physics of Musical Instruments*, Springer, 2nd edition, 2010.

[17] B. A. Blesser, "An interdisciplinary synthesis of reverberation viewpoints," *Journal of the Audio Engineering Society*, vol. 49, no. 10, pp. 867–903, October 2001.

[18] M. Vitruvius, *De architectura*, Harvard University Press, Cambridge, MA, 1931.

[19] S. Sadie and J. Tyrrell, *The New Grove Dictionary of Music and Musicians*, Macmillan, New York, second edition, 2001.

[20] J. A. Moorer, "About this reverberation business," *Computer Music Journal*, vol. 3, no. 2, pp. 13–28, June 1979.

[21] D. Griesinger, "Practical processors and programs for digital reverberation," in *Audio Engineering Society Conference: Audio in Digital Times*, Toronto, Canada, May 1 1989, vol. 7, pp. 187–195.

[22] J.-M. Jot, "An analysis/synthesis approach to real-time artificial reverberation," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, San Francisco, CA, March 23–26 1992, vol. 2, pp. 221–224.

[23] J. O. Smith III, "Elimination of limit cycles and overflow oscillations in time-varying lattice and ladder digital filters," CCRMA Technical Report STAN-M-35, CCRMA, Stanford University, Stanford, California, 1986.

[24] M. R. Schroeder, "Improvement of feedback stability of public address systems by frequency shifting," *Journal of the Audio Engineering Society*, vol. 10, no. 2, pp. 108–109, April 1962.

[25] E. Berdahl and D. Harris, "Frequency shifting for acoustic howling suppression," in *International Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, September 6–10 2010, vol. 13.

[26] S. Costello, "ValhallaShimmer tips and tricks: Shimmering," Online, Dec. 2 2010, `https://valhalladsp.com/2010/12/02/valhallashimmer-tips-and-tricks-shimmering/`, Accessed September 22 2015.

[27] V. Norilo, "Exploring the vectored time variant comb filter," in *International Conference on Digital Audio Effects (DAFx-14)*, Erlangen, Germany, September 1–5 2014, vol. 17.

[28] P. White, "UA Lexicon 224: Reverb plug-in for UAD2 system," *Sound on Sound*, October 2011.

[29] S. Costello, "Algorithmic reverb, distortion, and noise," Online, July 7 2011, `https://valhalladsp.com/2011/07/07/algorithmic-reverbs-distortion-and-noise/`, Accessed September 22 2015.

[30] J. Parker, H. Penttinen, S. Bilbao, and J. S. Abel, "Modeling methods for the highly dispersive slinky spring: a novel musical toy," in *International Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, September 6–10 2010, vol. 13.

[31] M. Mathews and J. O. Smith III, "Methods for synthesizing very high Q parametrically well behaved two pole filters," in *Stockholm Musical Acoustics Conference (SMAC)*, Stockholm, Sweden, August 6–9 2003.

[32] D. Massie, "Coefficient interpolation for the Max Mathews phasor filter," in *Audio Engineering Society Convention*, San Francisco, CA, October 26–29 2012, vol. 133.

[33] Phil Collins, "Face value," LP, February 9 1981.

[34] "The 80s | AMS Neve," `http://ams-neve.com/about-us/ams-neve-history/80s`.

[35] A. Wang and J. O. Smith III, "On fast FIR filters implemented as tail-canceling IIR filters," *IEEE Transactions on Signal Processing*, vol. 45, no. 6, pp. 1415–1427, 1997.

[36] J. S. Abel and M. J. Wilson, "Luciverb: Iterated convolution for the impatient," in *Audio Engineering Convention*, San Francisco, CA, October 26–29 2012, vol. 133.

[37] E. Piirilä, T. Lokki, and V. Välimäki, "Digital signal processing techniques for non-exponentially decaying reverberation," in *COST-G6 Workshop on Digital Audio Effects (DAFx)*, Barcelona, Spain, November 19–21 1998, vol. 1.

[38] K. S. Lee and J. S. Abel, "A reverberator with two-stage decay and onset time controls," in *Audio Engineering Society Convention*, San Francisco, CA, November 4–7 2010, vol. 129.

[39] Boston, "Boston," LP, August 25 1976.

[40] Queen, "A night at the opera," LP, November 21 1975.

[41] H. Thornburg, "Antialiasing for nonlinearities: Acoustic modeling and synthesis applications," in *International Computer Music Conference*, Beijing, China, October 22–28 1999.

[42] M. Dolson, "The phase vocoder: A tutorial," *Computer Music Journal*, vol. 10, no. 4, pp. 14–27, Winter 1986.

[43] J. Laroche and M. Dolson, "Improved phase vocoder time-scale modification of audio," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 323–332, May 1999.

STEREO SIGNAL SEPARATION AND UPMIXING BY MID-SIDE DECOMPOSITION IN THE FREQUENCY-DOMAIN

Sebastian Kraft, Udo Zölzer

Department of Signal Processing and Communications
Helmut-Schmidt-University
Hamburg, Germany
sebastian.kraft@hsu-hh.de

ABSTRACT

An algorithm to estimate the perceived azimuth directions in a stereo signal is derived from a typical signal model. These estimated directions can then be used to separate direct and ambient signal components and to remix the original stereo track. The processing is based on the idea of a bandwise mid-side decomposition in the frequency-domain which allows an intuitive and easy to understand mathematical derivation. An implementation as a stereo to five channel upmix is able to deliver a high quality surround experience at low computational costs and demonstrates the practical applicability of the presented approach.

1. INTRODUCTION

The classical stereo format with a left and right speaker was developed by Blumlein in the 1930s. Although it allows to create a certain spaciousness by placing virtual sound sources on the azimuth angle between both loudspeakers, the result is still far away from a really realistic rendering of sound scenes. The main disadvantages of stereo are:

1. Impossibility to produce a real envelopment of the listener as rear sound sources are missing. These would be in particular important for a reproduction of ambient reflections to get a realistic impression of the room properties.
2. Phantom source positions are only properly reproduced for listeners in the sweet spot.

It has long been known that both problems can be solved by adding speakers in the rear for playback of ambient reflections and more speakers in the front for a stabilisation of the phantom source image. Today, surround sound is widely established in film and dedicated distribution media like DVD or Blu-Ray generally offer at least a 5.1 multi-channel sound track. In contrast, the majority of popular music is still exclusively produced and distributed in stereo and would not benefit from playback on more than two speakers. Even if more music productions would become available in multi-channel formats in the next years, there is still a huge stock of old stereo recordings.

Several approaches were developed in the past to benefit from additional loudspeakers while playing back stereo source material. The most simple ones use time-domain mixing matrices [1] together with phase shifting to generate the additional channels. More advanced upmixing algorithms usually follow a spatial source separation approach as depicted in Fig. 1. They try to split the stereo signal into a direct part and a diffuse and more or less uncorrelated ambient part. The direct signal will be repanned on the target speaker layout while the ambient signal will be equally redistributed to all speakers creating a uniform and diffuse sound field. In order to retain the perceived positions of the direct signal sources in the upmix, it is necessary to estimate their azimuth directions from the stereo signal and to incorporate this knowledge in the calculation of the repanning coefficients.

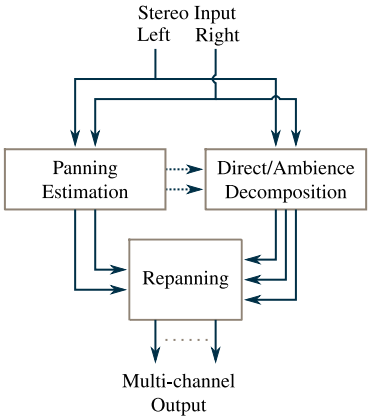


Figure 1: *Exemplary processing steps in a typical stereo to surround upmix algorithm.*

Dolby Pro Logic II [2] became quite popular in the 1990s and was intended to extract five channels from a stereo track. Basically it is a matrix encoder/decoder system but is also capable to extract decent multi-channel signals from any arbitrary stereo mix. For a simple and cost effective realization it only requires time domain operations like subtraction and addition of the left and right channels with additional phase shifts and VCAs (voltage controlled amplifiers) for simple directional steering.

Recent algorithms make use of frequency-domain processing to analyse the signal in discrete frequency bands. Avendano and Jot [3] calculate a bandwise inter-channel short-time coherence from the cross- and autocorrelations between the stereo channels which is then the basis for the estimation of a panning and ambience index [4]. Faller [5] uses a least squares method to derive an algorithm where the error between the extracted signals and a stereo signal model is minimised. Goodwin [6] examines the left and right stereo signals in a 2D vector space and extracts the direct and ambient sound with a principal component analysis. A similar geometric decomposition is described by Vickers [7] for the purpose of center extraction. An enhanced time-domain rever-

beration extraction upmixer was presented by Usher [8] where a normalised least mean squares (NLMS) algorithm is used to find a filter for the extraction of uncorrelated components. All the above algorithms [3–8] and also the one presented in this paper share a comparable stereo signal model with similar assumptions about the individual signal components.

The focus of this contribution is on a simplified mathematical description and derivation, which finally leads to a very fast and efficient implementation. It is shown that most of the processing principles, which are also known from other approaches, can be interpreted as a simple generalised mid-side decomposition which is performed in sub-bands.

In the following section 2, a typical mathematical model to describe stereo signals is derived and its connections to mid-side decomposition and principal component analysis are highlighted. Section 3 describes the extraction of the direct and ambient signals as well as the estimation of the direct signal directions. The upmixing of the separated components to a generic surround sound setup and other applications are outlined in section 4 before the results of an exemplary 2-to-5 upmix implementation are discussed in section 5. Section 6 will conclude the paper.

2. STEREO SIGNAL MODEL

The left and right channels of a stereo signal

$$x_L(n) = \left[\sum_{j=1}^{J} a_{L_j} \cdot d_j(n)\right] + n_L(n) \tag{1}$$

$$x_R(n) = \left[\sum_{j=1}^{J} a_{R_j} \cdot d_j(n)\right] + n_R(n) \tag{2}$$

are usually described as a weighted sum of J source signals $d_j(n)$ and additive uncorrelated ambient signals $n_L(n)$ and $n_R(n)$ in the left and right channel, respectively. The weightings a_{L_j} and a_{R_j} of the individual sources are called panning coefficients and are bound between zero and one. Their squared sum should be equal to one ($a_{L_j}^2 + a_{R_j}^2 = 1$) to achieve a constant power and loudness independent of the actual source position. As the panning coefficients are real-valued, this model only covers intensity stereophony where the weighted sources in the left and right channels are in phase.

The time-domain signal model can directly be transformed into the frequency-domain by a short time fourier transform (STFT)

$$X_L(b,k) = \left[\sum_{j=1}^{J} a_{L_j} \cdot D_j(b,k)\right] + N_L(b,k) \tag{3}$$

$$X_R(b,k) = \left[\sum_{j=1}^{J} a_{R_j} \cdot D_j(b,k)\right] + N_R(b,k) \tag{4}$$

where b and k denote the block and frequency indices. Based on the two stereo channels as input, it is impossible to mathematically retrieve the sources, panning coefficients and ambient signals as the equation systems (1)-(2), (3)-(4) are highly underdetermined. However, for a sufficiently high time and frequency resolution it is a common assumption [9] that at a certain time instant b and in a frequency band k only a single dominant source D_u is active and the contribution of other sources is close to zero

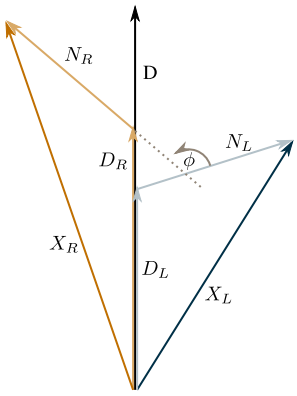


Figure 2: *The stereo signal model from (6)-(7) visualised as a complex vector diagram for a single frequency band k.*

($\sum_{\forall j \neq u} |D_j(b,k)| \approx 0$) . This allows to summarise the time-frequency representations of the individual sources

$$a_L(b,k) \cdot D(b,k) = \sum_{j=1}^{J} a_{L_j} \cdot D_j(b,k) \approx a_{L_u} \cdot D_u(b,k) \tag{5}$$

into a single source $D(b,k)$ and the panning coefficients to be written as a matrix $a_{L/R}(b,k)$. Moreover, the left and right ambient signal in one band k can be expected to have a similar magnitude but due to different paths and reflections in the room, they likely have a different phase and can be replaced by:

$$N_L(b,k) = N(b,k), \qquad N_R(b,k) = e^{j\phi} \cdot N(b,k).$$

Combining both steps leads to a simplified signal model

$$X_L(b,k) = a_L(b,k) \cdot D(b,k) + N(b,k) \tag{6}$$

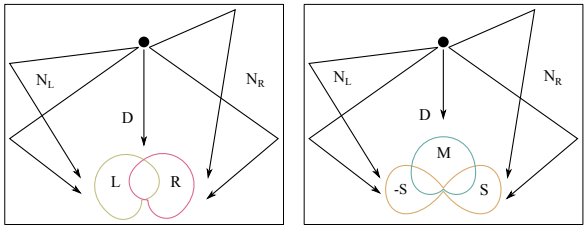
$$X_R(b,k) = a_R(b,k) \cdot D(b,k) + e^{j\phi} \cdot N(b,k) \tag{7}$$

with a highly reduced number of unknowns. For an improved readability the indices (b,k) are omitted in the next sections.

Overall, the simplifications in the signal model may seem to be quite drastic and one can doubt its validity in particular for complex and high density musical recordings. But the intention of the presented model is not to allow an exact extraction and reproduction of the original source signals. The idea is to have a generic signal model that is able to describe the relation between directional and diffuse signal components for an arbitrary stereo signal. The resulting ambient and direct signal components should be free of artefacts and sound realistic, but for that purpose they do not necessarily have to be identical to the real signals before the downmix.

2.1. Interpretation as complex vector diagram

The signals from the model (6)-(7) are complex-valued and can be depicted in form of a vector diagram to visualise their relation in phase and magnitude. In Fig. 2 the weighted direct signal components $D_{L/R} = a_{L/R} \cdot D$ are in phase with the direct signal D as the coefficients $a_{L/R}$ are only real-valued and do not alter the phase. In contrast the ambient components $N_{L/R}$ are out of phase



(a) *Left-right (XY) sterophony* (b) *Mid-side (MS) stereophony*

Figure 3: *Comparison between left-right and mid-side stereo recording techniques regarding the capturing of direct and ambient components.*

with the direct signal. Furthermore, N_L and N_R have a similar amplitude but a phase difference of ϕ.

For an angle $\phi = \pi$ the vector diagram transforms into the signal model from [6] where the direct signal is estimated as the principal component and the ambient signals are orthogonal to it.

2.2. Interpretation as mid/side stereophony

A stereo signal is typically recorded as a left and right channel (X_L/X_R) but it can also be represented by a mid and side channel (X_M/X_S). Both variants can be converted into each other

$$X_M = 0.5 \cdot (X_L + X_R) \tag{8}$$
$$X_S = 0.5 \cdot (X_L - X_R) \tag{9}$$

$$X_L = X_M + X_S \tag{10}$$
$$X_R = X_M - X_S \tag{11}$$

by sum and difference calculations. The connection between the above stereo representations and the signal model (6)-(7) becomes apparent in Fig. 3 where two classical stereo recording setups are depicted in a simple room model.

One way to record a left-right stereo signal is to use two cardoid microphones in a XY configuration as shown in Fig. 3 a). The direct sound D emitted from a central sound source will be recorded with the same intensity and phase by both microphones. The ambient signals N_L and N_R reach the left and right microphone with different phase but similar amplitude. This corresponds to the signal model from (6)-(7) in the case $a_L \equiv a_R$.

To record a mid-side stereo signal, a figure of eight microphone is faced sideways to capture the side signal and a cardoid micropone captures the mid signal. Placed in a room with a single central sound source as shown in Fig. 3 b), the mid microphone will nearly exclusively capture the direct signal, whereas the figure of eight mostly captures ambient reflections. Therefore, a mid-side stereo signal already implies a certain degree of separation between ambient and direct components. Having a left-right stereo signal pair as input, one can achieve a simple direct-ambience decomposition by calculating the mid and side signals with (8)-(9). Indeed, Dolby Pro Logic I and II [2,10], for example, already made use of this idea and basically obtained the center and ambient signals by sum and difference calculations of the left and right stereo channels. However, due to pure time-domain processing, their capability to separate multiple sources at the same time was limited.

3. SIGNAL EXTRACTION

3.1. Panning coefficient estimation

The panning coefficients in (6)-(7) are real-valued which means that they only create amplitude panning and do not introduce a phase difference between X_L and X_R. Any phase shift between the left and right channel would be solely caused by an additive ambient signal with a magnitude $|N| > 0$ (which is also apparent from Fig. 2). However, for typical music mixes the amplitude of the ambient signal N is far less than the amplitude of the direct signal D. This also means that the left and right channels

$$|X_L| \approx a_L \cdot |D| \tag{12}$$
$$|X_R| \approx a_R \cdot |D| \tag{13}$$

are sufficiently approximated by the weighted direct signal magnitude and the phase can be neglected in this relation. Rearranging and solving equations (12)-(13) with the constraint $a_L^2 + a_R^2 = 1$, the panning coefficients

$$\hat{a}_L = \frac{|X_L|}{\sqrt{|X_L|^2 + |X_R|^2}} \tag{14}$$
$$\hat{a}_R = \frac{|X_R|}{\sqrt{|X_L|^2 + |X_R|^2}} \tag{15}$$

can be estimated from X_L and X_R.

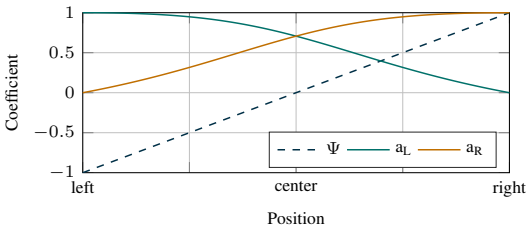
The "stereophonic law of sines" [11]

$$\frac{a_L - a_R}{a_L + a_R} = \frac{\sin\theta}{\sin \theta_0/2} = -\Psi \tag{16}$$

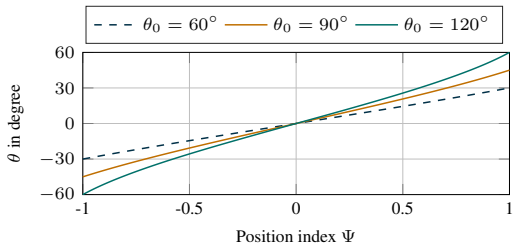
describes the perceived angle θ of a source if its amplitude is weighted by $a_{L/R}$ for playback on a left and right loudspeaker while the angle between the both speakers is defined by θ_0. The normalised position index Ψ, ranging from -1 for left and $+1$ for right positions, combines the coefficients $a_{L/R}$ in a single value (Fig. 4 a)). From (12)-(13) and (16) one can derive estimates for the position index and angle

$$\hat{\Psi} = \frac{|X_R| - |X_L|}{|X_L| + |X_R|} \tag{17}$$
$$\hat{\theta} = \arcsin\left(\sin \theta_0/2 \cdot \frac{|X_R| - |X_L|}{|X_L| + |X_R|}\right) \tag{18}$$

based on the magnitudes of the left and right stereo channel. When plotting the perceived source angle for different values of θ_0 and over the normalised position index Ψ in Fig 4 b), one can see that there is a nearly linear relation for the typical stereo setup with $\theta_0 = 60°$. Hence, the normalised position Ψ nicely matches the human perception and can be directly used as a linear pan-pot like control parameter to describe source positions. Comparing (17) with the mid-side decomposition mentioned in section 2.2, it appears that the position $\hat{\Psi}$ is the ratio between the side and mid component of an approximately coherent stereo signal.



(a) *Panning coefficients $a_{L/R}$ and position index Ψ*



(b) *Mapping between position index Ψ and the perceived angle θ in dependency of the angle between two speakers θ_0*

Figure 4: *Mapping between panning coefficients, position index and perceived angles.*

3.2. Direct and ambient signal separation

The estimated panning coefficients from the previous section can be used to solve the signal model (6)-(7) for

$$\hat{D} = \frac{X_L\, e^{j\phi} - X_R}{\hat{a}_L\, e^{j\phi} - \hat{a}_R} \tag{19}$$

$$\hat{N} = \frac{\hat{a}_L\, X_R - \hat{a}_R\, X_L}{\hat{a}_L\, e^{j\phi} - \hat{a}_R} \tag{20}$$

$$\hat{N}_L = \hat{N} \qquad = X_L - \hat{a}_L \cdot \hat{D}$$

$$\hat{N}_R = e^{j\phi} \cdot \hat{N} \qquad = X_R - \hat{a}_R \cdot \hat{D}$$

to get an estimate of the direct and ambient signal components. Currently no method is known to guess the angle ϕ and therefore, at the moment it is kept as an adjustable input parameter to influence the signal separation process. By setting $\phi = \pi$ the resulting left and right ambient signals are out of phase. While this may help to increase the perceived spatial depth of the ambient signals, it also causes unpleasant phase cancellations. Choosing ϕ in a range $\phi \in [0.5\,\pi, \pi]$ leads to less cancellations and for $\phi = 0.5\,\pi$ a maximum decorrelation between both ambient signals can be achieved. For the application of upmixing with a setup as described in the next section, an angle $\phi = 0.6\,\pi$ yielded the best balance between spatial depth and out-of-phase artefacts.

The formulas (19)-(20) already show a certain similarity with the mid-side decomposition previously described in section 2.2. Indeed, with $\hat{a}_L = \hat{a}_R = 1$ and a phase $\phi = \pi$ they become

$$\hat{D} = \frac{X_L\, e^{j\pi} - X_R}{e^{j\pi} - 1} = \frac{X_L + X_R}{2} \tag{21}$$

$$\hat{N} = \frac{X_R - X_L}{e^{j\pi} - 1} = \frac{X_L - X_R}{2} \tag{22}$$

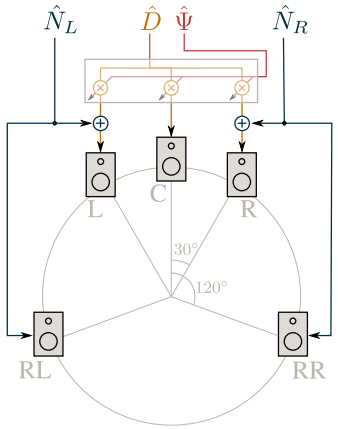


Figure 5: *Upmixing of the extracted ambient and direct signals to a five speaker surround setup.*

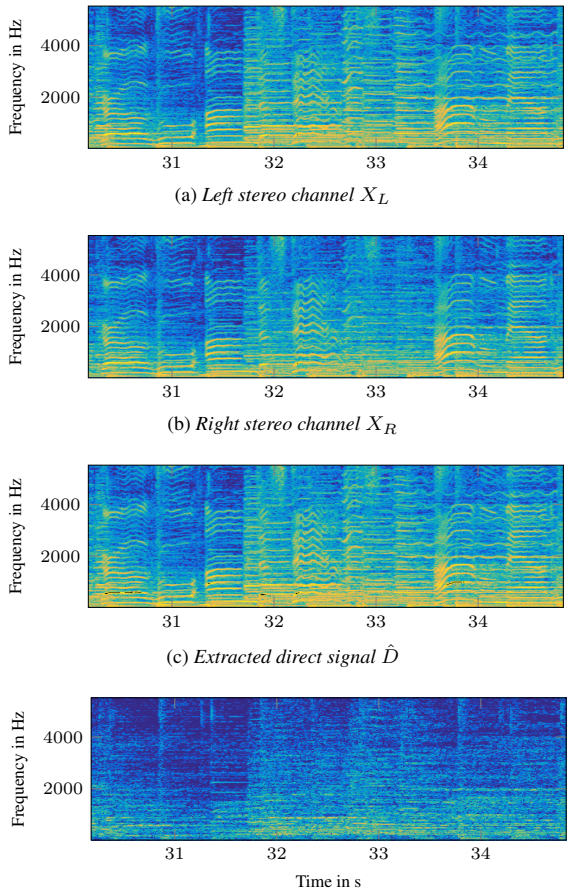
which is identical to (8)-(9) and shows that the derived direct and ambience separation essentially is a generalised mid-side decomposition. In contrast to a pure time-domain mid-side conversion, the panning coefficients in each sub-band are incorporated to allow a proper separation of direct signals which were not panned to the center.

4. REMIX

Using the separated signal components $\hat{D}, \hat{N}_L, \hat{N}_R$ and with the knowledge of the panning coefficients $\hat{a}_L, \hat{a}_R$ (or source positions $\hat{\Psi}$ and angles $\hat{\theta}$) from the previous section, it is possible to remix the original stereo signal. In the context of upmixing this would mean to redistribute the signals to a different loudspeaker arrangement. In most cases it is desired to retain the perceived source positions as they were placed in the stereo mix. However, it is also possible to widen or narrow the stereo panorama or to completely modify individual source positions. The ambient signal is equally distributed to all loudspeakers to create a diffuse sound field while the balance between ambient and direct signal can be modified.

The signal flow for a typical upmix scenario to five speakers is depicted in Fig. 5. The direct signal is played back by the three front speakers (L, C, R), whereas the corresponding weights for each front channel are obtained by *Vector Base Amplitude Panning* (VBAP) [12]. The left and right ambient signals are added to the four corner speakers (RL, RR, L, R). For a highly diffuse ambient sound field it is necessary to establish a low correlation between all ambient loudspeaker signals as pointed out by Kendall [13]. While the left and right ambient signals already have a low correlation if the angle ϕ is selected properly (cf. section 3.2), only the front and rear ambient signals on each side are fully correlated. In the most simple case these could be decorrelated by adding a small delay to the rear channels. More complex approaches in the frequency-domain are for example proposed by [13, 14].

Although the focus in this study is on the application of upmixing, the position index $\hat{\Psi}$ can also be used for general spatial source separation applications. First tests gave appealing results which are comparable to [15]. In combination with the separation of direct and ambient signals further applications as center channel



(a) *Left stereo channel* X_L

(b) *Right stereo channel* X_R

(c) *Extracted direct signal* $\hat{D}$

(d) *Extracted ambience signal* $\hat{N}$

Figure 6: *Spectrograms of the input signal a-b) and the extracted direct c) and ambient signals d).*

extraction [7], stereo speech enhancement [16,17] or the correction of panning errors with non-standard loudspeaker setups [18] could be possible.

5. DISCUSSION

The authors created a stereo to five channel upmixing VST plugin to test the practical suitability and sound quality of the described method. The target speaker layout and signal flow follows the diagram in Fig. 5. Rendered audio examples can be found on the website of the department[1].

The algorithm is quite efficient and only utilizes 3.0 % of a single CPU core on an Intel Core i5-2320 3 GHz processor at a sample rate of 44100 Hz. The size of the STFT blocks is set to 2048 samples with a hop size of 512 samples between two consecutive transforms. Profiling the code reveals that the main load is caused by the 2 FFT and 5 iFFT calculations which are required for a 2-to-5 frequency-domain upmix. The actual extraction of the source

[1] `http://www2.hsu-hh.de/ant/webbox/audio/kraft/DAFX15_upmix/`

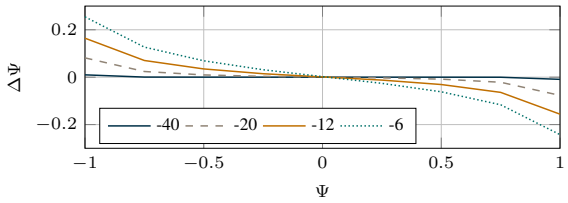


Figure 7: *Position index error* $\Delta\Psi$ *influenced by different ambience/direct power ratios in dB.*

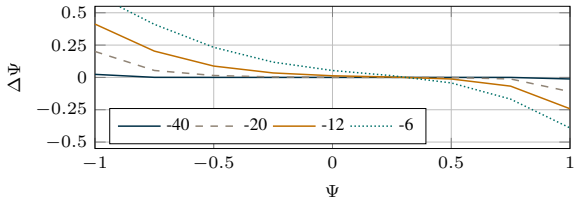


Figure 8: *Position index error* $\Delta\Psi$ *influenced by a second source with different power ratios (in dB) and fixed* $\Psi = 0.33$.

directions and the direct/ambience separation in the frequency domain with equations (14)-(15) and (19)-(20) only require a few instructions.

Spectrograms of the extracted signals for a sample song are shown in Fig. 6. It is apparent that the extracted direct signal in Fig. 6 c) includes most of the tonal energy from the input signal shown in Fig. 6 a-b). In contrast, the ambient signal Fig. 6 d) is diffuse and of lower energy. The tonal structure is only barely visible. The extracted ambient and direct signals sound realistic and although in particular the isolated ambient signal is not free of artefacts, they are not audible in the overall mix. This is caused by the fact that the upmixing process just redistributes signal components to more loudspeakers but nothing is removed or added compared to the original stereo signal. Although the positions of the sources are retained quite well when switching from stereo to surround, the perceived width of the spatial image tends to narrow a little bit in the upmix.

5.1. Estimated position error analysis

The generally proper reproduction of the source positions and pleasing ambient signals are a good indication that the simplifications and assumptions described in the above derivation do not cause any audible impairments and are valid for typical music material. However, it would be interesting to see how a violation of these assumptions will influence the accuracy of the estimated positions. For that purpose a direct signal consisting of white noise was panned to various positions $\Psi \in [-1, 1]$ and ambience was added using a *Large Hall* impulse response from a Bricasti M7 stereo reverb unit. The power ratio between the wet and dry signal was varied between -40 dB and -6 dB. The resulting error $\Delta\Psi = \hat{\Psi} - \Psi$ is plotted in Fig. 7 and it can be observed that the position error increases while the ambient signal power is increased. The consequence is that in particular extremely panned sources are estimated too close to the center. The same effect appears when a second source is added and the assumption of a single source per frequency band is violated. In Fig. 8 a white noise source signal

moving from left to right was overlaid with another fixed noise source at $\Psi = 0.33$ and different power levels. No ambient signal was added in this case. The error curve becomes asymmetric as the second source shifts the energy towards the right side but it has the same shape and behaviour as in the previous example. Still the error increases with a higher power of the disturbing source and with increased difference between both source positions. Although this is no detailed error analysis and only synthetic signals were used, the results confirm and visualise what was already perceived in the upmix application described before.

5.2. STFT resolution

The trade-off between time and frequency resolution is an important parameter for the algorithm as it is only capable to deal with a single source in a specific time-frequency point (b, k). Different block lengths for the STFT in a range from 256 to 8192 samples were investigated at a sample rate of 44100 Hz. The best sounding results are achieved with block lengths of 2048 samples, whereas the difference to 1024 or 4096 samples is only barely audible. First attempts were made to summarise frequency bins in the STFT into perceptual bands as done by Faller [5]. No obvious artefacts appeared even if the 2048 spectral bins were summarised in only 24 Bark bands. This opens a wide field of different methods to reduce and interpolate the spectral resolution and it might be in particular interesting to see if this enables us to increase the precision of the position estimation by using redundant information.

6. CONCLUSION

In this paper a new algorithm to estimate the perceived azimuth directions in a stereo signal was derived from a typical signal model. With the estimated directions it is possible to separate the direct and ambient signal components and to remix the stereo signal. Both, the signal separation and estimation of directions show similarity to a classical mid-side decomposition of stereo signals. However, in the presented form it is applied in sub-bands with the help of a short-time fourier transform and generalised to non-center panned signals. This allows to separate multiple sources with the constraint that only one dominant source is active at a specific time instant and frequency band. An implementation as a stereo to five channel upmixing VST plugin demonstrates the applicability and high sound quality of the proposed method at a very low computational cost.

For future enhancements it would be interesting to further investigate the influence of the STFT resolution and in particular the usage of perceptually motivated non-linear resolutions on the quality of the separated signals. Another aspect becoming more important with an increasing amount of loudspeakers is a proper decorrelation of the ambient signals to achieve a smooth and diffuse ambient sound field. Several approaches are known in the literature but their suitability in the context of the presented upmix algorithm have not been examined by the authors, yet.

7. REFERENCES

[1] Michael A. Gerzon, "Optimal Reproduction Matrices for Multispeaker Stereo," in *Proc. of the 91st AES Convention*, 1991.

[2] Roger Dressler, "Dolby Surround Pro Logic II decoder principles of operation," *Dolby White paper*, 2000.

[3] Carlos Avendano and Jean-Marc Jot, "A frequency-domain approach to multichannel upmix," *Journal of the Audio Engineering Society*, vol. 52, no. 7, pp. 740–749, 2004.

[4] Carlos Avendano and Jean-Marc Jot, "Ambience extraction and synthesis from stereo signals for multi-channel audio up-mix," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2002.

[5] Christof Faller, "Multiple-loudspeaker playback of stereo signals," *Journal of the Audio Engineering Society*, vol. 54, no. 11, pp. 1051–1064, 2006.

[6] Michael M. Goodwin and Jean-Marc Jot, "Primary-ambient signal decomposition and vector-based localization for spatial audio coding and enhancement," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2007.

[7] Earl Vickers, "Frequency-Domain Two- to Three-Channel Upmix for Center Channel Derivation and Speech Enhancement," in *Proc. of the 127th AES Convention*, 2009.

[8] John Usher and Jacob Benesty, "Enhancement of Spatial Sound Quality: A New Reverberation-Extraction Audio Upmixer," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 7, pp. 2141–2150, Sept. 2007.

[9] Alexander Jourjine, Scott Rickard, and Özgür Yilmaz, "Blind separation of disjoint orthogonal signals: demixing N sources from 2 mixtures," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2000.

[10] Roger Dressler, "Dolby Surround Pro Logic Decoder Principles Of Operation," *Dolby White paper*, 1982.

[11] Benjamin B. Bauer, "Phasor analysis of some stereophonic phenomena," *IRE Transactions on Audio*, vol. 10, no. 1, pp. 143–146, 1962.

[12] Ville Pulkki, "Virtual sound source positioning using vector base amplitude panning," *Journal of the Audio Engineering Society*, vol. 45, no. 6, pp. 456–466, 1997.

[13] Gary S. Kendall, "The Decorrelation of Audio Signals and Its Impact on Spatial Imagery," *Computer Music Journal*, vol. 19, no. 4, pp. 71–87, 1995.

[14] Marco Fink, Sebastian Kraft, and Udo Zölzer, "Downmix-compatibe conversion from mono to stereo in time- and frequency-domain," in *Proc. of the 18th Int. Conference on Digital Audio Effects*, 2015.

[15] Dan Barry, Bob Lawlor, and Eugene Coyle, "Sound source separation: Azimuth discriminiation and resynthesis," in *Proc. of the 7th Int. Conference on Digital Audio Effects*, 2004.

[16] Alexandra Cracuin, Christian Uhle, and Tom Bäckström, "An evaluation of stereo speech enhancement methods for different audio-visual scenarios," in *Proc. of the 23rd European Signal Processing Conference (EUSIPCO)*, 2015.

[17] Jürgen T. Geiger, Peter Grosche, and Yesenia Lacouture Parodi, "Dialog Enhancement of Stereo Sound," in *Proc. of the 23rd European Signal Processing Conference (EUSIPCO)*, 2015.

[18] Alexander Adami, Michael Schoeffler, and Jürgen Herre, "Re-Panning of Directional Signals and its Impact on Localization," in *Proc. of the 23rd European Signal Processing Conference (EUSIPCO)*, 2015.

AUTOMATIC SUBGROUPING OF MULTITRACK AUDIO

David Ronan, Hatice Gunes*
Centre for Intelligent Sensing,
Queen Mary University of London
London, UK
{d.m.ronan, h.gunes}@qmul.ac.uk

David Moffat, Joshua D. Reiss
Centre for Digital Music,
Queen Mary University of London
London, UK
{d.j.moffat, joshua.reiss}@qmul.ac.uk

ABSTRACT

Subgrouping is a mixing technique where the outputs of a subset of audio tracks in a multitrack are summed to a single audio bus. This is done so that the mix engineer can apply signal processing to an entire subgroup, speed up the mix work flow and manipulate a number of audio tracks at once. In this work, we investigate which audio features from a set of 159 can be used to automatically subgroup multitrack audio. We determine a subset of audio features from the original 159 audio features to use for automatic subgrouping, by performing feature selection using a Random Forest classifier on a dataset of 54 individual multitracks. We show that by using agglomerative clustering on 5 test multitracks, the entire set of audio features incorrectly clusters 35.08% of the audio tracks, while the subset of audio features incorrectly clusters only 7.89% of the audio tracks. Furthermore, we also show that using the entire set of audio features, ten incorrect subgroups are created. However, when using the subset of audio features, only five incorrect subgroups are created. This indicates that our reduced set of audio features provides a significant increase in classification accuracy for the creation of subgroups automatically.

1. INTRODUCTION

At the early stages of the mixing and editing process of a multitrack mix, the mix engineer will typically group audio tracks into subgroups. An example of this would be grouping guitar tracks with other guitar tracks or vocal tracks with other vocal tracks. Subgrouping can speed up the mix work flow by allowing the mix engineer to manipulate a number of audio tracks at once, for example by changing the level of all drums with one fader movement, instead of changing the level of each drum track individually. It also allows for processing that is not possible to achieve by manipulation of individual audio tracks. For instance, when non-linear processing such as dynamic range compression or harmonic distortion is applied to a subgroup, the processor will affect the sum of the sources differently than when it would be applied separately to every audio track. A Voltage Controlled Amplifier (VCA) group is another type of subgroup by which the individual faders of the group can be moved in unison, but each channel is not summed together and no subgroup processing can occur. An example of a typical subgrouping setup can be seen in Figure 1. Due to the varying amount of instrumentation used in recordings, there seems to be no clearly defined rules on how these subgroups are created, but it was found that the most commonly used approach is to group by instrument family [1].

* This work was supported by the EPSRC

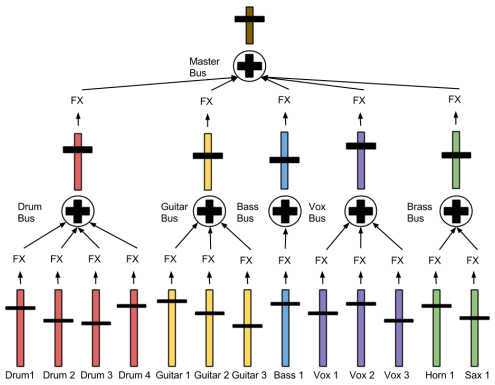


Figure 1: Typical mixing desk

In the literature reviewed, there is currently no proposals or discussions of a system that attempts to automate the subgrouping process [2, 3, 4, 5]. In this paper, we suggest that this can be done autonomously using machine learning techniques. The motivation is two-fold. Firstly, not only would it be possible to subgroup the audio tracks in the conventional sense, but through analysis of each audio track's spectro-temporal audio features, we may discover in this study that there are more intelligent ways to create subgroups.

Secondly, the audio features that are determined to be important can be used to answer the research question are we putting the instruments in the correct subgroups? Whereby, if we have good audio features to determine subgroups, this may inform us that a certain audio track or even certain sections of an audio track should be subgrouped differently from how they would be typically subgrouped [1]. An example of how this may work would be when we find over time that an audio track changes and may become more similar to another audio track in another subgroup. This could occur if the bass player suddenly switched from picking the bass guitar to playing in the style of slap bass. The audio track that was once in the bass subgroup could now be subgrouped with the percussive instrument audio tracks. At this point, it would make sense to split the single bass guitar audio track into two individual audio tracks and have them designated to their appropriate subgroups.

In light of the above discussion, the subgroup classification problem can been seen as somewhat similar to musical instrument identification, which has been done before for orchestral style instruments [6, 7, 8, 9]. However, in subgrouping classification we are not trying to classify traditional instrument families, but defined groups of instrumentation that would be used for the mix-

ing of audio from a specific genre. For example, in rock music the drum subgroup would consist of hi-hats, kicks and snares etc. while the percussion subgroup may contain tambourines, shakers and bongos. In practice, the genre of the music will dictate the type of instrumentation being used, the style in which the instrumentation will be played and what subgroup the instrument belongs to. It is also worth noting that typical subgroups such as vocals or guitars can be further broken down into smaller subgroups. In the case of vocals the two smaller subgroups might be lead vocals and background vocals. Furthermore, we can never assume that the multitrack recordings being used are good quality recordings. They may contain background noise, microphone bleed interference or other recording artefacts. All of these factors can affect the accuracy of a classification algorithm.

The purpose of this study is to determine the best set of audio features that can be extracted from multitrack audio in order to perform automatic subgrouping. In our particular case, we looked at multitracks that would be considered as Rock, Pop, Indie, Blues, Reggae, Metal and Punk genres, where the subgroups would typically be drums, bass, guitars, vocals etc. The rest of the paper is organised as follows. Section 2 describes the dataset used for feature selection and testing. Section 3 provides a list of features used and describes how they were extracted. Section 4 explains how the experiments, classification and clustering were performed. Section 5 presents the results obtained. Section 6 discusses the results and then finally the paper is concluded in section 7.

2. DATASET

The amount of data available for multitrack research is limited due to a multitrack being an important asset of a record label and the copyright issues that come with distributing them. A subset was selected from a larger multitrack test bed consisting of roughly 1.3 TB of data, including raw audio recordings, stems, mixdowns and recording session files. The Open Multitrack Testbed was used because it is one of the largest of its kind and contains data that is available for public use [10].

The subset used for feature selection consists of 54 separate multitracks and 1467 audio tracks in total once all duplicate audio tracks were removed. The multitracks that were used span a wide variety of musical genres such as Pop, Rock, Blues, Indie, Reggae, Metal, and Punk. We annotated each track by referring to its filename and then listening to each file for a brief moment to confirm its instrument type. The labels used for each audio file were based on commonly used subgroup instrument types [1]. These were drums, vocals, guitars, keys, bass, percussion, strings and brass. Table 1 shows the breakdown of all the multitrack data used for feature selection relative to what subgroup each audio track would normally belong to. It is worth noting the imbalance of label types in our dataset. This is because the most common instruments in our multitrack dataset are drums, vocals and guitars. Furthermore, the drum subgroup consists of many different types of drums such as kicks, snares, hi-hats etc. meaning it tends be the largest subgroup.

The subset used to test if the selected features were useful or not consists of five unseen multitracks. The breakdown of the different types of audio tracks for each test multitrack can be seen in Table 2.

Table 1: *Details of the subset used for feature selection*

Subgroup type	No. of tracks	Percentage of subset
Drums	436	29.72%
Guitars	365	24.88%
Vocals	363	24.74%
Keys	103	7.02%
Bass	93	6.34%
Percussion	80	5.45%
Strings	19	1.30%
Brass	8	0.55%

3. EXTRACTED AUDIO FETAURES

Each audio track in the dataset was downsampled to 22050 Hz and summed to mono using batch audio resampling software. The audio features were then extracted from the 30 secs of highest energy in each audio track. This was done to speed up the feature extraction process as we did not see the need to extract features from long periods of silence that occur in multitrack recordings. We extracted 159 low level audio features in total with a window size of 1024 samples and a hop size of 512 samples. A list of the audio features and the relevant references are in Table 3. Overall we have 42 different low level audio feature types and the majority of these are window based. Only three audio features were whole audio track features and not window based. Since the whole track audio features were not windowed like the others, no pooling was required [11]. We took the mean, standard deviation, maximum and minimum values of each windowed audio feature over the 30 secs of audio used for feature extraction. This allowed us to pool the windowed features over the 30 secs of audio and is the reason why we have 159 audio features in total [11].

4. EXPERIMENT

Two experiments were conducted. The first experiment determined a reduced set of audio features from the 159 audio features that we extracted previously. This was done by performing feature selection. The goal of this experiment was to determine the best subset of the 159 original audio features that could be used for automatic subgrouping. A second experiment was conducted where five test multitracks were agglomeratively clustered using all of the 159 audio features extracted and then agglomeratively clustered using

Table 2: *Details of the subset used for testing*

Subgroup type	MT 1	MT 2	MT 3	MT 4	MT 5
Drums	11	8	9	10	1
Vocals	17	11	6	9	3
Guitars	12	2	6	2	0
Keys	1	4	2	4	3
Bass	1	1	1	1	1
Percussion	1	0	1	0	0
Strings	0	0	0	0	6
Brass	0	0	0	0	0

Table 3: *Audio features*

Category	Feature	Reference
Dynamic	RMS	
	Peak Amplitude	
	Crest Factor	
	Periodicity	[12]
	Entropy of Energy	[13]
	Low Energy	[14]
Spectral	Zero Crossing Rate	[15]
	Centroid	.
	Spread	.
	Skewness	.
	Kurtosis	.
	Brightness	.
	Flatness	.
	Roll-Off (.85 and .95)	.
	Entropy	.
	Flux	.
	MFCC's 1-12	.
	Delta-MFCC's 1-12	[15]
	Spectral Crest Factor	[16]

the reduced feature set for comparison. This was done to investigate how well the reduced audio feature set compared to the entire audio feature set when performing automatic subgrouping.

4.1. Feature Selection

Random Forest is a particular type of Ensemble Learning method based on growing decision trees. This can be used for either classification or regression problems, but can also be used for feature selection. Random Forest is based on the idea of bootstrap aggregating or more commonly know as bagging. After training has occurred on a dataset each decision tree that is grown predicts an outcome. For regression decision trees, the output is the average value predicted by all of the decision trees grown. For classification decision trees it is the classification outcome that was voted most popular by all of the decision trees grown [17]. Random Forest was chosen because it has been proven to work very well for feature selection in other fields such as bioinformatics and medicine [18, 19].

Determining the most salient features using the Random Forest classifier was performed as follows. 100 decision trees were grown arbitrarily and a feature importance index was calculated. It will be seen further on in Section 5 that this was an appropriate amount of decision trees to grow. Random Forest feature importance can be defined for X^i, where the vector $X = (X^1, ...X^p)$, contains feature values and where p is the number of audio features used. For each tree t in the Random Forest, consider the associated OOB_t sample (this is the out-of-bag data that is not used to construct t). $errOOB_t$ denotes the error of a single tree t using the OOB_t sample. The error being a measure of the Random Forest classifier's accuracy. If the values of X^i are randomly permuted in OOB_t to get a different sample denoted by $\widetilde{OOB}_t^j$ and we compute $\widetilde{errOOB}_t^j$. $\widetilde{errOOB}_t^j$ being the error of t because of the different sample. The feature importance of X^i is equal to:

$$FI(X^i) = \frac{1}{ntree} \sum_t (\widetilde{errOOB}_t^j - \widetilde{OOB}_t^j) \qquad (1)$$

where the sum is over all trees t of the Random Forest and $ntree$ is the number of trees in the Random Forest [20].

The feature importance index was calculated for each of the 159 audio features. The average feature index was then calculated and the audio features that performed below the average were eliminated. The use of the average importance index was found to give us the most satisfactory set of audio features.

We also tried eliminating the 20% worst performing audio features, then retraining on the new audio feature set and repeating the 20% worst performing audio feature elimination process. This process would then stop once the out-of-bag error began to rise. However, we found that this was found to give us an unsatisfactory set of audio features. They were unsatisfactory because when we used these audio features to automatically create subgroups, the subgroups created were mostly incorrect e.g. drums in the same subgroup as guitars. This was the search method that was used in [20].

It should also be noted that when using the Random Forest classifier we set prior probabilities for each class based on our imbalanced dataset. The prior probabilities were set using the data in the *Percentage of subset* column in Table 1

4.2. Agglomerative clustering

Agglomerative clustering is a type of Hierarchical clustering. Generally in Hierarchical clustering a cluster hierarchy or a tree of clusters, also known as a dendrogram is constructed. This is not to be confused with the decision trees used in Section 4.1. An example of a dendrogram can be seen in Figure 4. Hierarchical clustering methods are categorised into agglomerative and divisive. The agglomerative clustering method is what we use in this experiment. The idea is that the algorithm starts with singular clusters and recursively merges two or more of the most similar clusters [21]. The reason why we chose agglomerative clustering is because the algorithmic process is similar to how a human would create subgroups in a multitrack. Initially, a human would find two audio tracks that belong together in a subgroup and then keep adding audio tracks until a subgroup is formed. An example would be pairing a kick track with a snare track and then pairing them with a hi-hat track to create a drum subgroup [1]. It is also worth noting that Figure 1 which is a typical subgrouping setup can be likened to a tree structure, so it would make sense to attempt to cluster audio tracks in a tree like fashion.

The agglomerative clustering algorithm can be described as thus [22]. Given a set of N audio feature vectors to be clustered.

1. Assign each audio feature vector N to its own singleton cluster and number the clusters 1 through c.
2. Compute the between cluster distance $d(r, s)$ as the between object distance of the two objects in r and s respectively, $r, s = 1, 2, ..., c$. Where $d(r, s) = \sqrt{\sum_c (r_c - s_c)^2}$ is the euclidean distance function and let the square matrix $D = (d(r, s))$.
3. Find the most similar pair of clusters r and s, such that the distance, $D(r, s)$, is minimum among all the pairwise distances, $d(c_i, c_j) = min\{d(r, s) : r \in c_i, s \in c_j\}$. This is what is known as the linkage function. A similar pair of clusters could be a snare track and a hi-hat track.

4. Merge r and s to a new cluster t and compute the between-cluster distance $d(t,k)$ for any existing cluster $k \neq r, s$. Once the distances are obtained, remove the rows and columns corresponding to the old cluster r and s in D, since r and s do not exist any more. Then add a new row and column in D corresponding to cluster t. Merging two clusters is like grouping two audio tracks together or else adding an audio track to an existing subgroup.
5. Iteratively repeat steps 3 to 5 a total of $c-1$ times until all the data items are merged into one cluster.

In our case the similarity is found between every pair of audio feature vectors that represent the audio tracks in our dataset. This is normally calculated using a distance function such as euclidean, manhattan or mahalanobis distance. We used euclidean distance as we found it gave us more realistic clusters. It is also worth noting that we normalised each instance in our dataset using L2-normalisation, while each audio feature value was normalised between zero and one. This was done due to the euclidean distance function being used. We then linked together audio feature vectors into binary pairs that were in close proximity to each other using a linkage function. We used the shortest distance measure as our linkage function, as this would make the most sense in our case as we are trying subgroup similar audio tracks based on instrumentation. The newly formed clusters created through the linkage function were then used to create even larger clusters with other audio feature vectors. Once linkage has occurred between all the audio feature vector clusters, all the branches of the tree below a specified cut-off are pruned. This cut-off can be specified as an arbitrary height in the tree or else the maximum amount of clusters to create. A maximum number of eight clusters was specified in our case. This was due to there only being eight labels in the original dataset used for feature selection.

Figure 4 depicts that any two audio tracks in the dataset become linked together at some level of the dendrogram. The height of the link is known as the cophenetic distance and represents the distance between the two clusters that contain those two audio tracks. If the agglomerative clustering is suited to a dataset, the linking of audio tracks in the dendrogram should have a strong correlation with the distances between audio tracks generated by the distance function. A cophenetic correlation coefficient can be calculated to measure this relationship. The cophenetic correlation coefficient is measured from -1 to 1 and the closer the value is to 1 the more accurately the dendrogram reflects the dataset. Suppose that the previous example dataset N_i has been modelled using the above cluster method to produce a dendrogram T_i. The cophenetic correlation coefficient is calculated as such

$$c = \frac{\sum_{i<j}(d(i,j)-\bar{d})(t(i,j)-\bar{t})}{\sqrt{\left[\sum_{i<j}(d(i,j)-\bar{d})^2\right]\left[\sum_{i<j}(t(i,j)-\bar{t})^2\right]}} \quad (2)$$

where $d(i,j)$ is the ordinary euclidean distance between the ith and jth observations of the dataset and $t(i,j)$ is the cophenetic distance between the dendrogram points T_i and T_j. $\bar{d}$ is the average of the $d(i,j)$ and $\bar{t}$ is the average of the $t(i,j)$.

5. RESULTS

In this section we present the results of the experiments conducted. We firstly show the results of the feature selection performed and then show the results of the agglomerative clustering. We also present the resulting dendrograms from the clustering.

5.1. Selected Features

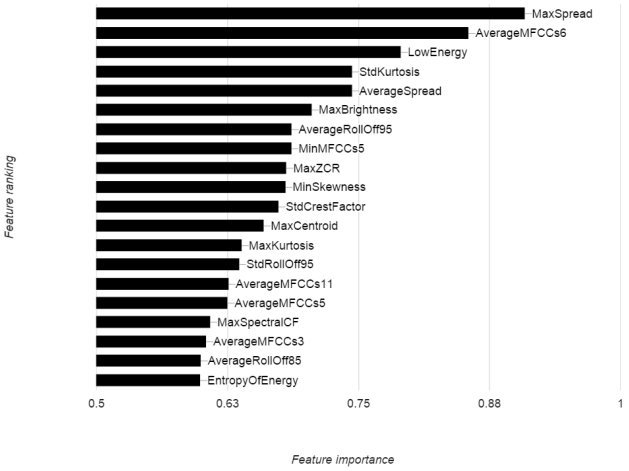


Figure 2: The 20 most important features

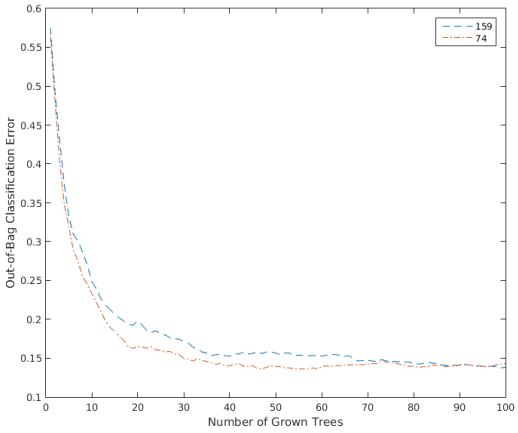


Figure 3: Cumulative out-of-bag classification errors for both feature sets

Using the feature selection method mentioned in Section 4.1 we determined a subset of 74 audio features from the original 159. The average feature importance index was 0.421 with a standard deviation of 0.1569. The maximum value for feature importance index was 0.9086 and the minimum was -0.0135. The 20 most important features are depicted in Figure 2. This illustrates some of the audio features that would occur in an audio feature vector used during agglomerative clustering.

The cumulative out-of-bag error having grown 100 trees with the full audio feature set was 0.1384. Using the reduced feature set and growing 100 trees the cumulative out-of-bag error was 0.1431.

Table 4: *Agglomerative clustering results using all features and the reduced feature set*

159 Features	**MT 1**	**MT 2**	**MT 3**	**MT 4**	**MT 5**
Cophenetic C.C.	0.799	0.844	0.751	0.894	0.814
Audio tracks	43	26	25	26	14
Incorrect subgroups	3	1	3	1	2
Incorrect audio tracks	19	7	5	7	2
Percentage incorrect audio tracks	44%	26.9%	20%	26.9%	14%
74 Features	**MT 1**	**MT 2**	**MT 3**	**MT 4**	**MT 5**
Cophenetic C.C.	0.771	0.887	0.806	0.924	0.956
Audio tracks	43	26	25	26	14
Incorrect groups	2	0	1	0	2
Incorrect audio tracks	6	0	1	0	2
Percentage incorrect audio tracks	13%	0%	4%	0%	14%

Table 5: *Agglomerative clustering results for all multitracks*

	159 Features	**74 Features**
Avg. Cophenetic C.C.	0.8203	0.8642
Total no. audio tracks	114	114
Avg. no. audio tracks	26.1	26.1
Total incorrect subgroups	10	5
Total incorrect audio tracks	40	9
Percentage incorrect audio tracks	35.08%	7.89%

Figure 3 shows that these results converge and start becoming very close after about 70 trees. This also supports our original choice to arbitrarily grow 100 decision trees for feature selection.

5.2. Agglomerative clustering

In Table 4 we present the results for each of the five multitracks that were agglomeratively clustered using the entire audio feature set and the reduced audio feature set. Also, we give the cophenetic correlation coefficients as described in Section 4.2. Also, we give the number of audio tracks in each multitrack as well as how many incorrect subgroups were created to show how well the clustering is at creating meaningful subgroups [1]. An incorrect subgroup would be where at least two different audio tracks with different instrument types are subgrouped together. An example of an incorrect subgroup would be if a subgroup consisted of drums, guitars and vocals. These three instrument types would normally be separate. There will always be eight subgroups due to the labels used in the training dataset, but these eight subgroups may not always be constructed correctly using agglomerative clustering. The number of incorrect audio tracks is measured by how many audio tracks were placed in a cluster where the majority of the instrument types where incompatible. An example being if we had a cluster of six guitars and two vocals. The guitars are the majority, so the incorrect audio tracks would be the vocal tracks. We also show this measure as a percentage of all the audio tracks in each multitrack.

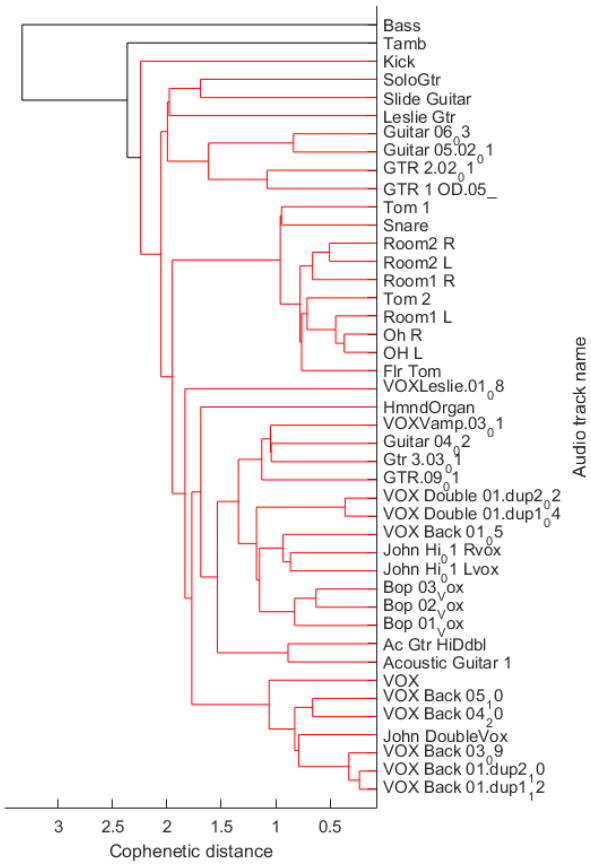


Figure 4: Dendrogram of MT 1 using the reduced feature set

6. ANALYSIS AND DISCUSSION

6.1. Selected Features

Looking at Figure 2 we can see the list is dominated by spectral features and has only three features related to dynamics. We were not surprised to see MFCC's in the 74 selected audio features as they have been proven before to perform quite well in speech recognition and audio classification tasks [23, 24, 25]. The Low Energy audio feature also plays a very significant role in classification. The Low Energy audio feature can be defined as the percentage of frames showing less than average RMS energy [14]. Vocals with silences or drum hits would have a high low energy rate compared to say a bowed string, so this may be one of the reasons it was so successful.

The maximum and average spectral spread as well as the standard deviation of kurtosis are also placed in top five ranked audio features. This suggests that the shape of the audio spectrum for each audio track was one of the most important factors. The spectral centroid, brightness and roll off 95% also featured in the top

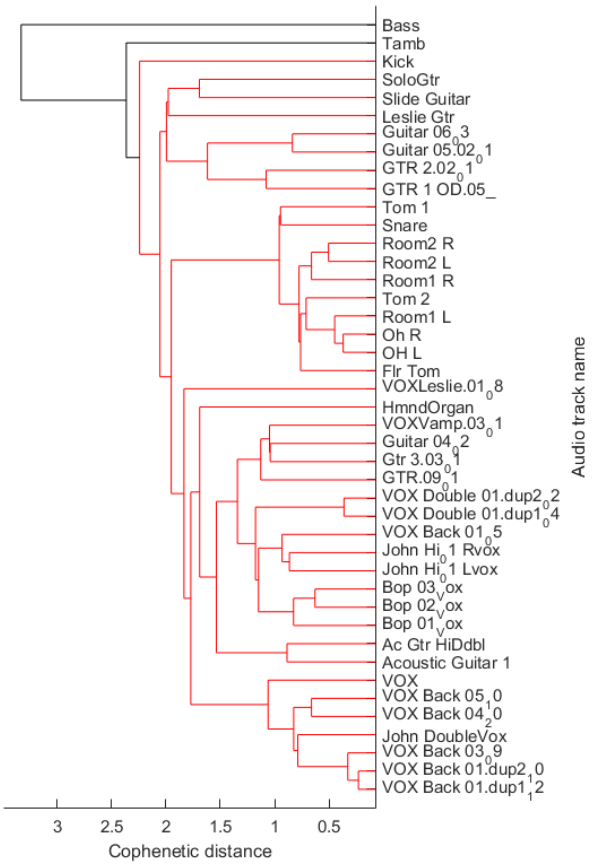


Figure 5: Dendrogram of MT 2 using the reduced feature set

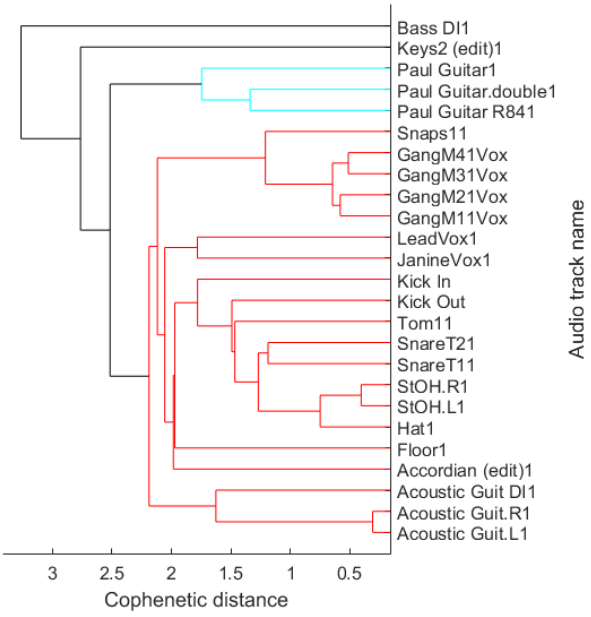


Figure 6: Dendrogram of MT 3 using the reduced feature set

20, which are all spectral features.

We were expecting the Periodicity feature to perform much better, but it did not even make it into the subset of 74 audio features. We expected this to be important for drum and percussion classification. Ideally, this would be predictably high for drums, but low for vocals.

6.2. Agglomerative clustering

If we compare the results from agglomerative clustering using the entire audio feature set and the reduced audio feature set we can clearly see that the reduced audio feature set achieved a higher performance. The overall percentage of incorrectly clustered audio tracks changes from 35.08% for the entire audio feature set to 7.89% for the reduced audio feature set. We also found that the reduced audio feature set has a slightly higher average cophenetic correlation coefficient than the entire audio feature set. This suggests the clustering better fits the reduced audio feature dataset. Furthermore, the total number of incorrectly created subgroups was halved when using the reduced audio feature set. Table 5 shows these results.

There is also an overall trend of higher performance for the reduced audio feature set when we examine each multitrack separately. MT 1 was the worst performing multitrack for both the entire audio feature set and the reduced audio feature set. MT 1 when using the reduced audio feature set, had a lower misclassification measure than MT 1 using the entire audio feature set, but surprisingly has a slightly lower cophenetic correlation coefficient. Overall, MT 1 had the lowest cophenetic correlation coefficient for both sets of audio features and this maybe because it also had the most amount of audio tracks to cluster. This may have been improved by using a varying maximum amount of clusters based on how many audio tracks are present. It is also worth mentioning that once the experiment was finished we listened back to the incorrectly subgrouped audio tracks for the reduced audio feature set and we found that these audio tracks suffered badly from microphone bleed. This is most likely the cause of the poor classification accuracy as two different instrument types can be heard on the recording. This problem could be addressed by using an automatic noise gate to reduce the microphone bleed [26].

The four other multitracks had greater success than MT 1 when clustered, but this may be due to them having less audio tracks to cluster. When we compare the results of the entire audio feature set versus the reduced audio feature set we can see a big improvement in results. Especially in MT 2 and MT 4 where the misclassification measure dropped to 0% in both cases. In MT 3, when using the reduced audio feature set we see that we had only one misclassification. This was the 'Snaps' audio track being subgrouped with the 'GangM' vocal tracks and is depicted in Figure 6. There is a small amount of microphone bleed on the 'GangM' vocal tracks, so this may be the reason why we are seeing this misclassifica-

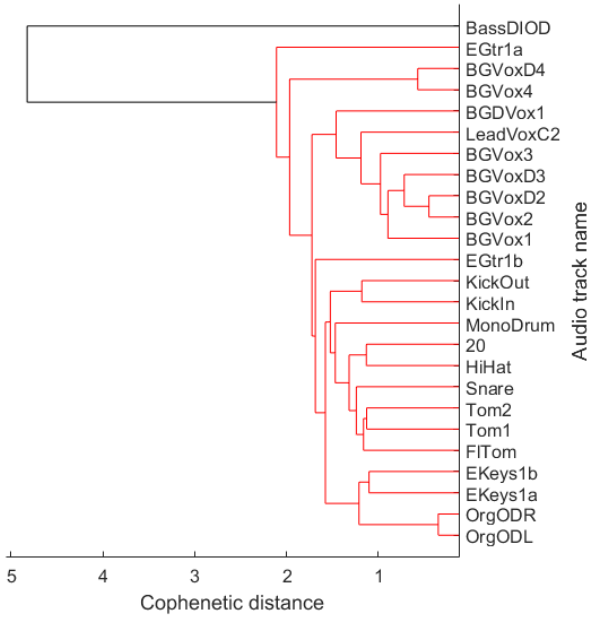


Figure 7: Dendrogram of MT 4 using the reduced feature set

tion. In MT 5 the misclassification is more difficult to explain as there does not seem to be any audible microphone bleed. This may be because the synthesiser has a similar timbre to the lead vocalist. Figure 8 shows that 'Synth21' is further away from the violins than the 'Synth11' is from the vocals, suggesting that 'Synth11' is similar to the vocal audio tracks.

When looking at Figure 4, Figure 5, Figure 6, Figure 7 and Figure 8 generally the lower parts of the trees tend to cluster the audio tracks together correctly. It is very easy to pick out drum, vocal and guitar clusters especially. The best examples are shown in Figure 5, Figure 6 and Figure 7. Interestingly, the 'Bass' audio track is the furthest distance from any other audio track in each of the multitracks. This most likely has to do with this instrument occupying the lower frequency bands and the rest of the instruments tending to be in mid and upper frequency ranges.

7. CONCLUSION

In this paper, we determined a set of audio features that could be used to automatically subgroup multitrack audio using a Random Forest for feature selection. We took a set of 159 low level audio features and reduced this to 74 low level audio features using feature selection. We selected these features from a dataset of 54 individual multitrack recordings of varying musical genre. We also showed that the most important audio features tended to be spectral features. We used the reduced audio feature set to agglomeratively cluster five unseen multitrack recordings. We then compared the results of the agglomerative clustering using the entire audio feature set to the agglomerative clustering using the reduced audio feature set. We were able to show that the overall misclassification measure went from 35.08% using the entire audio feature set

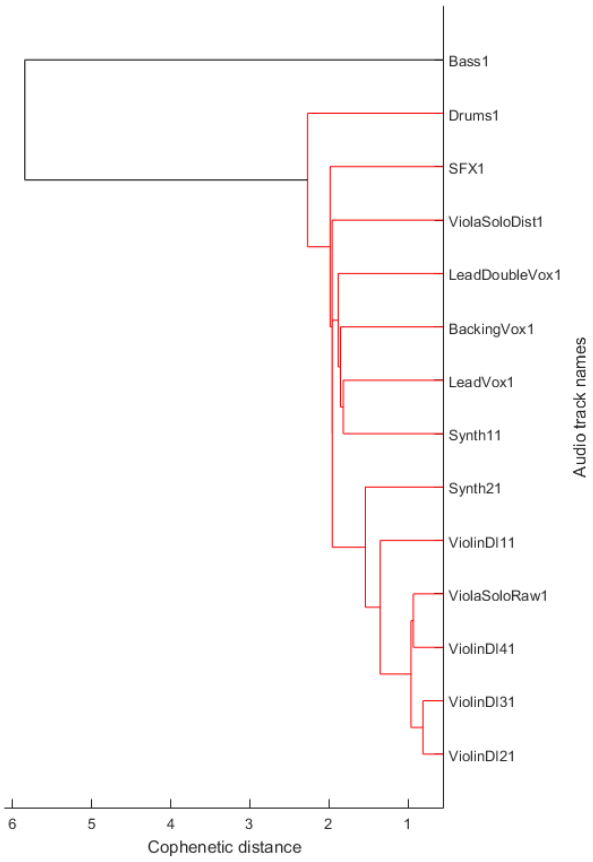


Figure 8: Dendrogram of MT 5 using the reduced feature set

to 7.89% using the reduced audio feature set. Thus indicating that our reduced set of audio features provides a significant increase in classification accuracy for the creation of automatic subgroups. Part of the novelty of this approach was that we were trying to classify audio tracks of entire multitrack recordings. Whereby, multitracks have the issue where recordings may contain artefacts such as microphone bleed. This did cause us problems in some cases, but we were easily able to identify the cause by listening to the problematic audio tracks.

In future work, automatic subgrouping could be applied to music from the Dance or Jazz music genres. In this case we only applied automatic subgrouping to Pop, Rock, Indie etc. However, it would seem that currently the subgroups for the Dance or Jazz music genres are not very well defined, so further research would be needed on best practices in subgrouping for music production of this kind. It would also be interesting to see how automatic subgrouping could be used in current automatic mixing systems like [27, 28, 29], where each automatic mixing algorithm is used on each subgroup of instruments individually to create a submix. Then once all the subgroups are automatically mixed, the auto-

matic mixing algorithm would be used to mix each individual subgroup. In this work we inspected the correctness of the automatically generated subgroups manually, in further work we would like to test the validity of this technique automatically by using cross validation.

8. REFERENCES

[1] David Ronan, Brecht De Man, Hatice Gunes, and Joshua D. Reiss, "The impact of subgrouping practices on the perception of multitrack mixes," in *139th Convention of the Audio Engineering Society*, October 2015.

[2] Jeffrey Scott and Youngmoo E Kim, "Instrument identification informed multi-track mixing.," in *14th International Society for Music Information Retrieval Conference (ISMIR 2013)*, 2013, pp. 305–310.

[3] Jeffrey Scott, Matthew Prockup, Erik M Schmidt, and Youngmoo E Kim, "Automatic multi-track mixing using linear dynamical systems," in *Proceedings of the 8th Sound and Music Computing Conference, Padova, Italy*, 2011.

[4] Joshua D Reiss, "Intelligent systems for mixing multichannel audio," in *17th International Conference on Digital Signal Processing (DSP), 2011*. IEEE, 2011, pp. 1–6.

[5] Enrique Perez-Gonzalez and Joshua D Reiss, "Automatic mixing," *DAFX: Digital Audio Effects, Second Edition*, pp. 523–549, 2011.

[6] Philippe Hamel, Sean Wood, and Douglas Eck, "Automatic identification of instrument classes in polyphonic and poly-instrument audio.," in *10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009, pp. 399–404.

[7] Judith C Brown, Olivier Houix, and Stephen McAdams, "Feature dependence in the automatic identification of musical woodwind instruments," *The Journal of the Acoustical Society of America*, vol. 109, no. 3, pp. 1064–1072, 2001.

[8] Antti Eronen and Anssi Klapuri, "Musical instrument recognition using cepstral coefficients and temporal features," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing, 2000. ICASSP'00.* IEEE, 2000, vol. 2, pp. II753–II756.

[9] Keith D Martin and Youngmoo E Kim, "Musical instrument identification: A pattern-recognition approach," *The Journal of the Acoustical Society of America*, vol. 104, no. 3, pp. 1768–1768, 1998.

[10] Brecht De Man, Brett Leonard, Richard King, and Joshua D. Reiss, "An analysis and evaluation of audio features for multitrack music mixtures," in *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, October 2014.

[11] Philippe Hamel, Simon Lemieux, Yoshua Bengio, and Douglas Eck, "Temporal pooling and multiscale learning for automatic annotation and ranking of music audio.," in *12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, 2011, pp. 729–734.

[12] Christian Uhle, "Tempo induction by investigating the metrical structure of music using a periodicity signal that relates to the tatum period," in *6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, 2005.

[13] Theodoros Giannakopoulos and Aggelos Pikrakis, *Introduction to Audio Analysis: A MATLAB® Approach*, Academic Press, 2014.

[14] George Tzanetakis and Perry Cook, "Musical genre classification of audio signals," *IEEE transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[15] Olivier Lartillot and Petri Toiviainen, "A matlab toolbox for musical feature extraction from audio," in *International Conference on Digital Audio Effects*, 2007, pp. 237–244.

[16] Geoffroy Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project," 2004.

[17] Leo Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[18] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn, "Bias in random forest variable importance measures: Illustrations, sources and a solution," *BMC bioinformatics*, vol. 8, no. 1, pp. 25, 2007.

[19] Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis, "Conditional variable importance for random forests," *BMC bioinformatics*, vol. 9, no. 1, pp. 307, 2008.

[20] Robin Genuer, Jean-Michel Poggi, and Christine Tuleau-Malot, "Variable selection using random forests," *Pattern Recognition Letters*, vol. 31, no. 14, pp. 2225–2236, 2010.

[21] Pavel Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*, pp. 25–71. Springer, 2006.

[22] Stephen P Borgatti, "How to explain hierarchical clustering," 1994.

[23] Zhouyu Fu, Guojun Lu, Kai Ming Ting, and Dengsheng Zhang, "A survey of audio-based music classification and annotation," *IEEE Transactions on Multimedia*, vol. 13, no. 2, pp. 303–319, 2011.

[24] Tim Pohle, Elias Pampalk, and Gerhard Widmer, "Evaluation of frequently used audio features for classification of music into perceptual categories," in *Proceedings of the Fourth International Workshop on Content-Based Multimedia Indexing*. Citeseer, 2005, vol. 162.

[25] Todor Ganchev, Nikos Fakotakis, and George Kokkinakis, "Comparative evaluation of various mfcc implementations on the speaker verification task," in *Proceedings of the SPECOM*, 2005, vol. 1, pp. 191–194.

[26] Michael Terrell, Joshua D Reiss, and Mark Sandler, "Automatic noise gate settings for drum recordings containing bleed from secondary sources," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, pp. 10, 2010.

[27] Enrique Perez-Gonzalez and Joshua Reiss, "Automatic equalization of multichannel audio using cross-adaptive methods," in *127th Convention of the Audio Engineering Society*. Audio Engineering Society, 2009.

[28] Michael Terrell, Andrew Simpson, and Mark Sandler, "The mathematics of mixing," *Journal of the Audio Engineering Society*, vol. 62, no. 1/2, pp. 4–13, 2014.

[29] Zheng Ma, Brecht De Man, Pedro Duarte Pestana, Dawn A. A. Black, and Joshua D. Reiss, "Intelligent multitrack dynamic range compression," *Journal of the Audio Engineering Society*, 2015.

SEPARATION OF MUSICAL NOTES WITH HIGHLY OVERLAPPING PARTIALS USING PHASE AND TEMPORAL CONSTRAINED COMPLEX MATRIX FACTORIZATION

Yi-Ju Lin, Yu-Lin Wang, Alvin W.Y. Su,

SCREAM Lab., Department of CSIE,
National Cheng-Kung University,
Tainan, Taiwan
lyjca.cs96@g2.nctu.edu.tw

Li Su,

Research Center for Information Technology
Innovation, Academia Sinica,
Taipei, Taiwan
li.sowaterking@gmail.com

ABSTRACT

In note separation of polyphonic music, how to separate the overlapping partials is an important and difficult problem. Fifths and octaves, as the most challenging ones, are, however, usually seen in many cases. Non-negative matrix factorization (NMF) employs the constraints of energy and harmonic ratio to tackle this problem. Recently, complex matrix factorization (CMF) is proposed by combining the phase information in source separation problem. However, temporal magnitude modulation is still serious in the situation of fifths and octaves, when CMF is applied. In this work, we investigate the temporal smoothness model based on CMF approach. The temporal ac-tivation coefficient of a preceding note is constrained when the succeeding notes appear. Compare to the unconstraint CMF, the magnitude modulation are greatly reduced in our computer simulation. Performance indices including source-to-interference ratio (SIR), source-to-artifacts ratio (SAR), source-to-distortion ratio (SDR), as well as modulation error ratio (MER) are given.

1. INTRODUCTION

Musical note separation (MNS) is the extension of musical source separation (MSS). MNS means to extract every note from a mixture source (e.g., the fuzzy interference [1],and the regular regression [2–4]). The fuzzy interference [1] separates the notes by estimating the harmonic rate of each partial; the regular regression [2–4] separates recursively and time-varyingly. On the other hand, MSS means to separate several sources from a mixture (e.g., the statistics [5], the sparse de-composition [6, 7], non-negative matrix factorization (NMF) [8–14], and the complex matrix factorization (CMF) [15–18]). In music, fifths and octaves may co-exist. Both cases produce the severe overlapping partial problems; therefore, how to separate the notes becomes an important and difficult issue. Some separation methods (e.g., NMF) focus on analysing the temporal activation coefficient of each note, but without considering the phase information. Recently, the CMF is proposed to separate the source by investigating the phase information [15], where each note could be analysed with the corresponding source. Usually, the CMF-separated magnitude parts of over-lapping partials are affected by their corresponding phases [15–18]. CMF also estimates most of the likely variations and dis-continuities of phases [18]. Hence, CMF is found more appropriate to separate the notes from different instruments. However, in practice, when fifths and octaves are played in a *interleaving* (i.e. notes appearing one after another) way, there exists a serious effect of *temporal magnitude modulation* in the separated notes. Such modulation is fairly audible in the sustain part of a note, where a listener can easily heard the preceding note is played one more time. The more overlapping partials there are, the more serious the modulation there is, as we will see in Figure 2 in the later section. Even the temporal sparsity and phase evolution constraints have been proposed in [18], the problems of overlapping partials still exist. Virtanen [13] investigates the temporal continuity to constrain the NMF on separation and achieve good results. Applying this constraint is valid when the temporal activation coefficients of each note would not change frequently in a short time. In order to mitigate the effect of temporal magnitude modulation in CMF-based note separation, the temporal smoothness constraint is investigated in this work. The constraint is applied to the preceding note before the succeeding notes come in for keeping the variation of the activation coefficient better, if the succeeding notes are fifths and/or octaves. The modulation is reduced dramatically and the phase continuity is kept, although the performance of the overall separation is still unsatisfactory. In addition to SIR, SAR, and SDR [19], MER proposed in [20] are used to evaluate the performance. All the test files are clipped from the RWC database [21]. The paper is organized as follows. Section 2 describes the CMF algorithm with phase constraint, and section 3 incorporate the temporal smoothness constrains into the CMF for reduction of temporal magnitude modulation. In section 4, the experiment results are shown and compared to the prior method [18]. Finally, conclusions and future works are given in section 5.

2. BACKGROUND

In this paper, the phase constraint of CMF is investigated for improving the note separation. The methodology of CMF [15] and its phase constraint [18] are introduced in this section.

2.1. Complex Non-Negative Matrix Factorization

CMF is involved the phase information to the matrix decomposition. This factorization supports the analysis of audio signal. Given an $N \times M$ complex-valued short-time Fourier transform (STFT) matrix $X \in \mathbb{C}^{N \times M}$. X could be decomposed into a basis matrix $W \in \mathbb{C}^{N \times K}$ satisfying $\sum_n W_{n,k} = 1, \forall k = 1, ..., K$,an activation matrix $H \in \mathbb{R}_{\geq 0}^{K \times M}$, and a tensor with phase information which can be represented as $Phi \in \mathbb{R}^{K \times M}$. The objective function of CMF is written as follows:

$$D_X = \sum_{n,m} |X_{n,m} - \hat{X}_{n,m}| \quad (1)$$

$$D_{\texttt{TSparsity}} = \sum_{k,m} |Hk, m|^g \tag{2}$$

$$\text{Minimize} : D_{\texttt{CMF}} = \frac{1}{2} D_X + \lambda D_{\texttt{sparsity}} \tag{3}$$

where $\hat{X}_{n,m} = W_{n,k} H_{k,m} \exp(i\Phi_{n,k,m})$.Here X is the spectrogram of the audio signal, W consists of the K spectra of K notes, H is the K temporal activation coefficients correspond to the each spectrum in different time, λ is the temporal sparsity parameter to penalize the objective function, and g is a parameter for descripting the shape of the sparse distribution. The only one constraint, here, is the temporal sparse constraint.

2.2. CMF under Phase Evolution constraints

Phase is a physical quantity which evolves regularly for most musical signals. Therefore, J. Bronson et al. [18] proposed an additional constraint of phase evolution in CMF to separate overlapping partials. This constraint is based on several assumptions: first, the pitches of each source should be known; second, there is no spreading energy of each partial bin; third, all the notes are played by the same instrument; and finally, each source can be represented by the combination of sinusoidal functions as eq. (4):

$$X_k = \sum_{p=1}^{P_k} A_{k,p} \exp[(\pi f_{0_k} pT + \phi_{0_k}, p)] \tag{4}$$

The cost function is based on these assumptions, written as follows:

$$\begin{aligned} D_{\texttt{Phase}} = \sum_{n,k,p,m} & \mathbb{1}\mathcal{N}_{k,p} |\exp(i\Phi_{n,k,m}) \\ & -\exp(i\Phi_{n,k,m-1}) \exp(i2\pi f_{0_k} pLT)|^2 \end{aligned} \tag{5}$$

where L is the frame shift in samples, T is the sampling period, and the $\mathbb{1}\mathcal{N}_{k,p}$ is the set of the membership function between the bins of the kth fundamental frequency and partial frequency. The cost function constrains each phase of the cur-rent frame should approximate to the estimated value from the earlier phase information. The objective function is shown as below:

$$\text{Minimize} : D_{\texttt{CMF}_\texttt{p}} = \frac{1}{2} D_X + \lambda D_{\texttt{sparsity}} + \sigma D_{\texttt{Phase}} \tag{6}$$

where σ is the phase continuity parameter for increase the rate of convergence. The optimization formula is derived in [18].

3. TEMPORAL SMOOTHNESS CONSTRAINT

In addition to the phase evolution constraint, the proposed temporal smoothness constraint is based on two assumptions: first, the temporal activation coefficients vary slowly unless encountering an onset. Second, the pitch f_0 and onset timing m_0 of each note are known before processing.

3.1. Temporal Smoothness Cost Function

The temporal smoothness cost function [2, 4, 13] is stated as follows:

$$D_{\texttt{TSmoothness}} = \sum_{k,m} |H_{k,m} - H_{k,m-1}|^2 \tag{7}$$

Eq. (7) implies that for the musical signal under analysis, $H_{k,m}$ should be close to $H_{k,m-1}$. This constraint forces the temporal activation coefficients vary slow with time, thereby mitigate the temporal magnitude modulation effect of the separated notes.

3.2. CMF under Phase Evolution and Temporal Smoothness Constraints

Combining the CMF algorithm with both the phase evolution constraint and the temporal smoothness constraint, the objective function is given as below:

$$\begin{aligned} \text{Minimize} : D_{\texttt{CMF}_\texttt{pt}} = & \frac{1}{2} D_X + \lambda D_{\texttt{sparsity}} \\ & + \sigma D_{\texttt{Phase}} + \gamma D_{\texttt{TSmoothness}} \end{aligned} \tag{8}$$

where γ is the parameter regularizing temporal smoothness. We modify the basic cost function of CMF, the phase evolution and temporal smoothness constraints. The temporal activation coefficients are limited with the onset-timing of each note. The phase evolution constraint is necessary for keeping continuity of the phase. For each note, the temporal smoothness constraint is applied from 100 ms before the note onset the instance of note offset. In our preliminary study we found 100 ms gives satisfactory result in general. Figure 1 shows the constraint is applied to the frames within the dash line. The CMF with Phase Evolution and temporal smoothness constraints is described in Algorithm 1.

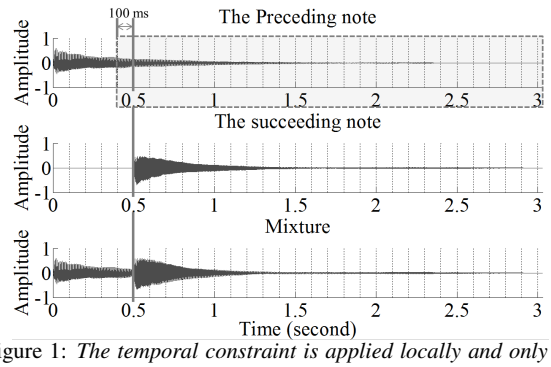


Figure 1: *The temporal constraint is applied locally and only on the preceding note starting from 100 ms before the on-set of the succeeding note till the end of signal.*

4. EXPERIMENTS AND RESULTS

In the following experiments, the proposed note separation method is evaluated on a set of mixture samples with two notes constituting intervals of perfect fifth, octave, and tritave (e.g., an octave plus a perfect fifth). Every mixture sample is made by adding the signals of the two notes together where the higher note is lagged by 0.5 second. For example, a mixture "A3+A4" contains a note A3 whose onset is at 0 second and a note A4 whose onset is at 0.5 second. In this way we can observe the temporal modulation effect of the lower note when the higher note joins in at 0.5 second. The notes are played with the same musical instrument for every sample. All sources are obtained from the part of piano, violin and guitar in the RWC database [21]. The sampling rate is 44.1 kHz. In computing the short-time Fourier transform (STFT) representation, we use a window with length of 4,096 samples and the hop size is 256 samples. The constraints of sparsity and phase have been discuss in the prior works [15, 18], we simply set the sparsity parameter $\lambda = 0.001$, and the phase continuity parameter $\sigma = 0.1$ according to the previous result. To show the temporal modulation

Algorithm 1: CMF with Phase Evolution and temporal smoothness constraints

Input: $X \in \mathbb{C}^{N\times M}$, $K \in \mathbb{N}$, $f_{0_k}(k = 1, ..., K)$ and $m_{0_k}(k = 1, ...K)$
Output: W, H, and Φ s.t.
$X_{n,m} \approx \sum_{k=1}^{K} W_{n,k}H_{k,m}\exp(i\Phi_{n,k,m})$
$W \in \mathbb{C}^{N\times M}$, $H \in \mathbb{R}^{N\times M}_{\geq 0}$, $\Phi \in \mathbb{R}^{N\times K\times M}$
Initialized;
while *stopping criteria not met* **do**
Compute β
$\beta_{n,k,m} = \frac{W_{n,k}H_{k,m}}{\sum_k W_{n,k}H_{k,m}}$
Compute $\hat{X}$
$\hat{X}_{n,k,m} = W_{n,k}H_{k,m}\exp(i\Phi_{n,k,m}) + \beta_{n,k,m}(X_{n,m} - \sum_k \hat{X}_{n,k,m})$
Compute $\hat{H}$
$\hat{H}_{k,m} = H_{k,m}$
Compute Φ
$\Phi_{n,k,m} = \mathrm{Arg}\{\frac{\hat{X}_{n,k,m}}{\beta_{n,k,m}}W_{n,k}H_{k,m} + \sigma\sum_p \mathbb{1}\mathcal{N}_{k,p}[\exp(i\Phi_{n,k,m-1})\exp(i2\pi f_{0_k} rLT) + \exp(i\Phi_{n,k,m+1})\exp(-i2\pi f_{0_k} rLT)]\}$
Compute W
$W_{n,k} = \frac{\sum_m H_{k,m}\Re[(\frac{\hat{X}_{n,k,m}}{\beta_{n,k,m}})\exp(-i\Phi_{n,k,m})]}{\sum_m \frac{H^2_{k,m}}{\beta_{n,k,m}}}$
Compute H
if $m_{0_{k+1}} + 100 < m \leq m_k$ **then**
$H_{k,m} = \frac{\sum_m W_{n,k}\Re[(\frac{\hat{X}_{n,k,m}}{\beta_{n,k,m}})\exp(-i\Phi_{n,k,m})]}{\sum_m \frac{W^2_{n,k}}{\beta_{n,k,m}} + \lambda g(\hat{H}_{k,m})^{g-2}}$
else if $m < m_{0_{k+1}} + 100$ **then**
$H_{k,m} = \frac{\gamma H_{k,m-1} + \sum_m W_{n,k}\Re[(\frac{\hat{X}_{n,k,m}}{\beta_{n,k,m}})\exp(-i\Phi_{n,k,m})]}{\sum_m \frac{W^2_{n,k}}{\beta_{n,k,m}} + \lambda g(\hat{H}_{k,m})^{g-2} + \gamma}$
else
$H_{k,m} = 0$
end
Project H onto non-negative orthant
iter=iter+1
end

effect and the separation performance using the smoothness constraint, the activation coefficient of each separated note with the lower pitch is displayed from 0.4 to 0.8 second. The magnitude of the higher pitch is not shown because it does not undergo the temporal magnitude modulation effect. We also compute and show dB-scaled SIR, SAR, SDR, and MER, where the first three are usually used to evaluate the separated sources in the audio source separation [19], and the last one is used to evaluate digital signal transmitter or receiver in a communications system, as formulated by [20]

$$\mathrm{MER} := 10\log_{10}(\frac{P_{\mathtt{signal}}}{P_{\mathtt{error}}}) \quad (9)$$

where $P_{\mathtt{signal}}$ is the RMS power of original signal and $P_{\mathtt{error}}$ is the RMS power of the error signal (e.g., the RMS power of the difference between the original and the separated signal). Here, we use MER to compare the temporal activation coefficients of the original and separated notes.

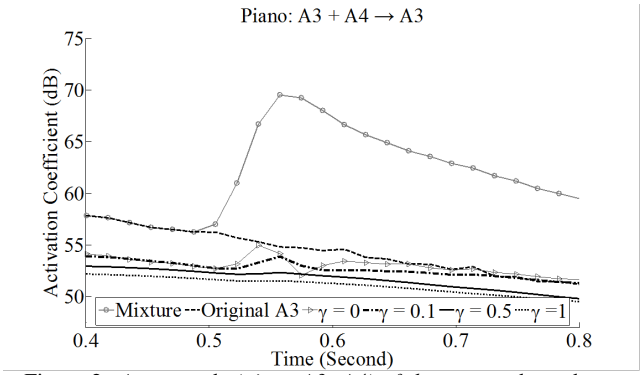


Figure 2: *An example (piano A3+A4) of the temporal envelopes of the preceding note (A3) with and without temporal smoothness constraint. The onset of the succeeding note (A4) is at about 0.5 second.*

4.1. Temporal Smoothness Parameter γ

We firstly investigate a case of A3+A4 in piano to observe the behaviors of the temporal magnitude modulation effect on the separated notes under different temporal smoothness parameters $\gamma = 0$ (no temporal smoothness constraint; the same case as in [18]), 0.1, 0.5 and 1. Figure 2 shows the temporal magnitudes of the mixture (A3+A4) before separation, and of the lower note (A3) before and after separation under different γ. Obviously, the magnitude of the separated A3 note without the temporal constraint (e.g., $\gamma = 0$) undergoes serious modulation after 0.5 second. For $\gamma > 0$, the temporal smoothness constraint suppresses the unwanted modulation. When γ increases, suppression becomes better while sacrificing overall note energy at the same time; the power of the separated A3 is by from 5 to 10 dB smaller than the original A3. In the following experiments on three different classes of instruments (i.e., piano, guitar and violin), we will compare only the case $\gamma = 0.5$ to the case $\gamma = 0$.

4.2. Piano

To give a more systematic evaluation, we consider three different note pairs (i.e., interval) and each with three distinctive pitch combinations: perfect fifth (i.e. A3+E4, E3+B3, and G3+D4), octave (i.e. A3+A4, E3+E4, G3+G4) and tritave (i.e. A3+E5, E3+B5, G3+D5), all of which are piano sounds. Figure 3 shows the temporal envelopes of the separated notes under three selected cases, and Table 1 shows the evaluation results. Similar to figure 2, from figure 3 we observe clearly that the temporal envelope becomes smoother after employing the temporal smoothness constraint. For the cases of fifth and tritave, there is less difference between $\gamma = 0$ and $\gamma = 0.5$ because there are less overlapping partials for the pairs. If there are more overlapping partials, such as the case of the octave where all the partials of the higher note are overlapped, the temporal smoothness constraint reduces the modulation a lot. However, from Table 1 we found no significant improvement for $\gamma = 0.5$ in terms of SIR, SAR, SDR and MER results, possibly due to the fact that the performance of piano note separation is intrinsically better: This can be seen from the case of perfect fifth and tritave, where the SDR with no temporal smoothness constraint is 9.2 and 9.9 dB, both of which are sufficiently high. Notably, the SDRs of the octave are only 1.3 and 1.6 dB for $\gamma = 0$

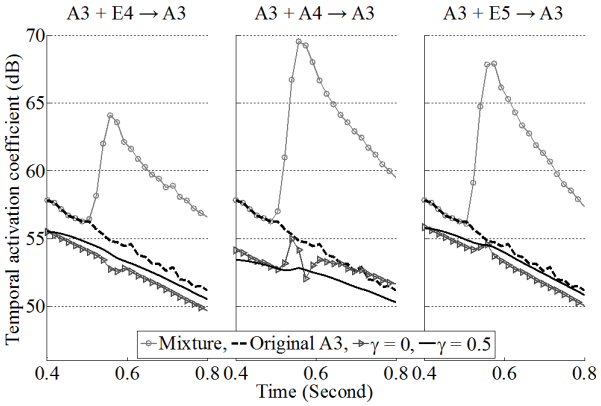


Figure 3: *Three cases of piano. Left: perfect fifth. Middle: octave. Right: tritave (circle: mixture, dashed line: Original A3, triangle:* $\gamma = 0$*, solid line:* $\gamma = 0.5$

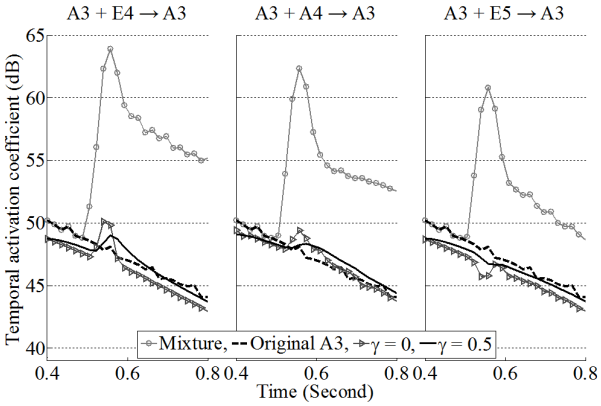


Figure 4: *Three cases of guitar. Left: perfect fifth. Middle: octave. Right: tritave (circle: mixture, dashed line: Original A3, triangle:* $\gamma = 0$*, solid line:* $\gamma = 0.5$

and 0.5, respectively, showing the challenge of separating octave notes.

Table 1: *The piano cases: The evaluation*

		Fifth		Octave		Tritave	
γ		0	0.5	0	0.5	0	0.5
Preceding note	SIR	14.7	14.7	4.5	5.2	17.2	17.0
	SAR	11.0	11.0	6.4	6.2	11.0	10.9
	SDR	9.2	9.2	1.3	1.6	9.9	9.8
	MER	13.8	14.0	9.1	9.3	13.6	13.3
Succeeding note	SIR	23.8	23.5	13.3	13.7	22.7	22.9
	SAR	15.0	15.0	12.9	12.8	15.3	15.2
	SDR	14.3	14.3	9.9	10.0	14.3	14.3
	MER	8.4	8.4	8.2	8.2	7.0	7.0

4.3. Guitar

We also present the cases of guitar sounds. Similarly, three cases with three sets of pitch combinations are considered: perfect fifth (i.e. A3+E4, E3+B3, and G3+D4), octave (i.e. A3+A4, E3+E4, G3+G4) and tritave (i.e. A3+E5, E3+B5, G3+D5). Figure 4 shows the temporal envelopes of the separated notes and Table 2 show the evaluation results. We also observe the suppression of temporal magnitude modulation, and a slightly improvement in terms of SIR, SAR, SDR and MER, in general. For example, in the case of perfect fifth, the SDR of preceding notes increases from 8.3 to 10.4 dB, SIR from 14.0 to 17.4 dB, and MER from 13.9 to 14.2 dB.

4.4. Violin

Finally, for violin, we consider the following cases: perfect fifth (i.e. A3+E4, B3+F4, and G3+D4), octave (i.e. A3+A4, B3+B4, G3+G4) and tritave (i.e. A3+E5, B3+F5, G3+D5). The experiment result is presented in Figure 5 and Table 3. In comparison to the cases of piano and guitar, the proposed temporal smoothness constraint is found more effective for violin. Not only can we see the suppression of the magnitude modulation by the higher note, but we also see general improvement of SIR, SAR, SDR and MER. Specifically, SIR is improved the most: in the case of tritave, the SIR of preceding notes is improved by 4.7 dB, and SDR by 4 dB. In all, the SIRs of the three cases are improved by at least 1.5 dB to 4.7 dB.

Table 2: *The guitar cases: The evaluation*

		Fifth		Octave		Tritave	
γ		0	0.5	0	0.5	0	0.5
Preceding note	SIR	14.0	17.4	12.5	13.6	16.7	18.4
	SAR	12.2	12.1	11.4	11.2	13.3	12.4
	SDR	8.3	10.4	7.0	7.8	9.7	10.3
	MER	13.9	14.2	17.0	17.0	17.5	17.0
Succeeding note	SIR	27.5	28.2	24.4	24.4	24.9	25.2
	SAR	13.6	13.9	13.9	13.8	11.9	12.2
	SDR	12.9	13.3	12.6	12.5	11.2	11.4
	MER	4.0	4.1	4.5	4.5	2.9	2.9

4.5. Discussion

First, according to the experimental results, the proposed temporal constraint approach greatly reduces the temporal magnitude modulation when separating notes with highly overlapping partials using CMF. In some cases we further observe improvement of the objective figures of merit. In particular, the SIR is increased by more than 4 dB in the cases for long-sustain notes like violin. However, for stuck-string or plucked-string instruments like piano or guitar, the objective performance indices are just marginally improved or unimproved. It is noted that the temporal constraint parameter shouldn't be too large and should be chosen carefully. Empirically, the parameter should be chosen between 0 and 1. Finally, in conventional CMF algorithm [15–18], the template matrix representing the frequency domain characteristics is fixed for all input features in the whole music piece. This is certainly unreasonable in most cases. As a fixed set of template is unsuitable for the modeling of vibrato, which is usually seen in violin notes. Since the objective measures such as SDR and SIR may not well reflect whether we have properly taken care of the modulation effect, in

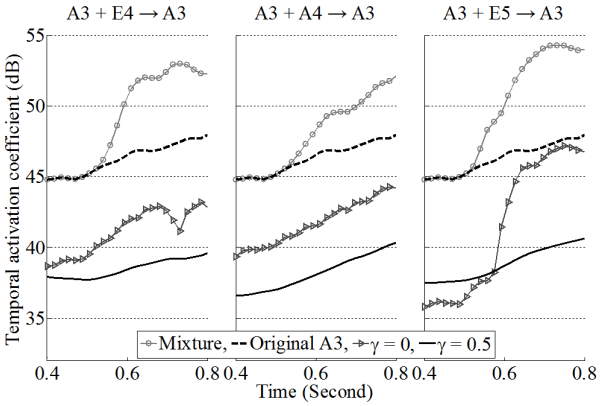


Figure 5: *Three cases of violin. Left: perfect fifth. Middle: octave. Right: tritave (circle: mixture, dashed line: Original A3, triangle:* $\gamma = 0$*, solid line:* $\gamma = 0.5$

Table 3: *The violin cases: The evaluation*

		Fifth		Octave		Tritave	
γ		0	0.5	0	0.5	0	0.5
Preceding note	SIR	10.2	11.6	8.7	12.6	17.2	21.9
	SAR	7.3	6.2	6.3	5.1	16.0	13.3
	SDR	4.6	4.6	3.3	4.0	9.0	12.0
	MER	4.0	3.2	4.4	3.3	2.4	3.8
Succeeding note	SIR	10.9	9.7	18.4	17.8	-0.6	0.4
	SAR	14.1	14.6	16.9	16.7	9.1	9.7
	SDR	8.9	8.3	10.1	9.5	-2.6	-1.7
	MER	6.4	7.6	10.3	12.0	5.8	10.5

the future we will also provide audio examples for subjective evaluation [22].

5. CONCLUSION AND FUTUREWORKS

Though CMF outperforms NMF in note separation applications, severe temporal magnitude modulation is presented when the notes have highly overlapping parts, especially in the cases of octaves. In this paper, the conventional CMF is combined with a temporal smoothness constraint, which not only reduces the temporal magnitude modulation but also improve the performance figures of merit including SIR, SDR, and SAR, and MER. Although the proposed method improves the original CMF note separation, the overall results are still unsatisfactory. Particularly, in the case of octave, all the objective figures of merit shown in this paper are still poor in comparison to other cases. Moreover, when processing vibrato notes, the situation is even worse, perhaps because that the template matrix corresponding to the frequency domain characteristics is fixed in the time domain during the separation process. This suggests a future work of developing a better solution such that the template matrix can be adaptable to the temporal of the notes.

6. ACKNOWLEDGMENT

The authors would like to thank the National Science Council, ROC, for its financial support of this work, under Contract No.MOST 103-2221-E-006-140-MY3.

7. REFERENCES

[1] G. Monti and M. B. Sandler, "Automatic polyphonic piano note extraction using fuzzy logic in a blackboard system," in *Proc. of the 5th Int. Conference on Digital Audio Effects (DAFx-02)*, 2002, number OCTOBER 2002, pp. 39–44.

[2] Y. J. Lin, T. M. Wang, T. C. Chen, Y. L. Chen, W. C. Chang, and A. W. Y. Su, "Musical note analysis of solo violin recordings using recursive regularization," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 25, pp. 1–13, 2014.

[3] Y. J. Lin, W. C. Chang, and T. M. Wang, "Timbre-constrained recursive time-varying analysis for musical note separation," in *Proc. of the 16th . . .* , 2013, pp. 3–7.

[4] T. M. Wang, T. C. Chen, Y. L. Chen, and A. W. Y. Su, "Time-Dependent Recursive Regularization for Sound Source Separation," in *Audio, Language and Image Processing (ICALIP), 2012 International Conference on*, 2012, pp. 235–240.

[5] A. Belouchrani, K. Abed-Meraim, J. F. Cardoso, and E. Moulines, "A blind source separation technique using second-order statistics," *IEEE Transactions on Signal Processing*, vol. 45, no. 2, pp. 434–444, 1997.

[6] M. Zibulevsky and B. A. Pearlmutter, "Blind source separation by sparse decomposition in a signal dictionary.," *Neural computation*, vol. 13, no. 4, pp. 863–882, 2001.

[7] P. Bofill and M. Zibulevsky, "Underdetermined blind source separation using sparse representations," *Signal Processing*, vol. 81, no. 11, pp. 2353–2362, 2001.

[8] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in neural information processing*, , no. 1, 2000.

[9] P. Smaragdis, "Non-negative matrix factorization for polyphonic music transcription," in *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on*, 2003, number 1, pp. 177–180.

[10] P. Smaragdis, "Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs," *Independent Component Analysis and Blind Signal Separation*, , no. 2, pp. 494–499, 2004.

[11] A. Cichocki, R. Zdunek, and S. Amari, "New algorithms for non-negative matrix factorization in applications to blind source separation," in *Acoustics, Speech and Signal Processing (ICASSP), 2006 IEEE International Conference on*, 2006, pp. 621–624.

[12] M. N. Schmidt and R. K. Olsson, "Single-channel speech separation using sparse non-negative matrix factorization," in *Spoken Language Processing ISCA International Conference on INTERSPEECH*, 2006, vol. 2, pp. 2–5.

[13] T. Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 3, pp. 1066–1074, 2007.

[14] R. Hennequin, R. Badeau, and B. David, "Time-dependent parametric and harmonic templates in non-negative matrix factorization," in *Proc. of the 13th International Conference on Digital Audio Effects (DAFx)*, 2010, number 1, pp. 1–8.

[15] H. Kameoka, N. Ono, K. Kashino, and S. Sagayama, "Complex NMF: A new sparse representation for acoustic signals," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2009, vol. 1, pp. 2353–2356.

[16] B. J. King and L. Atlas, "Single-Channel Source Separation Using Complex Matrix Factorization," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 8, pp. 2591–2597, 2011.

[17] B. J. King, *New methods of complex matrix factorization for single-channel source separation and analysis*, Ph.D. thesis, University of Washington, 2012.

[18] J. Bronson and P. Depalle, "Phase constrained complex NMF: Separating overlapping partials in mixtures of harmonic musical sources," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2014, pp. 7475–7479.

[19] E. Vincent, R. Gribonval, and C. Fevotte, "Performance measurement in blind audio source separation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 4, pp. 1462–1469, 2006.

[20] ETSI TR101290, "Digital video broadcasting (DVB); measurement guidelines for DVB systems," 2001.

[21] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Music genre database and musical instrument sound database," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2003, number October, pp. 229–230.

[22] Y. J. Lin, "Music Files of SEPARATION OF MUSICAL NOTES WITH HIGHLY OVERLAPPING PARTIALS USING PHASE AND TEMPORAL CONSTRAINED COMPLEX MATRIX FACTORIZATION," 2015.

AUTOMATIC CALIBRATION AND EQUALIZATION OF A LINE ARRAY SYSTEM

Fernando Vidal Wagner and Vesa Välimäki
Department of Signal Processing and Acoustics
Aalto University
Espoo, Finland
fvidalwagner@gmail.com vesa.valimaki@aalto.fi

ABSTRACT

This paper presents an automated Public Address processing unit, using delay and magnitude response adjustment. The aim is to achieve a flat frequency response and delay adjustment between different physically-placed speakers at the measuring point, which is nowadays usually made manually by the sound technician. The adjustment is obtained using three signal processing operations to the audio signal: time delay adjustment, crossover filtering, and graphic equalization. The automation is in the calculation of different parameter sets: estimation of the time delay, the selection of a suitable crossover frequency, and calculation of the gains for a third-octave graphic equalizer. These automatic methods reduce time and effort in the calibration of line-array PA systems, since only three sine sweeps must be played through the sound system. Measurements have been conducted in an anechoic chamber using a 1:10 scale model of a line array system to verify the functioning of the automatic calibration and equalization methods.

1. INTRODUCTION

During the last decade, the music industry business model has shifted from record releasing to promote live performances, raising the number and quality of concerts. Live audio systems have become more complex due to the computer-controlled digital signal processing part and the acoustical improvements using innovative loudspeaker systems.

The vast majority of concert Public Address (PA) systems used nowadays consist of hanging line arrays, which help reduce acoustic shadows and increase the distance of the line source effect [1, 2]. These speakers are used to reproduce the middle and high frequencies, usually above 100 Hz to 150 Hz. For the low frequency coverage different subwoofer configurations are used, which are usually placed on the floor in front of the stage.

These loudspeaker configurations requires various signal processing techniques in order to achieve a flat and coherent response. Basically the calibration consists of three operations: apply a crossover to split the audio band between the subwoofers and the line array, adjust a time delay [3] and equalize the complete system to achieve a flat magnitude response.

Figure 1 shows a two-dimensional diagram with approximate values of the relative delay problem. Different arrival times of the wavefront from the subwoofers and the line array speakers, which are physically at different positions provoke phase shading, strongly noticed in the crossover band. As line array elements can be divided in different vertical sections for short, mid or long throws, the system adjustment is also possible at several measuring positions, placed at different distances from the stage. In our work, just one measuring point has been taken in to account. The target area is usually placed at the Front of House position [4].

A similar phenomenon occurs between the loudspeaker components in the line array, which can be treated with different DSP procedures to compensate those problems [5], but the relative position between the array elements is limited. Lots of improvements using DSP and Wave Field Synthesis have been done the last years to control the response and directivity of line arrays [6, 7] but the complete system tuning has not had such attention. In the case of the subwoofers and the line array, those can be placed at very different positions and configurations depending on the venue, the characteristics of the stage, and the space where it takes place.

Nowadays there are several automated systems, though they are usually bundled to a specific brand with pre-loaded speaker data, such as Meyer Sound's Galileo [8], or need additional tools to integrate it with the system processor. The consequence of this is that most of the small and mid sized line array systems are still adjusted manually using PA processing units in addition to graphic equalizers. The manual procedure requires to play different excitation signals, usually pseudorandom [9] pink-spectrum noise through the PA, which is disturbing for the audience and time consuming. For this reason it has to be adjusted and configured hours prior to the venue. The sound engineer adjusts then the crossover frequency and filter type, the delay of the different loudspeakers and the graphic equalizer supported by a spectrum and phase analyzer. Some examples are EASERA [10] and Smaart [11].

In this paper we explain a method to automatically perform the adjustment of those three operations playing three sweep signals through the PA. This avoids all the manual procedure, and it can be done even with audience, as spreading the spectral energy along time and makes it less noticeable and disturbing than pseudorandom noise, being therefore much more time and effort

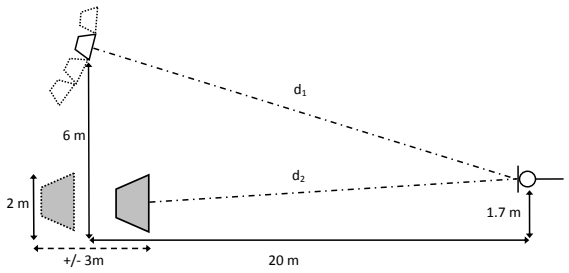


Figure 1: *Relative distances to measuring point.*

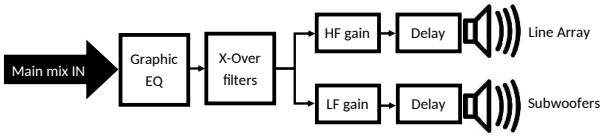


Figure 2: *One channel of the signal processing chain for a PA.*

efficient.

In order to evaluate the system, a 1:10 factor scale model was tested in an anechoic chamber. Thus, the distances were divided by 10, and the frequencies were multiplied by 10. To be able to work with standard audio material and avoid problems due to increased air absorption and distortions in ultrasound band, the system has been tested with frequencies up to 20 kHz in the scale model. This limitation is affordable as the main phase, crossover and equalization issues happen in the crossover and middle band. Therefore the results shown in the paper have to be analyzed as 10 times scaled to a real model and band limited to 2 kHz in a real application.

This paper is organized as follows. In Section 2, the calibration system and the test setup are explained. Section 3 describes the details of signal processing operations. Section 4 discusses the ground reflection problem and how it can be treated. Conclusions and future applications are explained in Section 5.

2. SYSTEM OVERVIEW

In this section a brief explanation of the different steps in the signal processing and parameter calculation is given. In Section 2.1 an introduction to the actual processing operations applied to the audio signal is made. In Sections 2.2 and 2.3 the different operations are explained in order to obtain automatically the parameters.

Three sweeps will be used in total. The first two sweeps collect separately the responses of the subwoofers and the line array to calculate the different audio processing chain parameters. The third sweep runs trough the designed processing chain and sounds through the whole PA for verifying and re-adjusting some parameters if necessary.

2.1. Audio processing chain

The automated processing unit is designed following the typical processing chain for PA systems shown in Figure 2. The main output signal of the mixing desk is conducted through a graphic equalizer, in order to apply the equalization directly to the whole system. This is the usual way to equalize PA systems, as it is much more intuitive for the mixing engineer to have a graphic curve of the complete band instead of each component group separately.

The output of the graphic equalizer feeds the PA processing unit. The first step is applying the crossover filters to divide the audio band into the different sub-bands (or ways) for each speaker group or driver into a specific speaker. In this case two bands are used: mid and high frequencies (HF) to feed the line array and low frequencies (LF) to feed the subwoofers. Different filter types are used for this purpose, but the most widespread type for digital crossover filters are Linkwitz-Riley filters, as they obtain a flat pass-band response and zero phase difference in the crossover frequency [12].

Once the signal has been split into two bands, individual processing for the low frequency and high frequency cabinets can be applied. At this point, the delay is applied in order to acoustically align the speakers at a precise point, and to achieve a coherent response in the crossover band. Also an overall gain is applied to each output.

2.2. Target responses

As in this case the desired output (flat response and phase coherent) is known, in order to be able to generate an adequate target response for each of the processing steps, the reverse procedure has been followed. A block diagram is shown in Figure 3.

To obtain a distortion-free response of the different loudspeaker ways [13], two logarithmic sweeps have been used to obtain separately the responses of the subwoofers and the line array. The impulse responses are obtained using de-convolution and using the second half part, which corresponds to the linear part [14].

Once the impulse responses have been obtained, the group delay and the frequency response are analyzed to extract the group delay values for each band and the crossover frequency. The signals are compensated to zero-time applying an inverse delay. Afterwards, the responses are filtered with the designed crossover filter.

An average gain is applied to each way before summing them to obtain the full band signal. The frequency response of the full band signal is used to calculate the gains of the graphic equalizer.

With these operations, all the needed parameters to adjust the PA are obtained. In order to simplify the system, the crossover filter type is not automatically adjusted, instead a fourth-order Linkwitz-Riley IIR filter has been designed and implemented with standard `Matlab` filter design functions. Thus, at the cutoff frequency, both ways are attenuated 6 dB and the filter presents a decay of 24 dB per octave.

2.3. Verification and re-adjusting

In order to verify the suitability of the calculated parameters and correct errors, caused mostly by ground reflections, a third full sweep though the whole PA (line array and subwoofers) is played. A block diagram is shown in Figure 4.

The parameters calculated in Section 2.2 are used to design a signal processing chain as the one explained in Section 2.1. The same sweep signal as the used in the first step is fed as input signal.

The response of the full measurement is compared to the expected output calculated at the output of the graphic equalizer. The difference signal between these two sweeps is used to readjust the graphic equalizer if needed.

The difference between these signals is mostly caused by reflections in the measurement which could not been canceled with pre-processing of the responses. In this case, if the graphic equalizer tries to equalize out the generated comb filter, the affected band is set to nominal value, leaving this band un-equalized.

2.4. Scale model

A 1:10 scale model of the PA system has been implemented in an anechoic chamber. Different speaker configurations have been tested in addition to filters implemented in the sound interface to obtain a similar scaled response of a real PA system.

In Figure 5 the setup in the anechoic chamber is shown. The microphone is a quarter inch free field pressure microphone. A structure holds a tweeter emulating an array element and a sphere loudspeaker with a 5 inch cone emulating the subwoofer. The

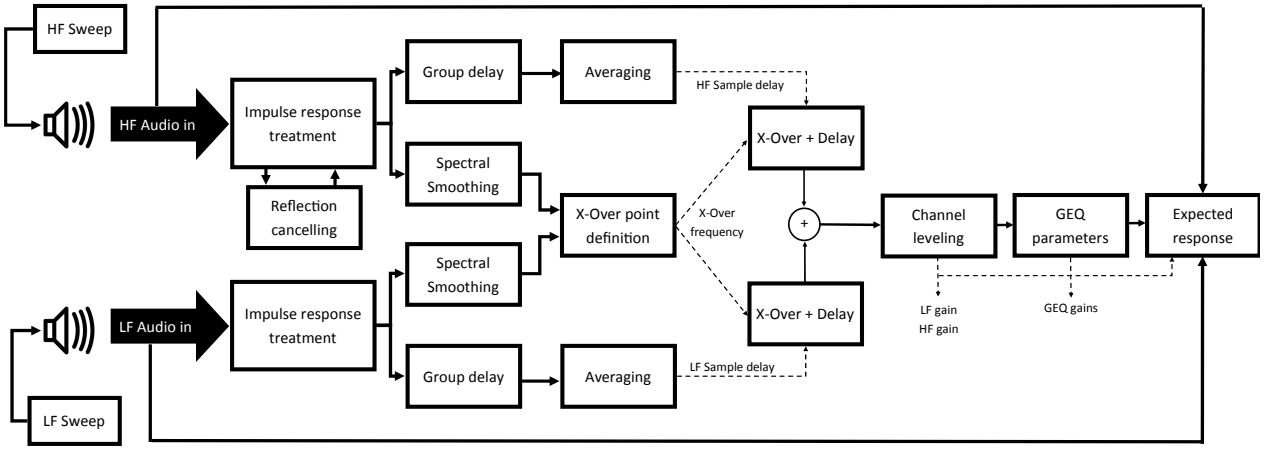


Figure 3: *Parameter calculation block diagram.*

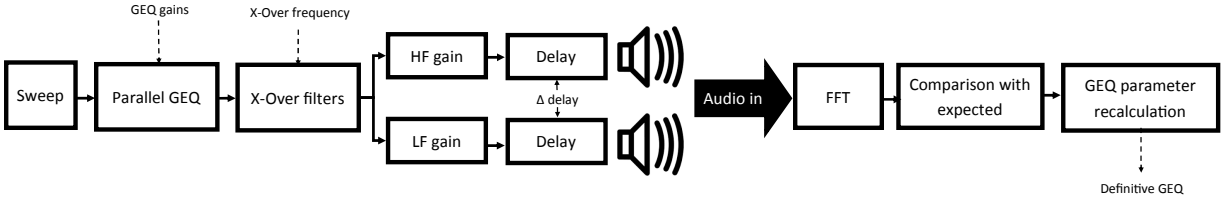


Figure 4: *Verification and re-adjusting block diagram.*

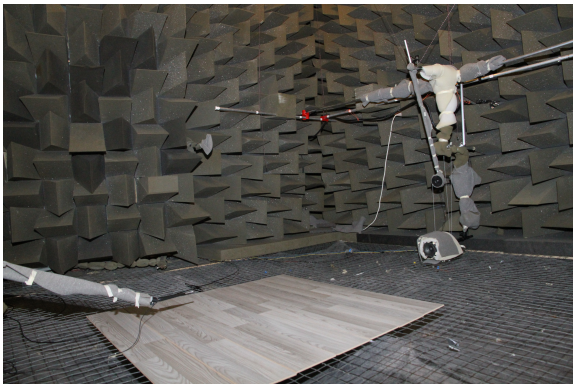


Figure 5: *Image of the scale model in the anechoic room. The bass and treble speakers are seen on the right, the microphone on the left, and a reflection plate below it.*

structure allows to move freely the high and low-frequency speakers to arrange different setups. In order to test the influence of the ground reflection, laminated wooden plates have been used.

The prediction is band limited from 300 Hz to 20 kHz, which equals a real case scenario from 30 Hz to 2 kHz. The sweeps have been performed from 200 Hz to 40 kHz during 3 seconds, as the used HF speaker has a reasnoable flat response up to this frequency, and could be equalized up to this band. A 96-kHz sampling frequency has been used for the anechoic measurements.

The fact of testing the system with a 1:10 scale model implies that the precision of the time related measures has to be also 10 times higher. Both measurements, with and without ground reflection, have been taken place.

The wooden panels used in the scale model exaggerate the intensity of a ground reflection in real case scenarios, with an average absorption coefficient around 0.4, which means that the reflected signal intensity is between -2 and -3 dB below the direct sound intensity.

3. CALIBRATION AND EQUALIZATION METHODS

In this section a detailed explanation of the procedures for obtaining the parameters explained in Section 2.2 is given.

3.1. Impulse response acquisition

As the system is designed to work in rough environments, a preprocessing of the measured responses is made to avoid undesired reflections to affect the measure and to compensate the measurement system errors.

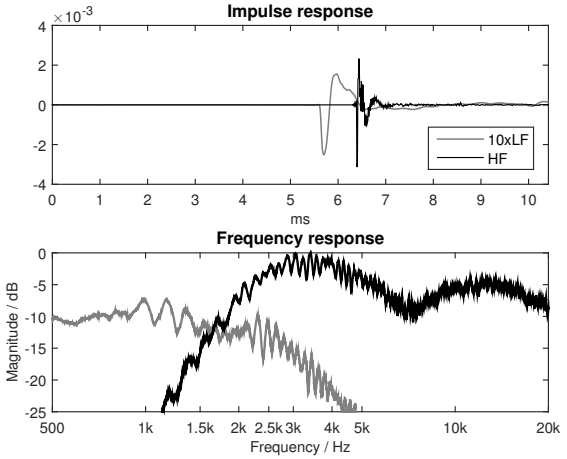


Figure 6: *(Top) Measured LF and HF impulse responses and (bottom) the corresponding magnitude frequency responses.*

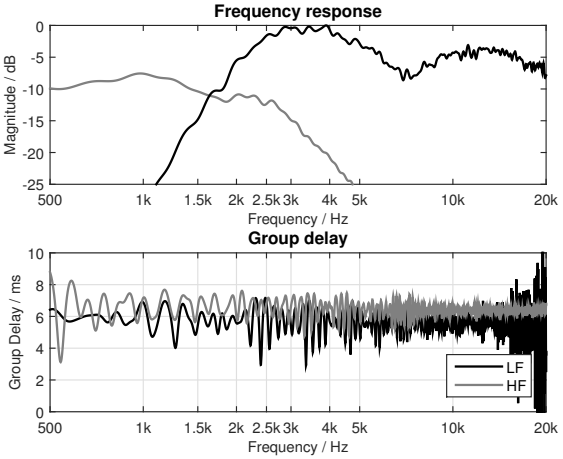


Figure 8: *(Top) Magnitude and (bottom) group-delay curves of the truncated LF and HF impulse responses.*

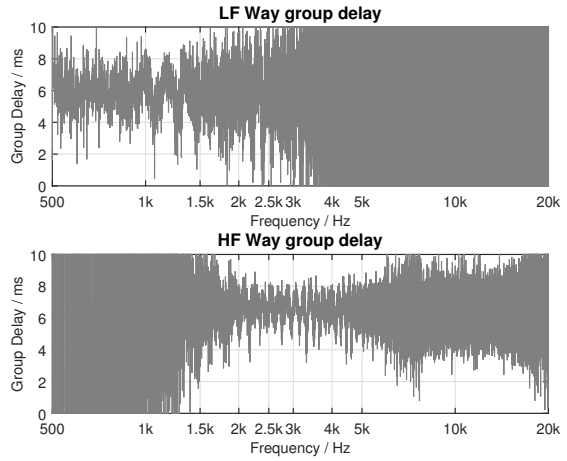


Figure 7: *Measured group delays without further processing.*

An initial calibration of the system is made. The group delay added by the sound cards and signal flow from the measurement system is measured using a feedback loop. Afterwards it will be subtracted from the measured group delay.

The different shifting from the 0 time point in each response shown in Figure 6 is caused by the time the wavefront travels from the speaker to the microphone. Thus, the different arrival times are caused by the different distances to the measuring point, as shown in Figure 1.

The different distances have also influence in the group delay. The measured group delay is very noisy as shown in Figure 7. The frequency response ripple and noisy group delay are caused mostly by late reflections in the impulse response. To avoid these effects, the impulse responses are truncated to the minimum number of samples possible. The compromise is between the minimum representable frequency and the aim to avoid reflections. The minimum frequency in this case has been chosen 250 Hz, in order to have a margin below the 300 Hz where the prediction mechanism starts.

When truncating the response, the delay caused by the time of flight has to be taken into account, as the response starts with a certain delay depending on the distance of the speaker to the measuring point. A maximum distance has to be defined in order to add this to the calculated minimum impulse response length. In this case 5 meters have been chosen, which in real case scenario would allow to do measurements up to 50 meters distance. The final length in samples is calculated as:

$$N_{imp} = \frac{f_s}{f_{min}} + \frac{D_{max} f_s}{c}, \tag{1}$$

where f_s stands for the sampling frequency, c for the speed of sound, f_{min} is the minimum representable frequency, D_{max} is the maximum measurable distance, and N_{imp} is the number of samples of the impulse response.

In Figure 8 the improvement on the group delay and the frequency response is shown. These responses will be used to compute the time delay for each way and to find the crossover point for the filters.

3.2. Time alignment

The time delay is calculated from the group delay. To compensate the measuring system delay, caused by filters and A/D converters, it is subtracted from the measured response in a calibration process;

$$\tau_g(f) = -\frac{d\phi_{meas}(f)}{df} + \frac{d\phi_{calib}(f)}{df}, \tag{2}$$

where $\phi_{meas}(f)$ is the phase of the measured response and $\phi_{calib}(f)$ stands for the phase of the calibration response.

In order to get accurate values of the time delay, the group delay of each way is averaged in the frequency range where the energy is located.

To achieve fine tuning in the calculated delay, fractional delay lines have been implemented using a linear interpolation filter [15]. As the linear interpolator, which is a two-tap FIR filter, has a lowpass magnitude response, it has only been implemented in the LF band. The fractional part of the HF band is subtracted from the LF band to compensate the relative delay. Also some offset samples are subtracted from the calculated time delay in order not to truncate the impulse response peak. The delay values are calculated as:

$$D_{HF}\prime = \frac{\sum_{f_1}^{f_2} \tau_g(f)}{f_2 - f_1} - N_{HF}, \tag{3}$$

$$D_{HF} = \lfloor D_{HF}\prime \rfloor, \tag{4}$$

$$D_{LF} = \frac{\sum_{f_3}^{f_4} \tau_g(f)}{f_4 - f_3} - N_{HF} - (D_{HF}\prime - D_{HF}), \tag{5}$$

where *N* stands for the offset in samples and *D* is the delay amount in samples. The values of f_3 and f_4 are 1 and 5 kHz for LF and f_1 and f_2 are 2 and 10 kHz in the HF band. These limits are chosen as the signals contain its energy this range, thus the group delay has valid values. In Table 1 a comparison between real measured distances and calculated via group delay averaging is presented.

The error is as low as 5 mm for high frequencies and 21 mm for low frequencies. The overall error is about 3 cm, which equals half wavelength at about 5.7 kHz, far beyond the interaction band of both ways.

Once the delay values have been determined, the signals are truncated according to the group delay adjusting them to the zero point. In Figure 9 the compensated impulse responses and the group delay are shown. The group delay is not exactly zero and flat. The slope is caused by the loudspeakers, as they are not exactly linear or minimum phase.

3.3. Choosing the crossover frequency

The crossover frequency is defined from the smoothed frequency responses obtained by the truncated impulse responses. For the LF and HF frequency responses the peak frequency and its magnitude value is searched. From the peak frequency a search for the -6 dB point is performed. For the LF band, the search is made from the peak toward higher frequencies, and for the HF band from the peak toward lower frequencies. These are estimated the cutoff frequencies for the subwoofers and the line array as shown in Figure 10. A limited crossover frequency band is determined, to avoid that peaks in higher areas of the HF frequency response cause shifting in the calulated crossover frequency (in the scale model between 1 and 4 kHz).

The crossover point can be determined by choosing a frequency point between the cutoff frequencies, allowing to exploit more the subwoofers using higher frequencies or the line array using lower frequencies in this band. In the model case, different options have been tested, and the middle point on the linear frequency scale has been used as the crossover frequency.

Table 1: *Measured vs calculated distances in mm.*

Way	Measured	Calculated	Δ
LF	1978	1957	-21
HF	2305	2310	5

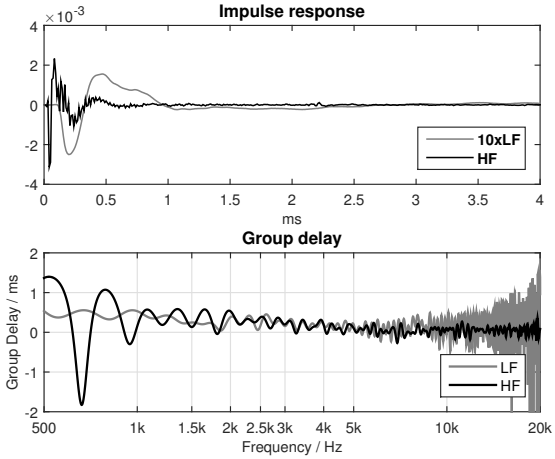


Figure 9: *(Top) Truncated and time aligned LF and HF impulses responses and (bottom) their group-delay curves.*

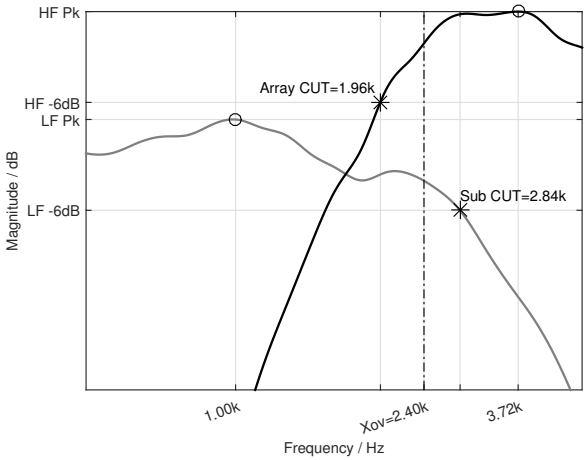


Figure 10: *Crossover frequency determination. The estimated peaks of LF and HF responses are marked with circles, and the –6-dB points are marked with asterisks. The selected crossover frequency is indicated with a vertical dash-dot line.*

At this point, the Linkwitz-Riley filters are designed and implemented. Two second-order Butterworth filters with the chosen cutoff frequency are chained. In Figure 11 the input frequency responses are shown as well as the filtered responses. Also it is observed that the group delays of the filtered signals are very close to each other in the crossover band.

Next an average gain is applied to each band before obtaining the full band signal. The average level values are calculated for each way. Then a general target level is calculated by averaging

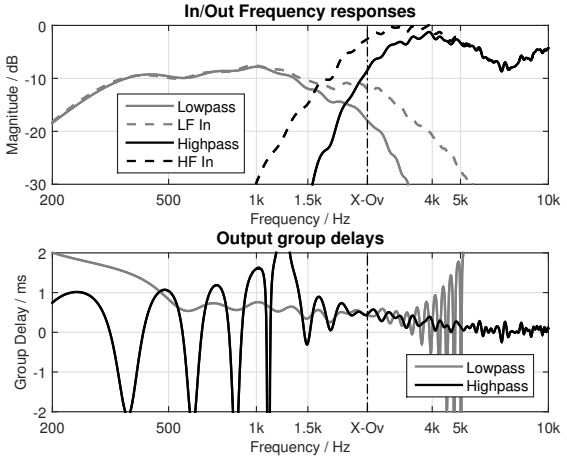


Figure 11: *(Top) Magnitude and (bottom) group-delay responses of the crossover highpass and lowpass filtered outputs. In the top plot, the responses before crossover filtering are given for reference. The vertical dash-dot line indicates the crossover frequency in this and in the following three figures.*

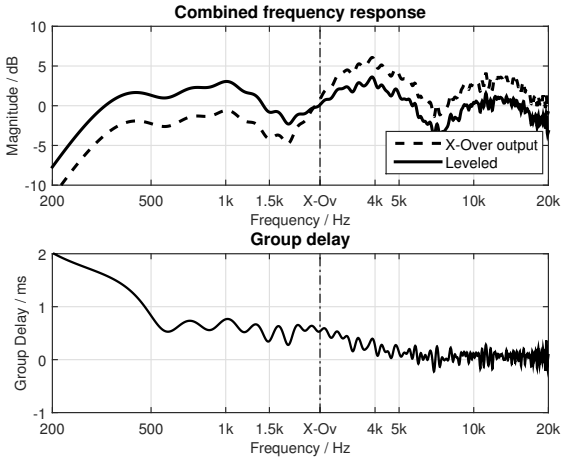


Figure 12: *(Top) Magnitude responses before and after gain correction (leveling) and (bottom) the full-band group delay.*

those two values levels. The gain is easily obtained by dividing the target level by its calculated level. Once the signals are gained, they are summed to obtain a full band signal.

As observed in Figure 6, the subwoofer band has less level than the line array, thus the LF band is amplified by 3.6 dB and the HF band is attenuated by 2.5 dB. The full band signal before and after amplifying each band is shown in Figure 12 It is also observed that the group delay maintains an acceptable ripple in the whole band. If the responses were not correctly aligned, a strong peak in the group delay would appear around the crossover frequency.

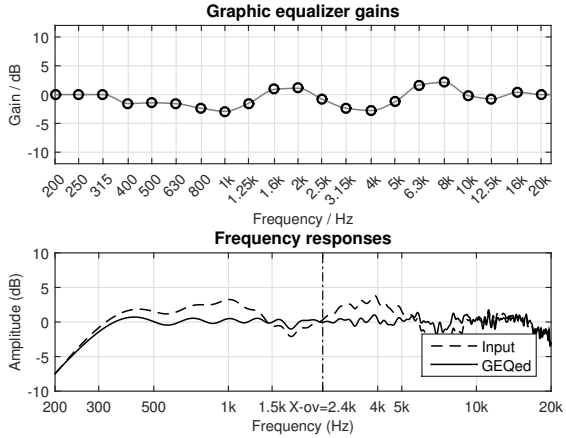


Figure 13: *(Top) Automatically calculated gains of the third-octave graphic equalizer and (bottom) the overall magnitude response before and after equalization.*

3.4. Graphic equalization

The graphic equalizer used for the PA equalization is a third-octave parallel design [16]. Precise amplitude and minimum phase characteristics make it the most suitable option for the fine tuning of the system.

The full band frequency response is used to obtain the graphic equalizer gains. First the average level of the full band signal is calculated. As the -3 dB frequency limits of each third-octave band of the graphic equalizer are known, the average level of each band is calculated and the gains are obtained by dividing the average level by each band level.

The frequency bands are limited from 315 Hz to 20 kHz. Figure 13 presents the obtained gains are presented and the comparison of the system frequency response before and after filtering with the parallel graphic equalizer.

3.5. Verification sweep

In order to verify that the system has the expected time alignment and frequency response, a third sweep is performed and processed with the processing chain in Figure 2. For determining the delay for each way, the difference between both time delays calculated in Section 3.2 is calculated. The way with the shortest time delay (i.e. the speaker closest to the measurement position - usually the subwoofer, as in Figure 1) is delayed by the number of samples of the calculated time difference. The fractional delay line is only implemented for the LF way.

In Figure 14 is shown that the system is time aligned as no peaks appear in the group delay and no cancellation occurs in the frequency domain. Additionally, the obtained frequency response is fairly linear, with ripple of less than +/- 2 dB in almost all the frequency band. Regarding the group delay, the values are below audible limits [17].

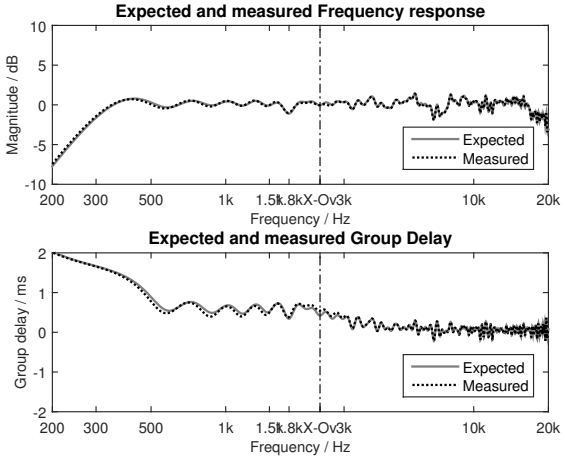


Figure 14: *(Top) Expected and measured overall magnitude response and (bottom) group delay.*

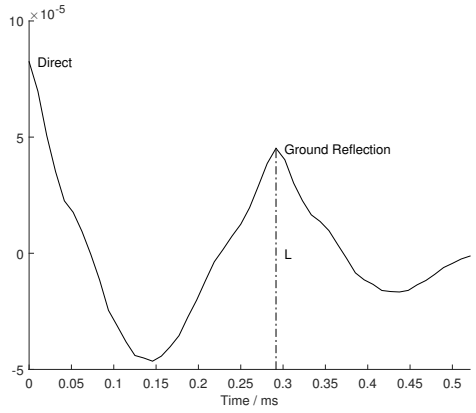


Figure 15: *Autocorrelation of input signal with ground reflection.*

4. GROUND REFLECTION ISSUES

As this system is initially designed for use in outdoor venues or arenas, the majority of reflective areas are far away from the measurement point, and the effects over the impulse response are shifted far from the direct sound impulse. This is not the case with the ground reflection, as the microphone is usually placed at ear height, 1 to 1.70 m depending if the audience is seated or standing. The ground reflections can affect the measurement creating a comb filter effect over the frequency response and a shifting in the group delay.

The upper frequency response in Figure 16 shows the influence of the ground plates used in the measurement. The extra distance of the ground reflection is approximately 17 cm, which is equivalent to a half wavelength approximately at 2 kHz, where the first notch appears.

To correct the comb filter effect of the reflection, a filter has been designed with an inverse response to the reflection. A first-order IIR filter, shown in Figure 17 creates a cancellation signal to the reflection. The parameters for this filter, delay *L* and coefficient

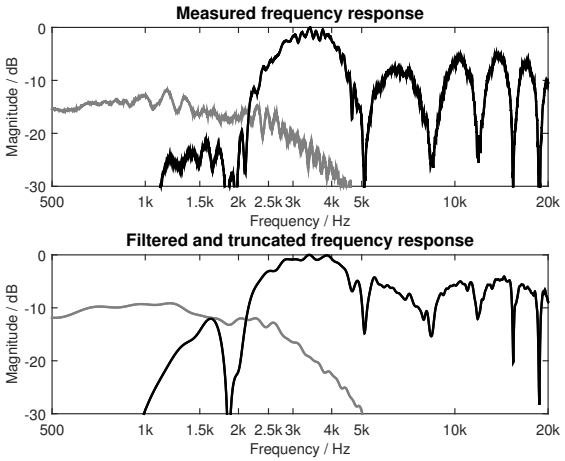


Figure 16: *(Top) Measured magnitude response with the reflection plate and (bottom) the corrected magnitude response.*

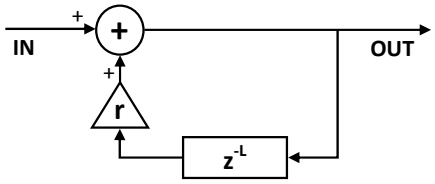


Figure 17: *Reflection cancellation filter.*

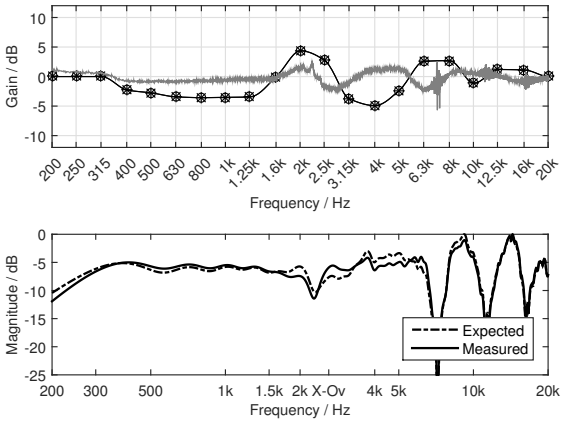


Figure 18: *(Top) Difference between the expected and measured magnitude responses and the calculated gains for the graphic equalizer: asterisks correspond to the first calculation and circles second calculation. (Bottom) Expected and measured magnitude responses.*

r, are obtained from the auto-correlation of the measured signal, shown in Figure 15. The distance between the zero-point peak and the second peak in the auto-correlation indicates the relative delay of the reflected signal, and therefore the delay *L* of the loop. The relative level between the the direct sound peak and the second peak determines the gain of the feedback loop *r*. This filter has been used as it is enough to obtain a proper frequency response to tune the system as shown in Figure 16.

The main influence of the comb filter effect is noticed in the automated graphic equalizer. To avoid that a notch or a bump in the frequency band falsify the gains of the equalizer, the full band verification sweep is used to compare both responses. The average difference between the expected and the measured band is calculated. If both, the difference and the equalizer gain in a certain band is higher than 2 dB, it is considered that the band has been falsified and the gain is set to 0 dB.

In Figure 18 the calculated equalization is made with the corrected response in Figure 16. If it is compared to the equalization without ground reflection in Figure 13, very similar responses are obtained. In this case no corrections to the graphic equalizer is done, as the difference signal between the expected signal and the measured signal is fairly low.

5. CONCLUSION

An automatic calibration and equalization system for line-array PAs has been presented in this paper. The procedure for calculating the different parameters has been tested using a 1:10 scale model, and the obtained results have been described and analyzer. Four sets of parameters have been successfully obtained from the subwoofer and line-array responses to time align and equalize the whole system: delay time for the closest cabinets, crossover frequency, average gain for each way, and graphic equalization parameters.

The obtained results are in the line with what we expected: a close to flat frequency response and low group delay can be obtained using phase (delay) adjustment in the crossover band. Such a system could be incorporated in PA processors making the line-array adjustment a much faster and easier task than it is today.

Even so, some improvements could be introduced to the system. Future work could be oriented in adapting the system for multi-channel operation, being able to equalize stereo or more channels (as central channel or outfills). Also subwoofer arrays could be adjusted with similar procedures, based on obtaining the response of each cabinet separately. This would allow to use phase adjustments to generate different subwoofer configurations and SPL patterns.

6. REFERENCES

[1] Mark S. Ureda, "Line arrays: Theory and applications," in *Audio Engineering Society Convention 110*, Amsterdam, The Netherlands, May 2001.

[2] Mark S. Ureda, "Analysis of loudspeaker line arrays," *J. Audio Eng. Soc.*, vol. 52, no. 5, pp. 467–495, 2004.

[3] Natàlia Milán and Joan Amate, "Time alignment of subwoofers in large PA systems," in *Audio Engineering Society Convention 130*, London, UK, May 2011.

[4] Anthony B. Kitson, "Equalisation of sound systems by ear and instruments: Similarities and differences," in *Audio Engineering Society 5th Australian Regional Convention*, Sydney, Australia, Mar. 1995.

[5] Ambrose Thompson, "Improved methods for controlling touring loudspeaker arrays," in *Audio Engineering Society Convention 127*, New York, NY, USA, Oct. 2009.

[6] Stefan Feistel, Mario Sempf, Kilian Köhler, and Holger Schmalle, "Adapting loudspeaker array radiation to the venue using numerical optimization of fir filters," in *Audio Engineering Society Convention 135*, New York, NY, USA, Oct. 2013.

[7] Frank Schultz, Till Rettberg, and Sascha Spors, "On spatial-aliasing-free sound field reproduction using finite length line source arrays," in *Audio Engineering Society Convention 137*, Los Angeles, CA, USA, Oct. 2014.

[8] "Meyer Sound Galileo System," Available at http://www.meyersound.com/, accessed May 08, 2015.

[9] Jeffrey Borish and James B. Angell, "An efficient algorithm for measuring the impulse response using pseudorandom noise," *J. Audio Eng. Soc.*, vol. 31, no. 7/8, pp. 478–488, 1983.

[10] "EASERA measurement platform," Available at http://easera.afmg.eu/, accessed May 08, 2015.

[11] "Smaart measurement tool," Available at http://www.rationalacoustics.com/smaart/about-smaart/, accessed May 08, 2015.

[12] Siegfried H. Linkwitz, "Active crossover networks for noncoincident drivers," *J. Audio Eng. Soc.*, vol. 24, no. 1, pp. 2–8, 1976.

[13] Angelo Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique," in *Audio Engineering Society Convention 108*, Paris, France, Feb. 2000.

[14] Thomas Kite, "Measurement of audio equipment with log-swept sine chirps," in *Audio Engineering Society Convention 117*, San Francisco, CA, USA, Oct 2004.

[15] T.I. Laakso, V. Välimäki, M. Karjalainen, and U.K. Laine, "Splitting the unit delay: Tools for fractional delay filter design," *IEEE Signal Processing Magazine*, vol. 13, no. 1, pp. 30–60, Jan. 1996.

[16] Jussi Rämö, Vesa Välimäki, and Balazs Bank, "High-precision parallel graphic equalizer," *IEEE/ACM Trans. Audio, Speech and Lang. Process.*, vol. 22, no. 12, pp. 1894–1904, Dec. 2014.

[17] Sheila Flanagan, Brian C. J. Moore, and Michael A. Stone, "Discrimination of group delay in clicklike signals presented via headphones and loudspeakers," *J. Audio Eng. Soc.*, vol. 53, no. 7/8, pp. 593–611, 2005.

AM/FM DAFX

Antonio José Homsi Goulart
Computer Science Department
University of São Paulo
São Paulo, Brazil
antonio.goulart@usp.br

Victor Lazzarini, Joseph Timoney
Sound and Digital Music Technology Group
Maynooth University
Maynooth, Co. Kildare, Ireland
[vlazzarini, jtimoney]@nuim.ie

ABSTRACT

In this work we explore audio effects based on the manipulation of estimated AM/FM decomposition of input signals, followed by resynthesis. The framework is based on an incoherent mono-component based decomposition. Contrary to reports that discourage the usage of this simple scenario, our results have shown that the artefacts introduced in the audio produced are acceptable and not even noticeable in some cases. Useful and musically interesting effects were obtained in this study, illustrated with audio samples that accompany the text. We also make available Octave code for future experiments and new Csound opcodes for real-time implementations.

1. INTRODUCTION

Analysis-synthesis methods are a common way of implementing digital audio effects. The Vocoder [1] is a good example of this approach. In this ubiquitous audio effect, the signal spectrum is broken down in a number of sub-bands, and effects are implemented by spectral manipulation followed by resynthesis. The success of the Vocoder paradigm can be assessed by the various applications [2] and the number of refinements [3, 4] proposed since its basic conception.

In this work we explore digital audio effects based on AM/FM decomposition of signals, processing the estimated signals for the envelope and instantaneous frequency, followed by resynthesis using these modified signals. This work follows a previous study where we derived selected psychoacoustic metrics in order to assess the extent of the modifications imposed by different configurations of smoothers acting on the AM and FM estimations of sample waveforms taken from an analog synthesizer [5]. Results showed that there is potential in creating different audio effects using the technique.

Recent surveys, such as the one in [6], have considered the level where the transformations in the signal are applied. Operations directly on the signal, in time-domain, are seen as surface operations, since sound is represented by the waveform samples. Effects like chorus, delay lines and distortion are good examples of such operations. On the other hand, systems based on analysis/-transformation/resynthesis work at a higher level with regards to how far the sound representation is unfolded (transformed). Thus, it can be accurately refined in a new domain, followed by the related inverse transform. AM/FM processing can de understood as a mix of both levels, as the input signal is transformed to a joint time-domain representation and "surface operations" are applied to the new signals. To put it in another way, the AM/FM decomposition unravels the mechanics of a (time-domain) signal in a couple of time-domain signals, and then we apply effects distorting these characteristics. Some examples in this paper are inspired on classic audio effects, but we highlight that the processing is applied in this different AM/FM domain. Hence, the effects obtained are not equivalent to the original ones, but can illustrate some possibilities with the technique. We also expect new kinds of effects to be developed within the framework.

AM/FM analysis is a topic extensively explored in the time-frequency literature. However, most of the studies are based on speech analysis, either with the intention of deriving parameters for analysis of speech, or for compression, transmission and resynthesis goals. Only a few works deal with audio as a broader scope, or specifically with music.

One line of work that explores music signals is Modulation Filtering, which is defined as the process of modifying a sub-band signal by filtering its estimated modulator and recombining the results with the original estimated carrier [7]. The series of works regarding the Modulation Vocoder [8, 9] explore a similar technique. The input signal is separated in bands based on center of gravities (COG), followed by a sub-band AM/FM decomposition. The goal is to achieve selective pitch transposition, which they implement by multiplying the FM envelopes in all bands by a same value. The paper reports supreme quality for the transposition, compared to commercial packages, but an inferior timbre preservation. Pitch transposition is the only effect discussed, and we think there is space for further exploration within this and similar frameworks.

In this paper, however, we study the decomposition of the whole signal into AM/FM as a means for implementing a variety of audio effects. We hope to demonstrate that this simpler approach can yield a number of useful transformations, which are economical in computational terms (if compared, for instance, to the sub-band methods), and can be implemented in real-time. In order to characterise the effects presented here, we employed a short guitar lick consisting of a bend followed by a vibrato. This signal was chosen for its smooth but varying frequency evolution, which can be observed in its spectrogram (Figure 1).

This paper is organised as follows. We first discuss our decomposition scheme, following to a description of new audio effects based on this decomposition. Then we present some reflections regarding the method and conclude the text. Audio examples with the sonorities obtained are available[1] alongside code for implementing the techniques. Octave [10] programs were used during the development of the work, and Csound [11] opcodes were written for real-time implementation of the techniques presented in the paper.

[1] http://www.ime.usp.br/~ag/dl/dafx15.zip

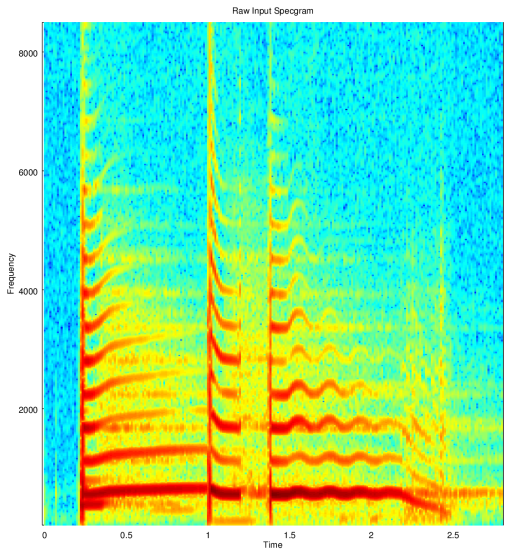


Figure 1: *Evolution of frequencies present in our test signal. Notice the bend through the first second, followed by a vibrato.*

2. AM/FM ANALYSIS

The AM/FM decomposition is a powerful method for the analysis of non-stationary signals [12]. For the mono-component case, the scheme presupposes the signal to be processed as a sole sinusoidal tone modulated both in amplitude and frequency. This follows the principle of Dudley's first speech synthesizer [13], in which speech and other non-stationary signals were interpreted as low-frequency processes that modulate higher frequency carriers representing fine structure [7]. This kind of decomposition conceptually involves two separate notions of frequency, that of the sinusoid carrier and another one for the modulating signal [7]. In other words, the decomposition consists of a transform of a one-dimensional broadband signal into a two-dimensional joint frequency representation [14].

Considering a signal

$$x(t) = m\cos(\omega t + \phi), \tag{1}$$

being m the amplitude, ω the frequency and ϕ the initial phase, the argument $(\omega t + \phi)$ is the instantaneous phase, and its derivative ω is the instantaneous frequency (IF) [15]. We could also have a signal modulated both in amplitude and frequency, as in [16]

$$x(t) = m(t)\cos(\theta(t)). \tag{2}$$

The phase derivative $\dot{\theta}(t)$ is the IF of the signal, and $m(t)$ its instantaneous amplitude (IA). The IF can be described as the frequency of the sinusoid that locally fits the signal at instant t [12].

The AM/FM signal analysis is intended to decompose a signal into functions for the AM (related to the IA signal) and the FM (related to the IF signal). A vast number of techniques exist for that [17, 16]. In this work we assume a mono component signal and apply the decomposition based on the analytic signal (AS). The AS, in turn, is based on the Hilbert Transform (HT) decomposition. The HT of a signal $x(t)$ is given by [12]

$$\hat{x}(t) = x(t) * \frac{1}{\pi t}. \tag{3}$$

This creates a 90° phase shifted version of the original, from which we build the analytic signal related to $x(t)$ as

$$z(t) = x(t) + j\hat{x}(t) = |z(t)|e^{j\theta(t)}, \tag{4}$$

where the AM/FM signals $a(t)$ and $f(t)$ are defined as

$$a(t) = |z(t)| \tag{5}$$

and

$$f(t) = \frac{d}{dt}\theta(t). \tag{6}$$

Another possibility for implementing the Hilbert Transform is by using filters, as described in [18]. A discrete-time Hilbert Transform can be approximated by a FIR filter with impulse response given by [18]

$$h(n) = \frac{1-\cos(\pi n)}{\pi n} = \begin{cases} \frac{2}{\pi n} & \text{for } n \text{ odd}, \\ 0 & \text{for } n \text{ even}. \end{cases} \tag{7}$$

As the filter need to be truncated in an appropriate length, an approximation of the theoretical transform will be achieved, and a short delay is necessary to make the filter causal.

Alternatively, it is possible to use a bank of allpass filters [19] for an efficient, latency-free, implementation of the HT. The allpass filters structure output a pair of signals with a 90° phase difference over a wide range of frequencies. The *hilbert* opcode[2] in Csound, used for the experiments in this paper, employs this algorithm.

For a signal of the form of (2), the absolute value of its associated analytic signal $z(t)$ can be used as an estimate for the AM portion, $a(t)$; the derivative of the unwrapped $\theta(t)$ can be used to estimate the FM portion, $f(t)$. The analysed signal can then be recreated using the expression

$$x(t) = a(t)\cos\left(2\pi\int_0^t f(\tau)d\tau\right) \tag{8}$$

For non real-time systems, real time systems with the acceptance of a delay long enough for a suitable frequency resolution, or off-line methods (such as our implementation in Octave), the Hilbert Transform can be implemented using the Fourier Transform. If $X(f)$ is the Fourier Transform of a real signal $x(t)$, the Fourier Transform of the analytic signal is given by

$$Z(f) = 2u(f)X(f), \tag{9}$$

where $u(f)$ is the unit-step signal [16]. By inverse transforming $Z(f)$ we get $z(t)$.

We are aware that a lot of criticism has been addressed to incoherent Hilbert based decompositions. Modulation filtering literature reports that the Hilbert based decomposition of arbitrary band-limited signals is often meaningless, as the instantaneous frequency estimation is discontinuous and with a bandwidth greater than the original signal one [7]. In [20] the authors argue that the

[2] `http://www.csounds.com/manual/html/hilbert.html`

model relating the IF to the analytic signal phase might be inadequate, and propose an alternative.

There is also an interesting discussion about what would be the best - or at least a suited - interpretation for the instantaneous frequency [21, 22]. Seeing it as the average frequency at each time is one possibility, but it only holds in the case of two components with the same amplitude [21]. It was shown that for a higher number of components, the equal-amplitude condition is not even sufficient [20]. Intuitively we can think that it makes sense to talk about IF for a mono-component signal, but not for multicomponent ones, even in the two equal amplitude component case.

So we should better understand what this IF "number" might mean. Regardless the complexity of the mathematical description of a signal, in most cases, if we step away from the math and observe the phenomenon we are able to notice an "instantaneous behaviour". For instance, it is easy to visualize this idea in a color changing light [21]. However, if we make a chord using two or more sinusoids we hear more than one frequency; but maybe we also hear this instantaneous metric related to the IF metric we get by applying the decomposition. Maybe by manipulating this quantity to achieve audio effects, more intuition can be achieved around this concept.

Yet another aspect regarding the decomposition of a signal is whether the signal is a mono- or multi-component one. Cohen [23] attempts to characterize this difference based on the instantaneous bandwidth of each ridge (related to the spread of the ridge) in a time-frequency plane description of a signal. He argues that if two ridges are close enough and present enough spread they coalesce and should be considered as a sole partial.

Another way to look at this is to consider the AM/FM decomposition as an analysis process that can be applied to a non-linear distortion synthesis scenario. The resynthesis is effectively a combination of both a heterodyne effect (amplitude modulation) and complex-signal (as opposed to sinusoidal) frequency modulation. It is in this context that we propose a number of possible DAFx.

The analysis-synthesis process is transparent: if we work with an unmodified pair of modulating functions, we are able to recover the original signal. Transforming the IA will lead to different types of modification of the original signal, as already reported in other contexts. While these changes might be undesirable in other applications, here there are no specific limitations beyond the intended aesthetic considerations of a good audio effect. Similarly, our modification of IF, which is behind most of the processes explored here, has a wide-ranging scope.

All of the effects discussed here can be implemented in real-time in Csound. Listing 1 shows how the analysis process is implemented in terms of user-defined opcodes (UDOs). The first UDO in the listing performs the phase differentiation and unwrapping, whereas the second is the actual full analysis process, which yields two outputs, the amplitude and frequency modulation signals. Depending on the input signal, we have observed that the approximation involved in the `hilbert` opcode implementation might sometimes result in some IF estimation errors. In this case, while reconstruction is still good, some artefacts might arise when the frequency modulation signal is scaled. Such issues are overcome, however, by the use of more precise HT calculation methods.

Listing 1: *AM/FM decomposition in Csound*

```
/* diff & unwrap */
opcode Udiff,a,a
        setksmps 1
 asig xin
 asig diff asig
 ksig = downsamp(asig)
 if ksig >= $M_PI then
  asig -= 2*$M_PI
 elseif ksig < -$M_PI then
  asig += 2*$M_PI
 endif
  xout asig
endop

/* AM/FM analysis */
opcode AmFmAnal,aa,a
  asig xin
  aim,are hilbert  asig
  am = sqrt(are^2 + aim^2)
  aph = taninv2(aim, are)
      xout  am, Udiff(aph)*sr/(2*$M_PI)
endop
```

3. AM/FM EFFECTS

In this section we show some effects experimented in this study. We focus on those which produced interesting sonorities. The effects are based on famous ones, but applied in this subverted paradigm of coping with the amplitude and frequency modulations descriptions of the signal.

After processing of the signals estimated by eqs.3 to 6 , the resynthesis (as defined in eq. 8) can be implemented in Csound with the code given in Listing 2. Alternatively, a sinusoidal oscillator will suffice.

Listing 2: *Csound resynthesis code*

```
/* AM/FM synthesis */
opcode AmFmSyn,a,aa
   am,aif xin
   xout am*cos(integ(aif)*2*$M_PI/sr)
endop

/* alternatively, just use an oscillator */
asig = oscili(am,aif)
```

3.1. OctIFer

The octaver effect mixes a signal with a transposed version of it, either an octave up or down. The OctIFer is an octaver-like process based on applying scaling to the estimated signal for the IF, before the integration for the resynthesis. As common sense might suggest, a simple multiplication of the IF by 0.5 produces a signal an octave lower than the original. That is indeed the case, but additionally to that a very interesting effect is introduced, that enriches the signal and this case resembles a very clean bass boost in a guitar amplifier. The spectrogram of the signal achieved in such a way is shown in Figure 2.

By doubling the IF, we would expect to get a signal that is an octave higher. However, the perception of the resultant signal is not a transposition, but instead, a beautiful coloration of the higher harmonics (this can be also visualized in the spectrogram in Figure 3). The resultant signal resembles a guitar single note played with a metal slide.

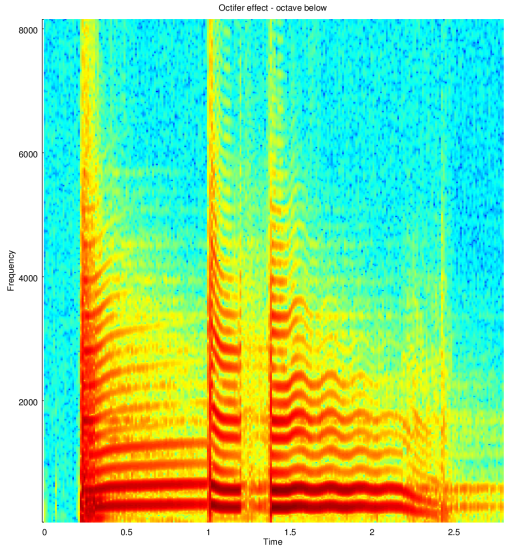


Figure 2: *Spectrogram of the octave below OctIFer effect. The lower partials introduced are clearly evident.*

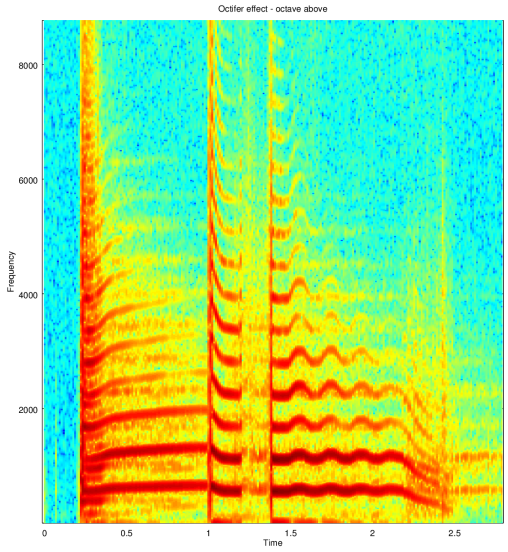


Figure 3: *Spectrogram of the octave up OctIFer effect. In comparison to the original sound spectrogram, a coloration can be observed.*

The reason for this effect is that while the frequency modulation signal is moved one octave, the amplitude modulation is unmodified. Since the synthesis expression is a ring modulation of the FM synthesis and the AM signal, we get a convolution of the two spectra. A similar process is at play in the other IF modification effects.

3.2. TransColorIFer

Values different than 2 or 0.5 might also be used to achieve general timbral transformations with the TransColorIFer technique. For instance by multiplying the IF by the relation 3:2 the effect introduced will add a coloration so the resultant signal bears resemblance to the fifth of the original note played. In such a way, a possible application is to build signals which are sums of signals resembling notes of specific chords. Each of the voices will have a different coloration and timbre, gently uncorrelated, leading to pleasant groupings. In the audio examples an example of a dominant (just minor, based on 9:5) 7th chord can give a good taste of that.

This same explanation holds for the case of the up-OctIFer. The original pitch is kept by the unprocessed IA signal, and the coloration only alters the timbre of the sound. In the case of the down-OctIFer, lower components are introduced, while the other harmonics introduced are aligned with the components present in the IA signal.

The expression for the TransColorIFer (and OctIFer) effect is given by

$$x(t) = a(t)\cos\left(r2\pi\int_0^t f(\tau)d\tau\right), \tag{10}$$

where r is the interval relation intended for the coloration.

3.3. HybridIFer

The cross-synthesis effect generates a sound based on the combination of two sound inputs [24], also sometimes referred to as *mutation between sounds*. With our HybridIFer effect we implement mutation between sounds by combining their instantaneous frequencies with complementary weights and keeping only one of the envelopes. Sounds obtained show that for the same two sounds, while using one of the envelopes can result in a noisy reconstruction, simply switching to the other can produce a clean outcome. Mix the two amplitude envelopes, instead of using only one, also tends to result in noisy outcomes.

The mutation types produced with the technique are interesting as they preserve the sonorities and characteristics of both sounds, also introducing a novel coloration. In the audio examples we provide mutations of all the combinations possible with a guitar, a saxophone, and a TB-303 synthesizer sample, using equal contribution of each sound IF estimation ($p = 0.5$). The spectrogram in Figure 4 show the spectrogram of a saxophone signal we used to combine with the guitar sound. The resultant spectrogram is shown in Figure 5.

The expression to implement the HybridIFer is given by

$$x(t) = a(t)\cos\left(2\pi\int_0^t (pf_1(\tau) + (1-p)f_2(\tau))d\tau\right), \tag{11}$$

where $f_1(t)$ and $f_2(t)$ are the IF for the two signals to be hybridized, $a(t)$ is the amplitude envelope for one of these signals and $0 \leq p \leq 1$.

It is interesting to note that this technique resembles the synthesis scheme of Double FM modulation [25], where a sum of sinusoids modulate a common carrier, where in our case the modulators are IFs. Double FM produces many sum and difference

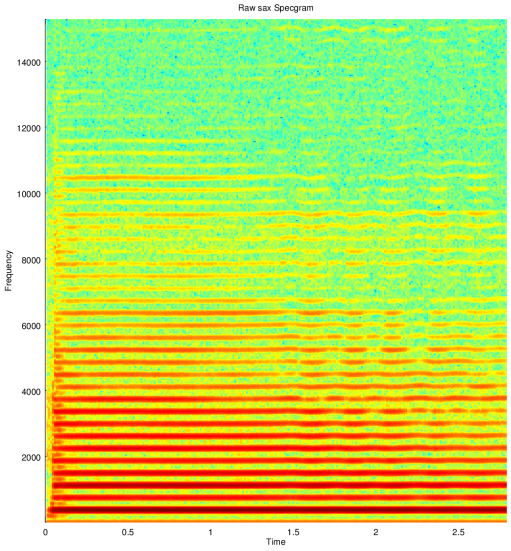


Figure 4: *Spectrogram of a saxophone sample.*

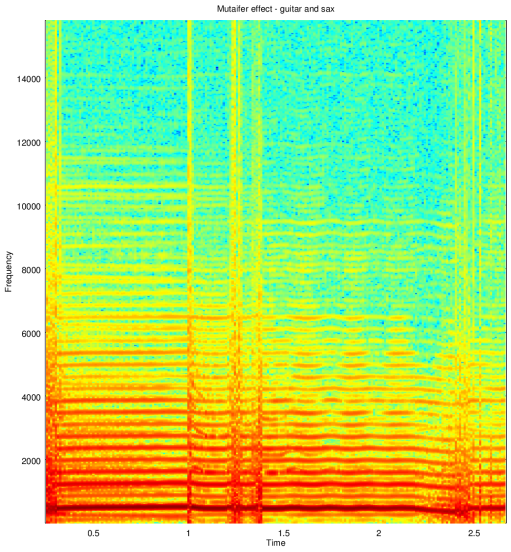


Figure 5: *Spectrogram of the guitar and sax combination using the HybridIFer effect.*

frequency components that are related to the frequencies of the modulators. It has a connection to the well-known AM technique of Ring Modulation [26, 27] that also produces sum and difference outputs.

3.4. ChorIFer

Chorusing is an effect based on the sum of slightly delayed and detuned versions of a signal. The resultant signal is perceived as a choir of the original sound. Due to the small differences in frequencies and phases, beatings and modulations in the resultant signal are very evident within this effect. Some guitar pedals in the market are even labeled as chorus/vibrato.

The ChorIFer is our version of the chorus effect, where the detunements are obtained with the method described in Sections 3.1 and 3.2. The resynthesized signal sounds feature a clean vibrato for low detunements values, up to 1% of the IF (this can be visualized in the spectrogram shown in Figure 6). For larger values, partials due to frequency modulation are also introduced and an interesting more dramatic variation is achieved. An effect similar to that is the animation effect in analog subtractive synthesizers, where closely detuned versions of the main oscillator are added [28].

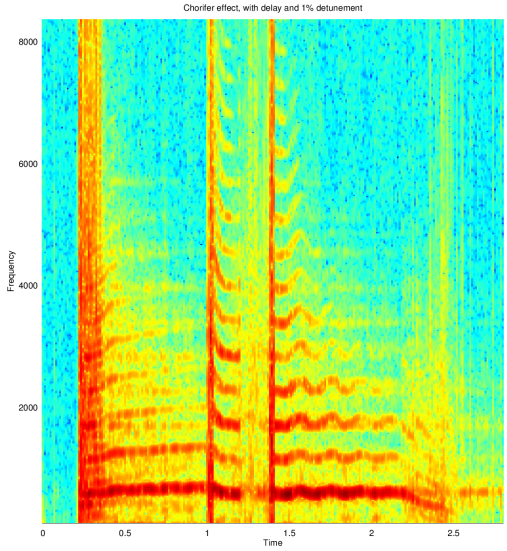


Figure 6: *Spectrogram of ChorIFer effect.*

It is important to be aware that the AM/FM decomposition produces synchronized IA and IF signals. By delaying the IF signal we are desynchronizing the IA and IF, and this can cause the sound to have a noisier attack. Both delayed and non-delayed versions are available in the examples. The delay periods experimented were around 20 milliseconds (the usual range for the chorus is from 10 to 25 ms [24]).

The ChorIFer mathematical expression is given by

$$x(t) = \frac{1}{2N+1} a(t) \sum_{k=-N}^{N} \cos\left(2\pi(1+kD)\int_0^t f(\tau - L_k)d\tau\right), \tag{12}$$

where $2N+1$ is the number of signals added to achieve the effect, D is a detunement value, and L_k is the delay for each version of the signal.

3.5. WahIFer

The classic and ubiquitous wah-wah effect is produced by adding to a signal a bandpass version of it, where the center frequency of a narrow bandpass varies over time [24]. Typically the center frequency is foot controlled with a pedal, but an alternative is to use a low frequency oscillator for it. This configuration is known as the auto-wah [24].

In the WahIFer the bandpass filtering is applied to the IF estimated signal. After that this processed IF is used along with the IA for the resynthesis, and added to the input signal. Notice that in the WahIFer the bandpass is not cutting out frequencies directly from an audio signal, but by smoothing the IF function. In such a way, the filter controls the generation of partials resultant of the frequency modulation process. In our experiments, frequencies around 1 Hz lead to the more interesting sonorities. However, even with this low value for the wah-wah filter swap artefacts of frequency modulation are present (Figure 7). Signals produced with higher values features noise and aliasing. Different sets of values are illustrated in the audio examples.

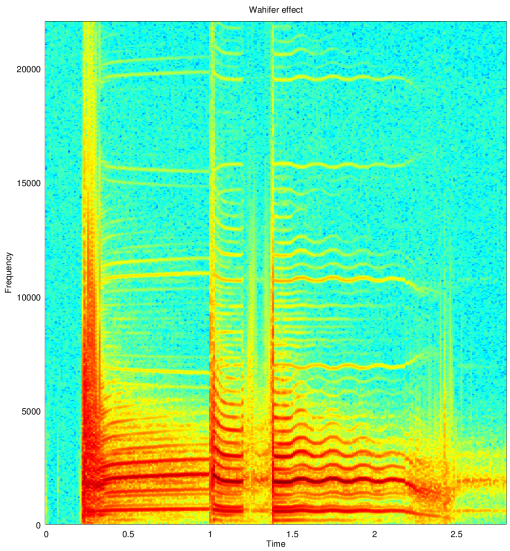


Figure 7: *Spectrogram of WahIFer effect.*

The expression for the WahIFer effect is given by

$$x(t) = a(t)\cos\left(2\pi \int_0^t h(\tau) * f(\tau) d\tau\right), \qquad (13)$$

where $*$ is the convolution operator and $h(t)$ is the impulse response for a wah-wah filter. A comprehensive description of a possible implementation of a wah-wah filter is described in [24]. We used this filter in our experiments.

4. CONCLUSIONS

In this work we presented different versions of long known audio effects. Instead of applying the processing directly on the signal, we decompose it into an amplitude envelope and instantaneous frequency using the Hilbert Transform. Then we process the IF signal and resynthesise the signal. The use of AM/FM decomposition within the context of non-linear sound synthesis techniques can open up a variety of avenues for DAFx design, some of which we have presented here. As we are applying these effects to analysis signals, the outcome is different from the application in their usual context as "surface" operations. For instance, in the WahIFer technique, the filter acts in the signal which ultimately is used in a non-linear operation, and thus it has an indirect effect on the output. While most of the effects discussed are achieved by manipulation of the IF, it is possible to envisage a whole suite of transformations that will act on the IA.

While this kind of demodulation has been criticised by the speech processing literature, it yields some interesting possibilities when brought into the context of DAFx for electronic music applications. In this environment, the requirements are not so narrowly defined (for instance, intelligibility is not a factor here), and a more creative approach to design sound transformations can be applied. We have demonstrated that there is enough scope for a variety of useful musical DAFx.

The main aim of this paper was to provide an initial exploration of AM/FM transformations. The ideas presented here can be used within any kind of single or multicomponent decomposition scheme, leading to different results according to the estimated signals. There is scope for the development of these techniques presented here in more complex systems, which would consider sub-band decomposition and coherent detection. In fact, multiband AM/FM audio effects based on different processing for the AM/FM estimations of each sub-band seem to be a rich tool for the creation of new DAFx and variation of classic ones. We understand that there is still space for further investigation, and encourage the use of the code that accompany the paper, so more AM/FM DAFx can be proposed.

5. ACKNOWLEDGMENTS

This work was partially supported by CAPES (proc. num. 8868-14-0) and was realized during Antonio Goulart's internship period at Maynooth University.

6. REFERENCES

[1] J. L. Flanagan and R. M. Golden, "Phase vocoder," *Bell System Technical Journal*, vol. 45, no. 9, 1966.

[2] Charles Dodge and Thomas Jerse, *Computer Music: Synthesis, composition and performance*, Schirmer Books, New York, NY, USA, second edition, 1997.

[3] Jean Laroche and Mark Dolson, "New phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects," in *Proceedings of ICASSP*, New Paltz, New York, October 1999.

[4] Alex Robel, "A new approach to transient processing in the phase vocoder," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-03)*, London, UK, 2003.

[5] Antonio Jose Homsi Goulart, Joseph Timoney, Victor Lazzarini, and Marcelo Gomes de Queiroz, "Psychoacoustic impact assessment of smoothed am/fm resonance signals," in

Proceedings of the Sound and Musical Computing Conference (submitted), Maynooth, Ireland, July 2015.

[6] Vincent Verfaille, Martin Holters, and Udo Zolzer, "Introduction," in *DAFX - Digital Audio Effects*, Udo Zolzer, Ed. Wiley, second edition, 2011.

[7] Pascal Clark and Les E. Atlas, "Time-frequency coherent modulation filtering of nonstationary signals," *IEEE Transactions on Signal Processing*, vol. 57, no. 11, november 2009.

[8] Sascha Disch and Bernd Edler, "An amplitude- and frequency-modulation vocoder for audio signal processing," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-08)*, Espoo, Finland, 2008.

[9] Sascha Disch and Bernd Edler, "An enhanced modulation vocoder for selective transposition of pitch," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, 2010.

[10] John Eaton, David Bateman, and Soren Hauberg, *GNU Octave Manual Version 3*, Network Theory Ltd., 2008.

[11] Richard Boulanger, Ed., *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming*, MIT Press, 2000.

[12] B. Boashash, "Estimating and interpreting the instantaneous frequency of a signal-part 1: Fundamentals," *Proceedings of the IEEE*, vol. 80, no. 4, Apr 1992.

[13] Homer Dudley, "Remaking speech," *Journal of the Acoustical Society of America*, vol. 11, no. 2, 1939.

[14] Steven Schimmel and Les Atlas, "Coherent envelope detection for modulation filtering of speech," in *Proceedings of ICASSP*, 2005, pp. 221–224.

[15] D. Gabor, "Theory of communication. part 1: The analysis of information," *Electrical Engineers - Part III: Radio and Communication Engineering, Journal of the Institution of*, vol. 93, no. 26, pp. 429–441, November 1946.

[16] Bernard Picinbono, "On instantaneous amplitude and phase of signals," *IEEE Transactions on Signal Processing*, vol. 45, no. 3, 1997.

[17] R. McAulay and T.F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 34, no. 4, pp. 744–754, Aug 1986.

[18] Pierre Dutilleux, Martin Holters, Sascha Disch, and Udo Zolzer, "Modulators and demodulators," in *DAFX - Digital Audio Effects*, Udo Zolzer, Ed. Wiley, second edition, 2011.

[19] M. Press, *Theory and Implementation of the Discrete Hilbert Transform*, pp. 14–42, MIT Press, 1969.

[20] P.M. Oliveira and V. Barroso, "Instantaneous frequency of mono and multicomponent signals," in *Time-Frequency and Time-Scale Analysis, 1998. Proceedings of the IEEE-SP International Symposium on*, Oct 1998, pp. 105–108.

[21] P.J. Loughlin and B. Tacer, "Instantaneous frequency and the conditional mean frequency of a signal," *Signal Processing*, vol. 60, no. 2, pp. 153–162, 1997.

[22] Wonchul Nho and Patrick Loughlin, "When is instantaneous frequency the average frequency at each time?," *IEEE Signal Processing Letters*, vol. 6, no. 4, April 1999.

[23] Leon Cohen, "What is a multicomponent signal?," in *Proceedings of ICASSP*, 1992.

[24] Udo Zolzer, Ed., *DAFX - Digital Audio Effects*, Wiley, second edition, 2011.

[25] B.T.G. Tan, S.L. Gan, S.M. Lim, and S.H. Tang, "Real-time implementation of double frequency modulation (dfm) synthesis," *Journal of the Audio Engineering Society*, vol. 42, no. 11, November 1994.

[26] Richard Hoffmann-Burchardi, "Digital simulation of the diode ring modulator for musical applications," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-08)*, Espoo, Finland, 2008.

[27] Julian Parker, "A simple digital model of the diode-based ring-modulator," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-11)*, Paris, France, 2011.

[28] Joseph Timoney, Victor Lazzarini, Jari Kleimola, and Vesa Valimaki, "Examining the oscillator waveform animation effect," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-14)*, Erlangen, Germany, September 2014.

ON COMPARISON OF PHASE ALIGNMENTS OF HARMONIC COMPONENTS

Wen Xue[1], *Lou Xiaoyan*[1], *Mark Sandler*[2]

Samsung Electronics[1]
Beijing, China

Queen Mary University of London[2]
London, UK

{xue.wen,xiaoyan.lou}@samsung.com[1],
mark.sandler@eecs.qmul.ac.uk[2]

ABSTRACT

This paper provides a method for comparing phase angles of harmonic sound sources. In particular, we propose an algorithm for decomposing the difference between two sets of phases into a harmonic part, which represents the phase progress of harmonic components, and a residue part, which represents all causes of deviations from perfect harmonicity. This decomposition allows us to compare phase alignments regardless of an arbitrary time shift, and handle harmonic and noise/inharmonic parts of the phase angle separately to improve existing algorithms that handles harmonic sound sources using phase measurements. These benefits are demonstrated with a new phase-based pitch marking algorithm and an improved time-scale and pitch modification scheme using traditional harmonic sinusoidal modelling.

1. INTRODUCTION

Pitched, or harmonic, sound sources produce periodical waveforms that can be represented as the sum of time-varying sinusoids (*partials*) whose frequencies are multiples of a fundamental *F*0[1][2]. This sinusoidal representation appears in audio processing either explicitly, e.g. in sinusoidal modelling[3], or implicitly, e.g. in the phase vocoder[4]. At any point each partial is associated with a *phase angle* that determines its positioning in time.

While each phase angle hardly makes any audible difference by itself, the alignment between phases affects the audio quality in various ways. Phase alignment between harmonic partials effects audible sub-period energy distribution [5], while that between binaural channels helps establish the perceived direction of the sound source [6]. A third type of phase alignment. i.e. that between phase angles sampled from the same partial at different time instants, affects sound quality via the frequency-phase relation. In particular, the perception of harmonicity relies on the partial frequencies being perfect multiples of a fundamental, which in turn requires the phase angles at different instants be aligned in a special way. An audio processing routine that does not preserve such phase alignment breaks the harmonicity, and possibly creates a chorus effect.

In this paper we address this third type of phase alignment in the context of pitched sounds. In particular, we compute a decomposition of the difference between two sets of phase angles into a harmonic progression and a least-square residue, the former evaluating how much progress the signal has made from one set of phases to the next, the latter evaluating whether the two sets are harmonically aligned, and if not, how far they are from being so. While being surprisingly simple, this treatment is useful in a variety of applications involving the handling of pitched sound sources.

2. HARMONIC PHASE ALIGNMENT AND MISALIGNMENT

Consider a set of M harmonically related sinusoids

$$s_m(t) = a_m \cdot \cos \varphi_m(t),\ t \in \mathbf{R},\ m = 1, \ldots, M, \tag{1a}$$

where

$$\varphi_m(t) = \varphi_m^0 + 2\pi m f_0 t,\ \forall m \tag{1b}$$

is the phase angle of the m^{th} harmonic *partial*. We define

$$\boldsymbol{\varphi}(t) = (\varphi_1, \ldots, \varphi_m)^{\mathsf{T}},\ \boldsymbol{\varphi}^0 = (\varphi_1^0, \ldots, \varphi_m^0)^{\mathsf{T}},\ \mathbf{m} = (1, \ldots, M)^{\mathsf{T}}. \tag{2}$$

It is trivial to show that

$$\boldsymbol{\varphi}(t) - \boldsymbol{\varphi}^0 = 2\pi f_0 t\ \mathbf{m}, \tag{3}$$

in which $f_0 t$ counts the number of periods between 0 and t. Phase values we encounter in actual computations are often subject to arbitrary modulo-2π shifts, so it's better to write (3) as

$$\boldsymbol{\varphi}(t) - \boldsymbol{\varphi}^0 = 2\pi f_0 t \cdot \mathbf{m} + 2\mathbf{k}\pi,\ \mathbf{k} \in \mathbf{Z}^M \tag{4a}$$

or

$$\boldsymbol{\varphi}(t) - \boldsymbol{\varphi}^0 \equiv 2\pi f_0 t \cdot \mathbf{m} \pmod{2\pi}. \tag{4b}$$

We say two vectors of phases $\boldsymbol{\varphi}_1 \in \mathbf{R}^M$ and $\boldsymbol{\varphi}_2 \in \mathbf{R}^M$ are *harmonically aligned* (*aligned* for short) if $\exists \delta \in \mathbf{R}$, so that

$$\boldsymbol{\varphi}_2 - \boldsymbol{\varphi}_1 \equiv \delta \cdot \mathbf{m} \pmod{2\pi}. \tag{5}$$

By this definition, the phase vector $\boldsymbol{\varphi}(t)$ in (4b) sampled at any time t is aligned to the initial $\boldsymbol{\varphi}^0$, while those sampled at any pair of t_1, t_2 are aligned between themselves. For various reasons, in real-world tasks the phase angles associated with harmonic sinusoids may not always satisfy (5), but carry an error term $\boldsymbol{\varepsilon}$:

$$\boldsymbol{\varphi}_2 - \boldsymbol{\varphi}_1 \equiv \delta \cdot \mathbf{m} + \boldsymbol{\varepsilon} \pmod{2\pi}. \tag{6}$$

We say $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$ in (6) are *harmonically misaligned* (*misaligned* for short) by $\boldsymbol{\varepsilon}$. The following statements are equivalent:

i. $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$ are harmonically misaligned by $\boldsymbol{\varepsilon}$;
ii. $\boldsymbol{\varphi}_2$ is harmonically aligned to $\boldsymbol{\varphi}_1 + \boldsymbol{\varepsilon}$;
iii. $\boldsymbol{\varphi}_2 - \boldsymbol{\varphi}_1$ is harmonically aligned to $\boldsymbol{\varepsilon}$;
iv. $\exists \delta \in \mathbf{R}$, $\mathbf{k} \in \mathbf{Z}^M$, so that

$$\boldsymbol{\varepsilon} = \boldsymbol{\varphi}_2 - \boldsymbol{\varphi}_1 - \delta \cdot \mathbf{m} + 2\mathbf{k}\pi. \tag{7}$$

Statement iv shows that (6) does not uniquely quantify $\boldsymbol{\varepsilon}$: any pair of *misalignments* $\boldsymbol{\varepsilon}_1$ and $\boldsymbol{\varepsilon}_2$ are equivalent as long as they are aligned to each other. To compare phase alignments quantitatively, we'd like to quantify $\boldsymbol{\varepsilon}$ so that smaller $\boldsymbol{\varepsilon}$ is associated with phase vectors closer to perfect alignment. Particularly, if $\boldsymbol{\varphi}_1$ and

$\boldsymbol{\varphi}_2$ are harmonically aligned then $\boldsymbol{\varepsilon}$ should be $\mathbf{0}$. This leads to the minimum misalignment detailed below.

3. MINIMUM HARMONIC PHASE MISALIGNMENT

Equation (7) gives the general form of misalignment between $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$. The minimum misalignment method seeks to minimize $\boldsymbol{\varepsilon}$, constrained by (7), as the unique quantification of $\boldsymbol{\varepsilon}$. Different forms of the minimum can be defined by specific choices of the minimization criterion. In this paper we consider two of them, based on L^2 and weighted L^2 norms, respectively.

3.1. Minimization by L^2

The L^2 norm of $\boldsymbol{\varepsilon}$ is

$$\|\boldsymbol{\varepsilon}\|^2 = \boldsymbol{\varepsilon}^{\mathsf{T}}\boldsymbol{\varepsilon}. \tag{8}$$

Define $\boldsymbol{\varepsilon}^o = \boldsymbol{\varphi}_2 - \boldsymbol{\varphi}_1$, we can rewrite (7) as

$$\boldsymbol{\varepsilon}(\delta, \mathbf{k}) = \boldsymbol{\varepsilon}^o - \delta\cdot\mathbf{m} + 2\mathbf{k}\pi. \tag{9}$$

Once δ is fixed, the minimization of $\|\boldsymbol{\varepsilon}\|^2$ with regard to $\mathbf{k}$ is trivial: we only need to take the minimal absolute residue of $\boldsymbol{\varepsilon}^o - \delta\cdot\mathbf{m}$ modulo 2π:

$$\min_{\mathbf{k}} \boldsymbol{\varepsilon}(\delta, \mathbf{k}) = \operatorname{res}(\boldsymbol{\varepsilon}^o - \delta\cdot\mathbf{m}, 2\pi), \tag{10}$$

where res($\mathbf{x}$, y) is the minimal absolute residue of $\mathbf{x}$ modulo y, obtained by shifting every entry of $\mathbf{x}$ by a multiple of y into the interval [-y/2, y/2). Since $\boldsymbol{\varepsilon}$ is periodical with regard to δ with period 2π, the task of minimizing $\|\boldsymbol{\varepsilon}\|^2$ is simplified to finding $\delta\in[-\pi, \pi)$ so that

$$\|\boldsymbol{\varepsilon}(\delta)\|^2 = \sum_{m=1}^{M} \operatorname{res}(\varepsilon_m^o - m\delta, 2\pi)^2 \tag{11}$$

becomes minimal, where ε_m^o is the m^{th} entry of $\boldsymbol{\varepsilon}^o$.

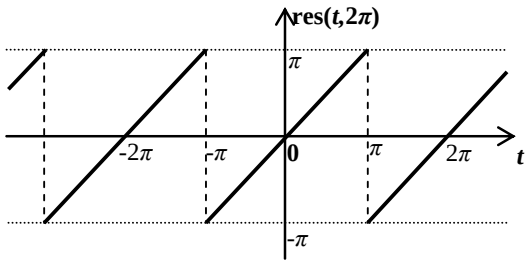


Figure 1 Minimal absolute residue modulo 2π

Since res(x,2π) is a piecewise linear function of x (Figure 1), $\|\boldsymbol{\varepsilon}(\delta)\|^2$ in (11) is a piecewise quadratic function of δ, whose minimum over the finite-length interval [$-\pi$, π) can be found by enumerating all quadratic pieces within this range.

3.2. Minimization by weighted L^2

L^2 minimization in 3.1 assumes equal impact from each partial. In practice it is often reasonable to emphasize some partials while deemphasizing some others. For example, some musical instruments have weaker even partials than odd ones, so that the phase angles measured for the odd partials are generally more reliable. It makes sense to emphasize the contribution from the stronger partials in formulating the minimization criterion.

The weighted L^2 norm of $\boldsymbol{\varepsilon}$ is

$$\|\boldsymbol{\varepsilon}\|_{\mathbf{w}}^2 = \boldsymbol{\varepsilon}^{\mathsf{T}}\cdot\operatorname{diag}(\mathbf{w})\cdot\boldsymbol{\varepsilon}. \tag{12}$$

where $\mathbf{w}=(w_1, \ldots, w_M)^{\mathsf{T}}$, $w_m\geq 0$, $\forall m$, contains the partial weights and diag($\mathbf{w}$) is a diagonal matrix whose main diagonal is specified by $\mathbf{w}$. $\|\boldsymbol{\varepsilon}\|_{\mathbf{w}}^2$ reduces to $\|\boldsymbol{\varepsilon}\|^2$ when all entries of $\mathbf{w}$ are 1.

The minimization of the weighted L^2 norm follows the same path as that of the L^2 norm. It is eventually reduced to finding $\delta\in[-\pi, \pi)$ that minimizes

$$\|\boldsymbol{\varepsilon}(\delta)\|_{\mathbf{w}}^2 = \sum_{m=1}^{M} w_m \cdot \operatorname{res}(\varepsilon_m^o - m\delta, 2\pi)^2 \tag{13}$$

A routine for the computation of δ in (11) is suggested in Appendix A, which also works for (11) if all weights are set to 1.

3.3. Interpretation

The minimum-by-L^2 method provides a least square solution to (6), which breaks the phase difference $\boldsymbol{\varphi}_2-\boldsymbol{\varphi}_1$ down to a harmonic progression $\delta\cdot\mathbf{m}$ and a least square mismatch term $\boldsymbol{\varepsilon}$.

If $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$ are sampled from the same harmonic source, then δ estimates the phase progress (modulo 2π) of the fundamental frequency between the sampled positions, and $\boldsymbol{\varepsilon}$ evaluates their difference in phase sampling error. If $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$ are sampled from different harmonic sources then δ estimates how much progress the first source has to make to be optimally aligned with the second source at the sampled position, and $\boldsymbol{\varepsilon}$ evaluates how well they can match in terms of phase alignments.

Although our discussion had started with steady tones, all formulations since (5) also apply to time-varying harmonic sound sources with amplitude and frequency modulations as well as timbre evolution. In the last case the phase vectors become landmarks during the transition from one timbre to the next, which can be used later to resynthesize the same transition.

Once we have estimated δ and $\boldsymbol{\varepsilon}$ we can write

$$\boldsymbol{\varphi}_2 - \boldsymbol{\varphi}_1 = (\delta + 2k\pi)\cdot\mathbf{m} + \boldsymbol{\varepsilon},\ k\in\mathbf{Z} \tag{14}$$

The difference from (7) is that the arbitrary factor is now $k\in\mathbf{Z}$ instead of $\mathbf{k}\in\mathbf{Z}^M$ in the original equation. This disambiguation comes from an implicit phase unwrapping during the minimization of $\boldsymbol{\varepsilon}$ with harmonicity constraint. This avoids possible loss of harmonicity during explicit phase unwrapping of each individual partial, e.g. in conventional sinusoidal synthesis [3].

4. APPLICATIONS IN PROCESSING HARMONIC SOUND SOURCES

The main advantage of our handling of the phase difference is that we may now attribute all pitch-related information to the harmonic progression part and focus on the residue part for handling non-harmonic aspects. By treating these two parts separately we can avoid the negative influence brought by one part to algorithms designed to handle the other.

In this section we demonstrate, in two unrelated applications, how we make use of the decomposition to handle harmonic sound sources. In a pitch marking example, we use the harmonic part to clock specific time instants within a period, and the residue part to inform on the degree of harmonicity. In another time-sale modification example, we demonstrate how we eliminate phase dispersion artefacts by avoiding frequency pollution from the misalignment part, while perfectly preserving wave shape evolution embedded in the misalignment.

4.1. Pitch marking

Pitch-synchronized processing, such as pitch-synchronous overlap-add (PSOLA)[7], operates on pitched sounds on the period level, offering quick response to period-to-period changes that is common in non-stationary quasi-harmonic signals like human speech. These algorithms rely on pre-determined time instants, known as *pitch marks*, as landmarks to synchronize their operations to. The basic requirement for pitch marking is that the interval between two adjacent pitch marks be precisely one speech period. Exactly where within a period the pitch mark should be placed has been a matter of the designer's choice: positions like prominent waveform peaks [8], significant excitation instants [9] and glottal closure instants [10] have all been reported for pitch marks. The chief concepts behind these choices are one: to clock the pitch marks consistently across periods and, if possible, across different sounds. The phase angle, by nature, handles exactly the clocking of an instant within the duration of a period. This motivates the new pitch marking algorithm presented below.

The standard procedure of pitch marking includes an initialization step, which places the first pitch mark, and a propagation step, which iteratively "grows" the existing set of pitch marks forward and backward to cover the whole length of a pitched sound. We will present our phase-based solution to the two steps in reverse order.

First let us define the phase vectors used in our algorithm. Let $s(t)$ be a periodical signal with period T_1. Given any time t_1, we consider the two-period interval $(t_1-T_1,\ t_1+T_1)$. A pitch-synchronized spectrum can be computed from the interval with

$$\hat{s}(k;t_1) = \sum_{t=t_1-T_1}^{t_1+T_1} w(t/T_1)s(t)e^{-j2\pi kt/2T_1}\ ,\ k = 0,\ 1,\ \ldots,\ T_1 \qquad (15)$$

where k is the harmonics index and $w(t)$ is an analysis window supported on [-1, 1]. The phase vector of $s(t)$ of size M ($M<T_1$) at t_1 is then taken as

$$\boldsymbol{\varphi}_1 = \left(\arg\hat{s}(1;t_1), \arg\hat{s}(2;t_1), \cdots, \arg\hat{s}(M;t_1)\right)^{\mathrm{T}} \qquad (16)$$

Now we let t_1 be where we have placed a pitch mark, and consider where to place the next. Ideally at point t_1+T_1 we should be able to sample the same phase vector $\boldsymbol{\varphi}_1$, which is rationale enough to place the next pitch mark there. However, in real tasks the signal is hardly exactly periodic, the estimate of T_1 is rarely perfect, and the period itself may have changed from T_1. Consequently at t_1+T_1 we only get some $\boldsymbol{\varphi}_2 \neq \boldsymbol{\varphi}_1$ which is not enough to signal the next pitch mark.

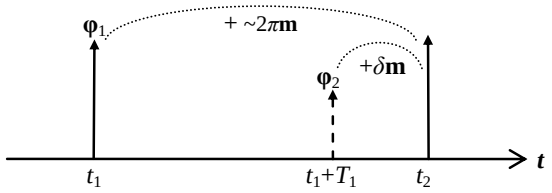


Figure 2 Pitch mark propagation by phase alignment

However, we can still place the next pitch mark *near* t_1+T_1 by harmonically progress $\boldsymbol{\varphi}_2$ by some $\delta\cdot\mathbf{m}$ to optimally match $\boldsymbol{\varphi}_1$. In other words, we put $\boldsymbol{\varphi}_1-\boldsymbol{\varphi}_2$ on the left side of (6) and solve for δ using the proposed routine (Appendix A). The new pitch mark is then placed at the adjusted position

$$t_2 \leftarrow t_1 + T_2,\quad T_2 = \frac{T_1}{1-\delta/2\pi}\ . \qquad (17)$$

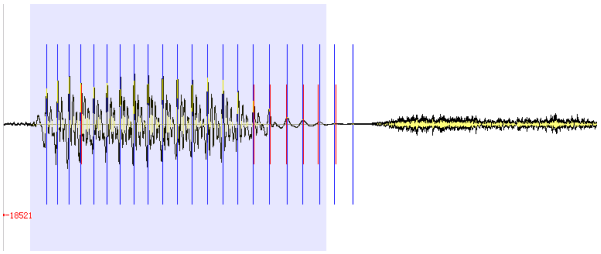
In (17) T_1 is multiplied by $2\pi/(2\pi-\delta)$, the ratio between the fundamental phase progression expected over a period and that observed over the duration of T_1. If the adjustment is large (i.e. $|\delta|$ is above some threshold), it makes sense to repeat the above adjustment process until δ is contained. Once the position of t_2 is determined, pitch marking may proceed from t_2 onwards with the new period T_2, until some termination criterion is met, such as the phase mismatch $\boldsymbol{\varepsilon}$ between pitch marks getting too large, or the correlation between marked periods getting too small. Backward propagation of pitch marks can be handled in exactly the same way.

We observe that the initial pitch mark provides a phase vector to which all other pitch marks are aligned, so its choice more or less determines what most pitch marks will be like. To better prepare for later stages that will use pitch marks, we would like all initial pitch marks have a consistent look. In [8], [9] and [10], this took the shape of waveform peak, excitation peak or glottal stop. For the phase-based pitch marking, we propose to initialize the first pitch mark at a position that is optimally aligned to the zero phase vector $\mathbf{0}^M$. This keeps the initial pitch mark close to minimum phase, which is related to high energy concentration and smooth spectral envelope preservation in PSOLA.

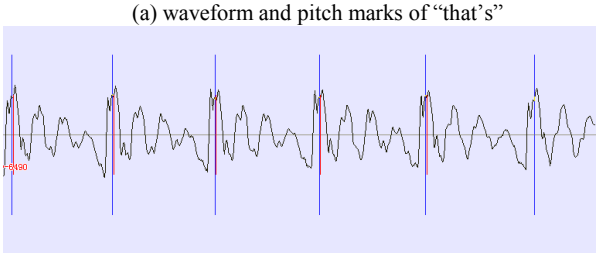
Let t_1 be a point around which the signal $s(t)$ has its period T_1 estimated with high periodicity, e.g. via autocorrelation. Let $\boldsymbol{\varphi}_1$ be the phase vector at t_1 given by (15)(16). We find out the harmonic phase progression $\delta\cdot\mathbf{m}$ for $\boldsymbol{\varphi}_1+\delta\cdot\mathbf{m}$ to optimally match $\mathbf{0}$. This is achieved by setting $\boldsymbol{\varphi}_2=\mathbf{0}$ on the left side of (6) and solve for δ with the proposed routine (Appendix A). We then update t_1 with

$$t_1 \leftarrow t_1 + T_1\cdot\delta/2\pi\ . \qquad (18)$$

This update may be repeated a few times if it brings the match closer to zero phase, which can be observed from the value of $\|\boldsymbol{\varepsilon}\|^2$ before and after each update.



(a) waveform and pitch marks of "that's"



(b) detail of (a)

Figure 3 Pitch marking example 1

Figure 3(a) shows the pitch marking result for part of a spoken sentence in which a female speaker says "that's …". Pitch marks are plotted onto the waveform as vertical lines. As we may expect, the pitch marks span the duration of the voiced (i.e. periodic) part of the speech, leaving the fricative /s/ clear. Each period of

the signal contains exactly one pitch mark. An inspection into the details (Figure 3(b)) reveals that the pitch marks are placed at the same position within its period. Each pitch mark sits between two secondary waveform peaks riding on top of the highest of the four primary peaks of each period. This agrees with the common understanding of minimum phase, as well as that of the ideal position for the waveform grain centres in PSOLA.

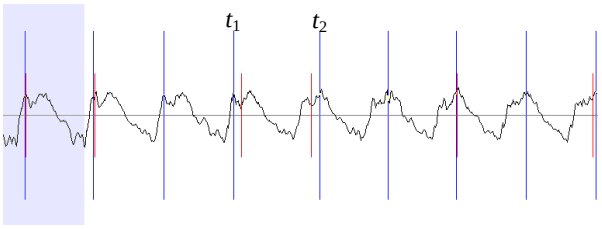


Figure 4 Pitch marking example 2

Figure 4 shows another pitch marking result during timbre evolution, in this case a change of phoneme without resting the vocal folds. We see that as the change progresses, the pitch mark shifts from one competing secondary peak (front peak at t_1) to another (rear peak at t_2). This is a common phenomenon with all landmark-based pitch markers: if each period contains two qualified landmarks, one slowly diminishes across the periods, and the other slowly rises, then a switch of the pitch mark between the two is sure to take place at least once during the transition. The advantage of our minimum phase pitch marking is that it also computes the fundamental phase progression from t_1 to t_2, so we are informed that t_2–t_1 exceeds a period's length, and by how much it does so. This information gains us better ground for adapting later processing stages, such as PSOLA, to handle such shifts properly.

4.2. Time-scale modification with sinusoidal model synthesis

Sinusoidal modelling [3] and its variants [2][11][12] represent sinusoids by sampling their amplitude, frequency and phase angle at predefined *measurement points* $\mathbf{t}=(t_0, \ldots, t_L)^\mathsf{T}$. In this part we consider only harmonic sinusoids. Let the angular frequency and phase angle of the m^{th} partial sampled at t_l be ω_l^m and φ_l^m, respectively, and let $\boldsymbol{\varphi}^m=(\varphi_0^m,\cdots,\varphi_L^m)^\mathsf{T}$, $\boldsymbol{\varphi}_l=(\varphi_l^1,\cdots,\varphi_l^M)^\mathsf{T}$, $\boldsymbol{\omega}^m=(\omega_0^m,\cdots,\omega_L^m)^\mathsf{T}$, $\boldsymbol{\omega}_l=(\omega_l^1,\cdots,\omega_l^M)^\mathsf{T}$. For each partial m, the sinusoidal model synthesis (SMS) reconstructs a new pair of frequency and phase functions $\tilde{\omega}^m(t)$ and $\tilde{\varphi}^m(t)$ by jointly interpolating $\boldsymbol{\varphi}^m$ and $\boldsymbol{\omega}^m$ with phase unwrapping, so that $\tilde{\omega}^m(t)$ is continuous and

$$\int_{t_l}^{t_{l+1}} \tilde{\omega}^m(t)dt \equiv \varphi_{l+1}^m - \varphi_l^m \pmod{2\pi},\ \forall l. \tag{19}$$

In the original SMS [3] $\tilde{\omega}^m(t)$ was constructed as piecewise quadratic. Finally the synthesizer reconstructs the phase with

$$\tilde{\varphi}^m(t) = \varphi_0^m + \int_{t_0}^{t} \tilde{\omega}^m(\tau)d\tau\ . \tag{20}$$

We consider the interval $[t_l, t_{l+1}]$, on which $\tilde{\omega}^m(t)$ depends on ω_l^m, ω_{l+1}^m, φ_l^m and φ_{l+1}^m. Let $\tilde{\boldsymbol{\omega}}_l(t)=\left(\tilde{\omega}^1(t),\cdots,\tilde{\omega}^M(t)\right)^\mathsf{T}$. We write $\tilde{\boldsymbol{\omega}}_l(t)=\tilde{\boldsymbol{\omega}}_l(t;\boldsymbol{\Omega},\boldsymbol{\varphi}_l,\boldsymbol{\varphi}_{l+1})$ to emphasize the dependency of $\tilde{\boldsymbol{\omega}}_l(t)$ on $\boldsymbol{\varphi}_l$ and $\boldsymbol{\varphi}_{l+1}$, with $\boldsymbol{\Omega}$ representing the frequencies. For all phase-aligned synthesizers, the dependency of $\tilde{\boldsymbol{\omega}}_l(t)$ on $\boldsymbol{\varphi}_l$ and $\boldsymbol{\varphi}_{l+1}$ is via their difference, so we can rewrite (19) in the following vector form:

$$\int_{t_l}^{t_{l+1}} \tilde{\boldsymbol{\omega}}_l(t;\boldsymbol{\Omega},\boldsymbol{\varphi}_{l+1}-\boldsymbol{\varphi}_l)dt \equiv \boldsymbol{\varphi}_{l+1}-\boldsymbol{\varphi}_l \pmod{2\pi}\ . \tag{21}$$

Ideally, for harmonic sinusoids the phase angles are always aligned, so that the right side of (21) is aligned to $\mathbf{0}$. In practice, due to inaccuracies of sampled frequencies and phases, $\boldsymbol{\varphi}(t_{l+1})$ is rarely perfectly aligned to $\boldsymbol{\varphi}(t_l)$, so that $\tilde{\boldsymbol{\omega}}(t)$ cannot be perfectly harmonic on $[t_l, t_{l+1}]$. This deviation from harmonicity is tolerable most of the time, as the misalignment of phase does not go much further beyond the magnitude of estimation error, so that the partials remain safe from destructive waveform interferences. However, this may not be the case when time-scale modification is involved.

The method

Sinusoidal modeling represents the time scale by the measurement points $\mathbf{t}$. Time-scale modification in this case involves selecting a new sequence $\mathbf{t}'$ and synthesizing a sound whose qualities (other than speed) at every t_l' are similar to those of the unmodified sound at t_l. In this paper we only consider the simplest type of time-scale modification, i.e. constant-rate time scaling, with

$$\mathbf{t}' = \rho\cdot\mathbf{t}\ . \tag{22}$$

where ρ is the scaling rate. The time scaling preserves the frequency value of $\tilde{\omega}^m(t)$ at point ρt, which we write as

$$\tilde{\boldsymbol{\omega}}_l'(\rho t) = \tilde{\boldsymbol{\omega}}_l(t;\boldsymbol{\Omega},,\boldsymbol{\varphi}_{l+1}-\boldsymbol{\varphi}_l)\ , \tag{23}$$

where $\tilde{\boldsymbol{\omega}}_l'(t)$ is the modified frequency function on $[t_l',t_{l+1}']$. Since the modified frequencies are linear stretching of the originals, the phase misalignment between $\boldsymbol{\varphi}_l$ and $\boldsymbol{\varphi}_{l+1}$ is multiplied by ρ. This change of misalignment can propagate across frames. As ρ becomes large the accumulated misalignment may incur destructive interference, which leads to deformed wave shape and eventually audible artefacts. This is known as *phase dispersion* and was well studied in [13].

From (23) we know that the phase progression of $\tilde{\boldsymbol{\omega}}_l'(t)$ between t_l' and t_{l+1}' is ρ times that of $\tilde{\boldsymbol{\omega}}(t)$ between t_l and t_{l+1}, including both the harmonic progression and the misalignment. However, for a harmonic sound source only the harmonic progression represents the true frequencies, which are what we aim to stretch by time scaling. The misalignment, on the other hand, represents estimation error and timbre evolution, neither a part of the true frequency, and should be left outside the integration of stretched frequency. As long as the misalignment is *not* multiplied by ρ, but remains unchanged between measurement points, there will be no extra misalignment to propagate across frame, therefore phase dispersion will not occur.

Back to synthesizer design, we break the phase difference $\boldsymbol{\varphi}_{l+1}-\boldsymbol{\varphi}_l$ into the sum of a harmonic progression and a least square residue:

$$\boldsymbol{\varphi}_{l+1}-\boldsymbol{\varphi}_l - 2\mathbf{k}\pi = \delta\cdot\mathbf{m}+\boldsymbol{\varepsilon}\ ,\ \mathbf{k}\in Z^M. \tag{24}$$

Then we construct $\tilde{\boldsymbol{\omega}}_l(t)$ in two parts:

$$\begin{gathered}\tilde{\boldsymbol{\omega}}_l(t;\boldsymbol{\Omega},\boldsymbol{\varphi}_{l+1}-\boldsymbol{\varphi}_l) = \tilde{\boldsymbol{\omega}}_l^{//}(t)+\tilde{\boldsymbol{\omega}}_l^{\perp}(t),\\ \tilde{\boldsymbol{\omega}}_l^{//}(t)=\tilde{\boldsymbol{\omega}}_l(t;\boldsymbol{\Omega},\delta\cdot\mathbf{m}),\quad \tilde{\boldsymbol{\omega}}_l^{\perp}(t)=\tilde{\boldsymbol{\omega}}_l(t;\mathbf{0},\boldsymbol{\varepsilon})\end{gathered}\ . \tag{25}$$

$\tilde{\boldsymbol{\omega}}_l^{//}(t)$ represents the harmonic frequencies which should be stretched; $\tilde{\boldsymbol{\omega}}_l^{\perp}(t)$ represents the residue frequency whose *contribution to phase* should be stretched. These two different types of stretching are implemented in a simple combined form:

$$\tilde{\boldsymbol{\omega}}_l'(\rho t) = \tilde{\boldsymbol{\omega}}_l^{//}(t) + \rho^{-1}\tilde{\boldsymbol{\omega}}_{//}^{\perp}(t), \tag{26}$$

which replaces (23) in time scaling. An intuitive interpretation of (26) is that we multiply the residue $\tilde{\boldsymbol{\omega}}_l^{\perp}(t)$ by ρ^{-1} to counteract the ρ times stretching, so that its integration between measurement points remain unchanged. A similar solution exists for handling phase in constant rate pitch scaling, which faces the same phase dispersion issue.

Example 1: artificially synthesized sound

We illustrate the time scaling algorithm above using synthesized harmonic sinusoids with M=3, L=3. We place the measurement points at 0, T, 2T and 3T, T=128.The fundamental frequency is constant at 0.005. The frequency and phase values at the measurement points are given to the synthesizers with simulated errors: (0.15,0.15,-0.15,-0.15)/T for the fundamental frequency and (0.1,0.1,-0.1,-0.1)π for the fundamental phase. For the 2nd and 3rd partials the errors are rotated by 1 and 2 slots, respectively. We use scaling rates ρ=1, 2 and 4, and compare the results with a baseline synthesizer that runs the same workflow but without the proposed phase handling, and Ninness and Henriksen's method [13], which implemented "phase invariant" time scaling to address the phase dispersion issue.

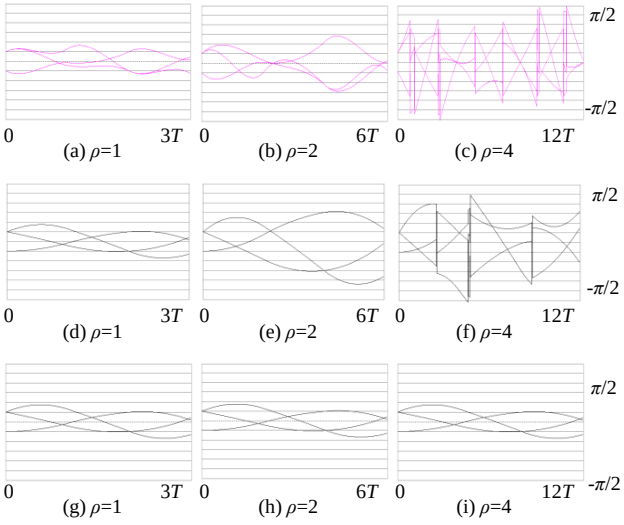


Figure 5 Phase misalignment during time scaling
(a)(b)(c)Ninness-Henriksen method; (d)(e)(f)baseline method; (g)(h)(i)baseline+proposed.

Figure 5 shows the harmonic misalignment in each setting, computed between the error-free initial phases and the synthesized phases. Each curve in Figure 5 shows the misalignment of one sinusoidal partial against time (x-axis). Results from the Ninness-Henriksen method are given in the first row, those from the baseline in the second, those from the baseline with proposed phase handling in the third. We see that the method of [13] does offer smaller misalignment than the baseline at ρ=2, but both degrades to similar level at ρ=4. The performance of the proposed method, on the other hand, is not affected by time scaling.

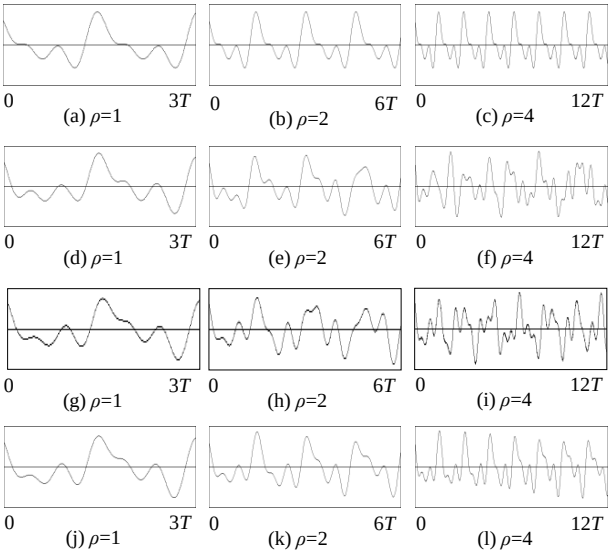
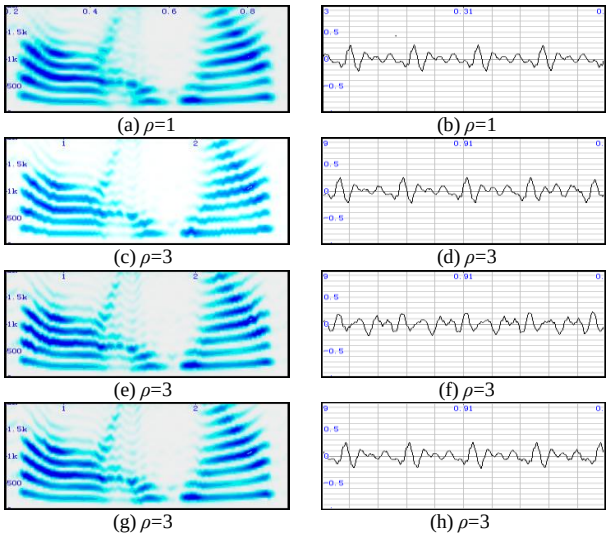


Figure 6 Waveforms before and after scaling
(a)(b)(c)natural extension; (d)(e)(f)Ninness-Henriksen method; (g)(h)(i)baseline method; (j)(k)(l)baseline+proposed.

Figure 6 shows the synthesized waveforms. The partial amplitudes are assigned the ratio $1:2^{-1/2}:3^{-1/3}$, and the initial phases of the three partials are 0, π/7 and 4π/7. The first row gives the time scaling result obtained by natural extension of the ground truth signal; the second to fourth rows give the results from the phase invariant method of [13], the baseline synthesizer, and the baseline with proposed phase handling, respectively, using accurate amplitude values and inaccurate frequency and phase values at the measurement points. We see that Ninness and Henriksen's phase invariant method has succeeded to preserve better wave shape than our baseline at ρ=2, but at ρ=4 both lose hold of the waveform. The proposed method, on the other hand, preserves the waveform equally well for ρ=1, 2 and 4.

Example 2: voiced speech

In this example we time-stretch a recording of a female voice saying "Offal is now thought to be very nutritious." Using simple harmonic sinusoidal modelling without the residue, we extract the voiced part whose spectrogram and waveform are shown in Figure 7(a) and Figure 7(b). To visualize the details Figure 7(a) only contains the "-fal is now" part and Figure 7(b) only a few periods. Three time stretchers are applied to the sentence with ρ=3, including a plain phase-aligned synthesizer, the time stretcher proposed by Ninness and Henrikson [13], and our proposed approach. Their results are given in Figure 7(c) ~ Figure 7(h), aligned to their corresponding parts in the original in Figure 7(a) and Figure 7(b). The plain synthesizer combats phase dispersion by enforcing phase alignment at measurement points, at the cost of frequency instability. Ninness and Henrikson's approach smoothes out most frequency problems of the former but, judged from by the change in wave shape, has not managed to avoid phase dispersion. Our proposed time stretcher based on harmonic phase decomposition produces no less smooth result while perfectly maintains the wave shape.



(a) ρ=1 (b) ρ=1
(c) ρ=3 (d) ρ=3
(e) ρ=3 (f) ρ=3
(g) ρ=3 (h) ρ=3

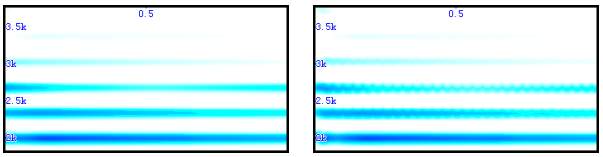
Figure 7 Spectrograms and waveforms (partial) before and after time scaling
(a)(b) original; (c)(d) plain phase-aligned synthesis; (e)(f) Ninness-Henriksen; (g)(h) proposed.

Example 3: piano

In another experiment we time-stretch a piano note extracted from a polyphonic recording using harmonic sinusoidal modelling, using the same three stretchers as above with ρ=3. As far as subjective listening and wave shape comparison are concerned, our results for the piano note are very similar to those for the spoken sentence above. However, a closer look into spectrogram details in the frequency range above the first few reveals a vibrato-like structure in some partials synthesized using harmonic phase decomposition, which is absent from those synthesized using the Ninness-Henriksen method (Figure 8). Our informal listening has not detected audible traces of this frequency modulation, probably because the modulation has not affected the strongest partials sufficiently.

We attribute the modulation to the fact that the piano sound is intrinsically inharmonic, so the harmonic phase progression does not accurately model the relationship between its partial frequencies. When the phase angles are manipulated during time scaling using (26), the computed phase progression between measurement points may deviate significantly from what would be made at the actual frequency. The synthesizer makes up for the discrepancy by bending the instantaneous frequency between adjacent measurement points, creating a frequency modulation with period ρT.

More detailed analysis (using very large DFT size) of the average partial frequencies in Figure 8(b) reveals that the frequency values maintain harmonic ratios between themselves *in groups*. For example, the frequencies of the 8th, 7th and 6th partials have the ratio 8:7:6, those of the 10th and 9th partials have 10:9, while between the 9th and 8th the ratio is larger than 9:8. In other words, the harmonic rule may predict the average frequency of some partial from the immediate predecessor with a significant gap. This gap turns out to cover whole frequency bins, which is the result of phase unwrapping during sinusoidal synthesis.



(a) ρ=3 (b) ρ=3

Figure 8 Spectrogram details of time-stretched piano note
(a) Ninness-Henriksen method; (b) proposed.

5. CONCLUSION AND FUTURE WORK

In this paper we proposed comparing two sets of harmonic phase angles by decomposing their difference into a harmonic progression and a least square residue, computed by minimizing a piecewise binomial function. The method does not mind how the phase angles are computed, nor requires the phase angle of every partial be available. One is allowed to attach weights to the phase values during the decomposition to suppress the contribution from the inaccurate phases, particularly those of weak partials.

Compared to raw phase values, this harmonically decomposed representation has several advantages: it allows comparison of phase vectors regardless of an arbitrary time shift; it allows clocking the difference between two phase vectors using information from all partials; it allows algorithms designed for processing periodic signals to focus on harmonic phase, those for non-periodic aspects on the residue. In two unrelated applications involving harmonic sound sources, we have demonstrated the use of our analysis approach to phase in two different manners, both achieving expected results. However, since phase alignment is such a common presence with harmonic and quasi-harmonic sounds, the proposed method is surely applicable in many more circumstances. For example, we have already demonstrated in Figure 5 the use of harmonic misalignment to track timbre change, which is probably more revealing than comparing waveforms, such as Figure 6.

On the other hand, we have seen that the harmonic phase decomposition comes with a few limitations. First, it requires harmonic sinusoidal analysis to provide reasonably accurate phase values of harmonically related partials, which can be a hard task in itself, especially in complex acoustical environments. Whether and how the proposed technique may be applied to more readily available forms of phase angles, such as that from the Fourier transform, remains a question to be looked into. Second, successful decomposition of the phase progression relies on good phase estimates of all participating partials, which in turn requires a mechanism to tag each phase estimate with a confidence label. In this paper we have included partial weights in section 3.2 to fill this role, but how the weights are to be best evaluated remains another question for future investigation. Moreover, the piano example shows that the proposed method does not accurately model sound sources with inharmonicity. In the case of time scaling, this has lead to additional frequency modulation of some partials, and has the potential to create audible artefacts. The adaptation of the proposed harmonic phase decomposition to sound sources with inharmonicity, therefore, may become another direction of future research into this world of aligned and misaligned phase angles.

6. ACKNOWLEDGMENTS

Part of this work was completed at Queen Mary, University of London.

7. REFERENCES

[1] D. Erro, I. Sainz, E. Navas and I. Hernaez, "Harmonics plus noise model based vocoder for statistical parametric speech synthesis," *IEEE Journal of Selected Topics in Signal Processing*, vol.8 no.2, April 2014, pp.184-194.

[2] Wen X. and M. Sandler, "Sinusoid modeling in a harmonic context," in *Proceedings of DAFx'07*, Bordeaux, 2007.

[3] R.J. McAulay and T.F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol.34, no.4, 1986, pp.744-754.

[4] R. Fischman, "The phase vocoder: theory and practice," *Organised Sound*, vol.2 no.2, July 1997, pp.127-145.

[5] H. Pobloth and W.B. Kleijn, "On phase perception in speech," in Proceedings of *ICASSP'99*, Phoenix, 1999.

[6] D.J. Breebaart, F. Nater and A.G. Kohlrausch, "Parametric binaural synthesis: Background, applications and standards,", in *NAG-DAGA 2009: International Conference on Acoustics*, Rotterdam, 2009.

[7] E. Moulines and F. Charpentier, "Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones," *Speech Communication* 9 (1990), Elsevier, North-Holland, pp.453-467.

[8] A.Chalamandaris, P. Tsiakoulis and S. Karabetsos, "An efficient and robust pitch marking algorithm on the speech waveform for TD-PSOLA," in *Proc. ICSIPA'09*, Kuala Lumpur, 2009.

[9] P.S. Murthy and B. Yegnanarayana, "Robustness of group-delay-based method for extraction of significant instants of excitation from speech signals," *IEEE Transactions on Speech and Audio Processing*, vol.7 no.6, November 1999, pp.609-619.

[10] T. Drugman and T. Dutoit, "Glottal closure and opening instant detection from speech signals," in *Proc. InterSpeech 2009*, Brighton, 2009.

[11] X. Serra and J. Smith III, "Spectral modeling synthesis: a sound analysis/synthesis based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol.14, 1990.

[12] S. N. Levine and J. Smith III, "A sines+transients+noise audio representation for data compression and time/pitch scale modifications," in *Proc. AES 105th Convention*, San Francisco, 1998.

[13] B. Ninness and S.J. Henriksen, "Time-scale modification of speech signals," *IEEE Transactions on Signal Processing*, vol.56 no.44, 2008.

APPENDIX A: COMPUTING MINIMUM HARMONIC PHASE MISALIGNMENT

Given $\boldsymbol{\varepsilon}^o=(\varepsilon_1^o, \ldots, \varepsilon_M^o)^\mathsf{T}$, $\mathbf{w}=(w_1, \ldots, w_M)^\mathsf{T}$, $w_m \geq 0$, $\forall m$, find $\delta \in [-\pi, \pi)$ that minimizes

$$D(\delta)=\sum_{m=1}^{M} w_m \cdot \mathrm{res}(\varepsilon_m^o + m\delta, 2\pi)^2 . \quad \text{(A.1)}$$

Notice that (A.1) gives δ a different sign from (13), which needs be switched back should the value of δ be required afterwards.

Because $\mathrm{res}(x,2\pi)$ is a piecewise linear regarding x, $D(\delta)$ is piecewise quadratic regarding δ. Let $\delta_0 = -\pi < \delta_1 < \ldots < \delta_L = \pi$ be the section points marking the end of these quadratic pieces, and let the left and right derivatives of $D(\delta)$ at δ_l be $D'(\delta_l)_-$ and $D'(\delta_l)_+$, then $D(\delta)$ has a local minimum in (δ_l, δ_{l+1}) if and only if $D'(\delta_l)_+ < 0$ and $D'(\delta_{l+1})_- > 0$; $D(\delta)$ has no minimum at any δ_l other than $-\pi$ and π as it has $-\infty$ derivative as such points. To find δ that minimizes $D(\delta)$, we first locate all the section points and compute the left and right derivatives, then enumerate the quadratic pieces for local minima, from which the smallest one is picked as the global minimum of $D(\delta)$.

Below is a routine for minimizing $D(\delta)$ that iteratively locates the section points contributed from each partial m and updates a section point list with derivatives.

routine 1: minimum harmonic phase misalignment

This routine maintains a list of section points $\{p_l, a_l, b_l\}$ indexed by l, in which p_l is the position of a section point and a_l, b_l are the left and right derivatives at p_l.

1 ºInitialize a sorted section point list with initial members $-\pi$ and π, both with left and right derivatives set to 0;

2 ºfor m=1, …, M, do 3 ~8 º;

3 ºcompute the first section point of partial m:

$$sp_0 \leftarrow -\pi + (\pi - \mathrm{res}(-m\pi + \varepsilon_m^o)) / m; \quad \text{(A.2)}$$

4 ºfor $sp=sp_0, sp_0 + 2\pi/m, \ldots, sp_0 + 2\pi(m-1)/m$, do 5 º;

5 ºif sp does not coincide with an existing point in the list, let the two listed section points immediately before and after sp be p_l and p_{l+1}, then we insert sp into the list with identical left and right derivatives given as

$$\frac{(p_{l+1} - sp)b_l + (sp - p_l)a_{l+1}}{p_{l+1} - p_l}; \quad \text{(A.3)}$$

6 ºfor all points p_l, l=0, 1, …, in the updated list, do 7 ~8 º;

7 ºif p_l is a section point of partial m, do

$$a_l \leftarrow a_l + m\pi \cdot w_m, \quad \text{(A.4)}$$

$$b_l \leftarrow b_l - m\pi \cdot w_m; \quad \text{(A.5)}$$

8 ºif not, do

$$a_l \leftarrow a_l + m \cdot \mathrm{res}(mp_l + \varepsilon_m^o, 2\pi) \cdot w_m, \quad \text{(A.6)}$$

$$b_l \leftarrow b_l + m \cdot \mathrm{res}(mp_l + \varepsilon_m^o, 2\pi) \cdot w_m; \quad \text{(A.7)}$$

9 ºinitialize the minimum $\delta_{\min} \leftarrow -\pi$;

10 ºfor l=1, 2…, do 11 ~13 º;

11 ºif $b_{l-1}<0$ and $a_l>0$, do

12 ºcompute local minimum

$$\delta \leftarrow \frac{p_{l-1}a_l - p_l b_{l-1}}{a_l - b_{l-1}}; \quad \text{(A.8)}$$

13 ºif $D(\delta) < D(\delta_{\min})$, $\delta_{\min} \leftarrow \delta$.

TOWARDS TRANSIENT RESTORATION IN SCORE-INFORMED AUDIO DECOMPOSITION

Christian Dittmar, Meinard Müller

International Audio Laboratories Erlangen*,
Erlangen, Germany
{christian.dittmar,meinard.mueller}@audiolabs-erlangen.de

ABSTRACT

Our goal is to improve the perceptual quality of transient signal components extracted in the context of music source separation. Many state-of-the-art techniques are based on applying a suitable decomposition to the magnitude of the Short-Time Fourier Transform (STFT) of the mixture signal. The phase information required for the reconstruction of individual component signals is usually taken from the mixture, resulting in a complex-valued, modified STFT (MSTFT). There are different methods for reconstructing a time-domain signal whose STFT approximates the target MSTFT. Due to phase inconsistencies, these reconstructed signals are likely to contain artifacts such as pre-echos preceding transient components. In this paper, we propose a simple, yet effective extension of the iterative signal reconstruction procedure by Griffin and Lim to remedy this problem. In a first experiment, under laboratory conditions, we show that our method considerably attenuates pre-echos while still showing similar convergence properties as the original approach. A second, more realistic experiment involving score-informed audio decomposition shows that the proposed method still yields improvements, although to a lesser extent, under non-idealized conditions.

1. INTRODUCTION

Music source separation aims at decomposing a polyphonic, multitimbral music recording into component signals such as singing voice, instrumental melodies, percussive instruments, or individual note events occurring in a mixture signal [1]. Besides being an important step in many music analysis and retrieval tasks, music source separation is also a fundamental prerequisite for applications such as music restoration, upmixing, and remixing. For these purposes, high fidelity in terms of perceptual quality of the separated components is desirable. The majority of existing separation techniques work on a time-frequency (TF) representation of the mixture signal, often the Short-Time Fourier Transform (STFT). The target component signals are usually reconstructed using a suitable inverse transform, which in turn can introduce audible artifacts such as musical noise, smeared transients or pre-echos, as exemplified in Figure 1(c).

In order to better preserve transient signal components, we propose in this paper a simple, yet effective extension to the signal reconstruction procedure by Griffin and Lim [2]. The original method iteratively estimates the phase information necessary for time-domain reconstruction from a magnitude STFT (STFTM) by going back and forth between the STFT and the time-domain, only updating the phase information, while keeping the STFTM fixed. Our proposed extension manipulates the intermediate time-domain reconstructions in order to attenuate the pre-echos that potentially precede the transients.

We conduct two kinds of evaluations in an audio decomposition scenario, where our objective is to extract isolated drum sounds from polyphonic drum recordings. To this end, we use a publicly available test set that is enriched with all necessary side information, such as the true "oracle" component signals and their precise transient positions. In the first experiment, under laboratory conditions, we make use of all side-information in order to focus on evaluating the benefit of our proposed method for transient preservation in signal reconstruction. Under these idealized conditions, we can show that our proposed method considerably attenuates pre-echos while still exhibiting similar convergence properties as the original method. In the second experiment, we employ a state-of-the-art decomposition technique [3, 4] with score-informed constraints [1] to estimate the component signal's STFTM from the mixture. Under these more realistic conditions, our proposed method still yields improvements yet to a lesser extent than in the idealized scenario.

The remainder of this paper is organized as follows: Section 2 provides a brief overview of related work before Section 3 introduces our new method. Section 4 details and discusses the experimental evaluation under laboratory conditions. Section 5 describes a more realistic application and evaluation of our proposed method in conjunction with score-informed audio decomposition. Finally, in Section 6 we conclude and indicate directions for future work.

2. RELATED WORK

Three research fields are important for our work: First, a number of publications on signal reconstruction and transient preservation are related and relevant for our proposed restoration method. Second, papers on score-informed audio decomposition (i.e., source separation) provide the basis for deploying our method in a real-world application.

2.1. Signal Reconstruction

The problem of signal reconstruction, also known as magnitude spectrogram inversion or phase estimation is a well researched topic. In their classic paper [2], Griffin and Lim proposed the so-called LSEE-MSTFTM algorithm (denoted as GL throughout this paper) for iterative, blind signal reconstruction from modified STFT magnitude (MSTFTM) spectrograms. In [5], Le Roux et al. developed a different view on this method by describing it using

* The International Audio Laboratories Erlangen is a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Fraunhofer-Institut für Integrierte Schaltungen IIS.

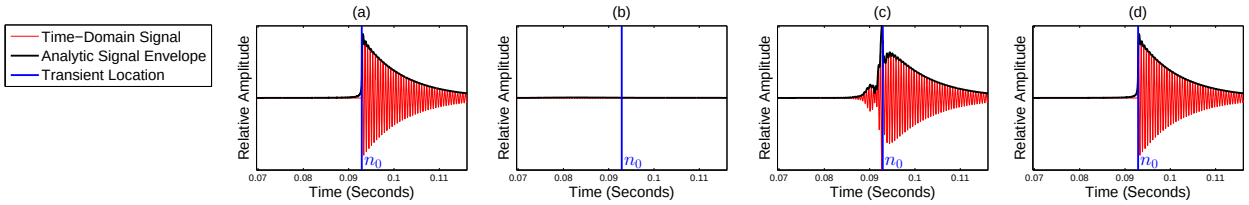


Figure 1: Illustration of the transient restoration. **(a):** Target component signal, an exponentially decaying sinusoid preceded by silence. **(b):** Reconstruction using zero phase. Due to destructive interference, the overall amplitude seemingly decreased to silence. **(c):** Reconstruction after 200 GL iterations, exhibiting pronounced transient smearing. **(d):** Reconstruction after 200 iterations of the proposed transient restoration method. The left hand legend applies to all plots, n_0 denotes the transient position.

a TF consistency criterion. By keeping the necessary operations entirely in the TF domain, several simplifications and approximations could be introduced that lower the computational load compared to the original procedure. Since the phase estimates obtained using GL can only converge to local optima, several publications were concerned with finding a good initial estimate for the phase information [6, 7]. Sturmel and Daudet [8] provided an in-depth review of signal reconstruction methods and pointed out unsolved problems. An extension of GL with respect to convergence speed was proposed in [9]. Other authors tried to formulate the phase estimation problem as a convex optimization scheme and arrived at promising results hampered by high computational complexity [10]. Another work [11] was concerned with applying the spectrogram consistency framework to signal reconstruction from wavelet-based magnitude spectrograms.

2.2. Transient Preservation

The problem of transient preservation has been extensively addressed in the field of perceptual audio coding, where pre-echo artifacts can occur ahead of transient signal components. Pre-echos are caused by the use of relatively long analysis and synthesis windows in conjunction with coding-related modification of TF bins such as quantization of spectral magnitudes according to a psycho-acoustic model. It can be considered as state-of-the-art to use block-switching to account for transient events [12]. An interesting approach was proposed in [13], where spectral coefficients are encoded by linear prediction along the frequency axis, automatically reducing pre-echos. Other authors proposed to decompose the signal into transient and residual components and use optimized coding parameters for each stream [14]. In [15], the authors proposed a scheme that unifies iterative signal reconstruction (see Section 2.2) and block-switching in the context of audio coding. Transient preservation has also been investigated in the context of time-scale modification methods based on the phase-vocoder [16]. In addition to an optimized treatment of the transient components, several authors follow the principle of phase-locking or re-initialization of phase in transient frames [17, 18].

2.3. Score-informed Audio Decomposition

The majority of music source separation techniques operate on a TF representation of the mixture signal. It is common practice to compute the mixture signal's STFT and apply suitable decomposition techniques (e.g., Non-Negative Matrix Factorization (NMF)) to the corresponding magnitude spectrogram. This yields an MSTFTM, ideally representing the isolated target signal component. The corresponding time-domain signal is usually derived by using the original phase information and applying signal reconstruction methods.

When striving for good perceptual quality of the separated target signals, many authors propose to impose score-informed constraints on the decomposition [19, 20, 1]. This has the advantage that the separation can be guided and constrained by information on the approximate location of component signals in time (onset, offset) and frequency (pitch, timbre). A few studies deal with source separation of strongly transient signals such as drums [21, 22]. Usage of the Non-Negative Matrix Factor Deconvolution (NMFD) for drum sound separation was first proposed in [3]. Later works applied it to drum sound detection using sparseness constraints [4] as well as regularisation in [23]. Others authors focus on the separation of harmonic vs. percussive components [24, 25, 26]. The importance of phase information for source separation quality is discussed in [27].

3. TRANSIENT RESTORATION

In the following, we first fix our notation and signal model and describe the employed signal reconstruction method. Afterward, we introduce our novel extension for transient preservation in the GL method and provide an illustrative example.

3.1. Notation and Signal Model

We consider the real-valued, discrete time-domain signal $x : \mathbb{Z} \to \mathbb{R}$ to be a linear mixture $x := \sum_{c=1}^{C} x_c$ of $C \in \mathbb{N}$ component signals x_c corresponding to individual instruments. As shown in Figure 2(a), each component signal contains at least one transient audio event produced by the corresponding instrument (in our case, by striking a drum). Furthermore, we assume that we have a symbolic transcription available that specifies the onset time (i.e., transient position) and instrument type for each of the audio events. From that transcription, we derive the total number of onset events S as well as the number of unique instruments C. Our aim is to extract individual component signals x_c from the mixture x as shown in Figure 2. For evaluation purposes (see Section 4), we assume to have the oracle component signals x_c available.

We decompose x in the TF-domain, to this end we employ STFT as follows. Let $\mathcal{X}(m,k)$ be a complex-valued TF coefficient at the m^{th} time frame and k^{th} spectral bin. The coefficient is computed by

$$\mathcal{X}(m,k) := \sum_{n=0}^{N-1} x(n+mH)w(n)\exp(-2\pi ikn/N), \qquad (1)$$

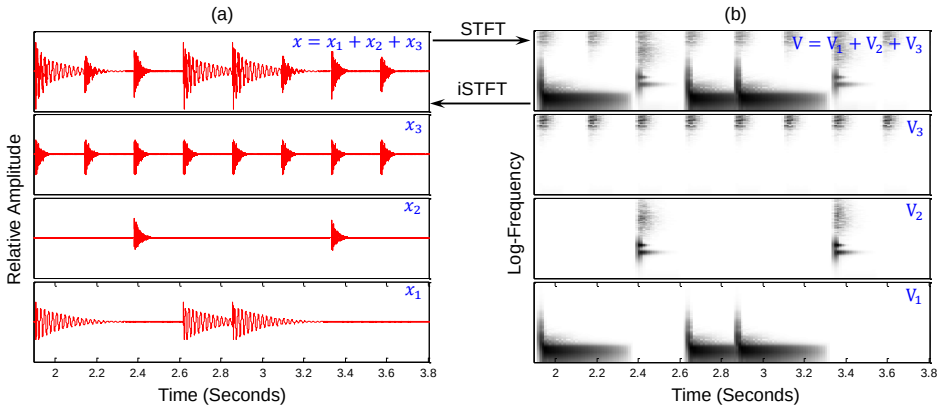


Figure 2: Illustration of our signal model. **(a):** Mixture signal x is the sum of $C = 3$ component signals x_c, each containing sequences of synthetic drum sounds sampled from a Roland TR 808 drum machine (x_1: kick drum, x_2: snare drum, x_3: hi-hat). **(b):** TF representation of the mixture's magnitude spectrogram V and $C = 3$ component magnitude spectrograms V_c. For better visibility, the frequency axis and the magnitudes are on a logarithmic scale.

where $w : [0 : N-1] \to \mathbb{R}$ is a suitable window function of blocksize $N \in \mathbb{N}$, and $H \in \mathbb{N}$ is the hop size parameter. The number of frequency bins is $K = N/2$ and the number of spectral frames $M \in [1 : M]$ is determined by the available signal samples. For simplicity, we also write $\mathcal{X} = \mathrm{STFT}(x)$. Following [5], we call $\mathcal{X}$ a consistent STFT since it is a set of complex numbers which has been obtained from the real time-domain signal x via (1). In contrast, an inconsistent STFT is a set of complex numbers that was not obtained from a real time-domain signal. From $\mathcal{X}$, the magnitude spectrogram $\mathcal{A}$ and the phase spectrogram φ are derived as

$$\mathcal{A}(m,k) := |\mathcal{X}(m,k)|\,, \tag{2}$$

$$\varphi(m,k) := \angle\mathcal{X}(m,k), \tag{3}$$

with $\varphi(m,k) \in [0, 2\pi)$.
Let $\mathrm{V} := \mathcal{A}^{\mathrm{T}} \in \mathbb{R}_{\geq 0}^{K \times M}$ be a non-negative matrix holding a transposed version of the mixture's magnitude spectrogram $\mathcal{A}$. Our objective is to decompose V into component magnitude spectrograms V_c that correspond to the distinct instruments as shown in Figure 2(b). For the moment, we assume that some oracle estimator extracts the desired $\mathcal{A}_c := \mathrm{V}_c^{\mathrm{T}}$. One possible approach to estimate the component magnitudes using a state-of-the-art decomposition technique will be described in Section 5. In order to reconstruct a specific component signal x_c, we set $\mathcal{X}_c := \mathcal{A}_c \odot \exp(i\varphi_c)$, where $\mathcal{A}_c = \mathrm{V}_c^{\mathrm{T}}$ and φ_c is an estimate of the component phase spectrogram. It is common practice to use the mixture phase information φ as an estimate for φ_c and to invert the resulting MSTFT via the LSEE-MSTFT reconstruction method from [2]. The method first applies the inverse Discrete Fourier Transform (DFT) to each spectral frame in $\mathcal{X}_c$, yielding a set of intermediate time signals y_m, with $m \in [0 : M-1]$, defined by

$$y_m(n) := \frac{1}{N} \sum_{k=0}^{N-1} \mathcal{X}_c(m,k) \exp(2\pi i k n/N), \tag{4}$$

for $n \in [0 : N-1]$ and $y_m(n) := 0$ for $n \in \mathbb{Z} \setminus [0 : N-1]$. Second, the least squares error reconstruction is achieved by

$$x_c(n) := \frac{\sum_{m\in\mathbb{Z}} y_m(n - mH) w(n - mH)}{\sum_{m\in\mathbb{Z}} w(n - mH)^2}, \tag{5}$$

$n \in \mathbb{Z}$, where the analysis window w is re-used as synthesis window. Please note that LSEE-MSTFT should not be confused with LSEE-MSTFTM (called GL in this work) that extends the signal reconstruction with iterative phase estimation (cf. Algorithm 3.2). In the following, for the sake of brevity, we will use $x_c = \mathrm{iSTFT}(\mathcal{X}_c)$ as short form for the application of (4) and (5).

3.2. Proposed Algorithm

Since we construct the MSTFT $\mathcal{X}_c$ in the TF domain, we have to consider that it may be an inconsistent STFT, i.e., there may not exist a real time-domain signal x_c fulfilling $\mathcal{X}_c = \mathrm{STFT}(x_c)$. Intuitively speaking, the complex relationship between magnitude and phase is likely corrupted as soon as the magnitude in certain TF bins is modified. In practice, this inconsistency can lead to transient smearing and pre-echos in x_c, especially for large N.
To remedy this problem, we propose to iteratively minimize the inconsistency of $\mathcal{X}_c$ by the following extension (denoted as TR) of the GL procedure [2]. For the moment, let's assume that $\mathcal{X}_c$ contains precisely one transient onset event, whose exact location in time n_0 is known. Now, we introduce the iteration index $\ell = 0, 1, 2, \ldots L \in \mathbb{N}$. Given $\mathcal{A}_c$ and some initial phase estimate $(\varphi_c)^{(0)}$, we introduce the initial STFT estimate of the target component signal $(\mathcal{X}_c)^{(0)} := \mathcal{A}_c \odot \exp(i(\varphi_c)^{(0)})$ and repeat for $\ell = 0, 1, 2, \ldots L$ the following steps

Transient Restoration (TR) Algorithm:

1. $(x_c)^{(\ell+1)} := \mathrm{iSTFT}\left((\mathcal{X}_c)^{(\ell)}\right)$ via (4) and (5)
2. Enforce $(x_c)^{(\ell+1)}(n) := 0$ for $n \in \mathbb{Z}, n < n_0$
3. $(\varphi_c)^{(\ell+1)} := \angle\mathrm{STFT}\left((x_c)^{(\ell+1)}\right)$ via (1) and (3)
4. $(\mathcal{X}_c)^{(\ell+1)} := \mathcal{A}_c \odot \exp\left(i(\varphi_c)^{(\ell+1)}\right)$

The crucial point of our proposed extension is the intermediate step

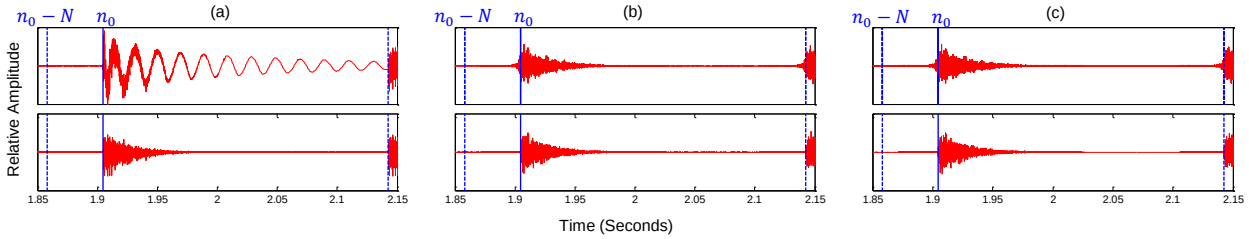


Figure 3: Different hi-hat component signals of our example drum loop. The transient position n_0 is given by the solid blue line, the excerpt boundaries by the dashed blue lines. **(a):** Mixture signal (top) vs. oracle hi-hat signal (bottom). **(b):** Hi-hat signal in Case 2, reconstruction after $L = 200$ iterations of GL (top) vs. TR (bottom). **(c):** Hi-hat signal in Case 4, reconstruction after $L = 200$ iterations of GL (top) vs. TR (bottom). Since the NMFD decomposition works very well for our example drum loop, there is almost no noticeable visual difference between (b) and (c).

2. which enforces transient constraints in the GL procedure. Figure 1 illustrates our proposed method with the target component signal in red, overlaid with the envelope of its analytic signal in Figure 1(a). The example signal exhibits transient behavior around n_0 (blue line) when the waveform transitions from silence to an exponentially decaying sinusoid. Figure 1(b) shows the time-domain reconstruction obtained from the iSTFT with $(\varphi_c)^{(0)} = 0$ (i.e., zero phase for all TF bins). Through destructive interference of overlapping frames, the transient is completely destroyed, the amplitude of the sinusoid is strongly decreased and the envelope looks nearly flat. Figure 1(c) shows the reconstruction with pronounced transient smearing after $L = 200$ GL iterations. Figure 1(d) shows that the restored transient after $L = 200$ iterations of the proposed method is much closer to the original signal. In real-world recordings, there usually exist multiple transient onsets event throughout the signal. In this case, one may apply the proposed method to signal excerpts localized between consecutive transients (resp. onsets) as shown in Figure 3.

4. EVALUATION UNDER LABORATORY CONDITIONS

For evaluation, we compared the conventional GL reconstruction with our proposed TR method under two different initialization strategies for $(\mathcal{X}_c)^{(0)}$. In the following, we describe the used data set, the test item generation, and our evaluation metrics.

4.1. Dataset

In principle, we follow the evaluation approach from [27]. In all our experiments, we use the publicly available "IDMT-SMT-Drums" dataset[1]. In the "WaveDrum02" subset, there are 60 drum loops, each given as perfectly isolated single track recordings (i.e., oracle component signals) of the three instruments kick drum, snare drum, and hi-hat. All 3×60 recordings are in uncompressed PCM WAV format with 44.1 kHz sampling rate, 16 Bit, mono. Mixing all three single tracks together, we obtain 60 mixture signals. Additionally, the onset times and thus the approximate n_0 of all onsets are available per individual instrument. Using this information, we constructed a test set of 4421 drum onset events by taking excerpts from the mixtures, each located between consecutive onsets of the target instrument. In doing so, we zero pad N samples ahead of each excerpt. The rationale is to deliberately prepend a section of silence in front of the local transient position. Inside that section, decay influence of preceding note onsets can be ruled out and potentially occurring pre-echos can be measured. In turn, this leads to a virtual shift of the local transient location to $n_0 + N$ (which we denote again as n_0 for notational convenience). In Figure 3, the adjusted excerpt boundaries are visualized by the dashed blue lines and the virtually shifted n_0 by the blue line. Since the drum loops are realistic rhythms, the excerpts exhibit varying degree of superposition with the remaining drum instruments played simultaneously. In Figure 3(a), the mixture (top) exhibits pronounced influence of the kick drum compared to the isolated hi-hat signal (bottom). For comparison, the two top plots in Figure 2(a) show a longer excerpt of the mixture x and the hi-hat component x_3 of our example signal. In the bottom plot in Figure 3(a), one can see the kick drum x_1 in isolation. It is sampled from a Roland TR 808 drum computer and resembles a decaying sinusoid.

[1] `http://www.idmt.fraunhofer.de/en/business_units/smt/drums.html`

Test case	Initial phase estimate	Fixed magnitude estimate
Case 1	$(\varphi_c)^{(0)} := \varphi^{\mathrm{Mix}}$	$\mathcal{A}_c := \mathcal{A}_c^{\mathrm{Oracle}}$
Case 2	$(\varphi_c)^{(0)} := 0$	$\mathcal{A}_c := \mathcal{A}_c^{\mathrm{Oracle}}$

Table 1: Configuration of the test cases in the experiment under laboratory conditions.

4.2. Evaluation Setting

For each mixture excerpt, we compute the STFT via (1) with $H = 512$ and $N = 2048$ and denote it as $\mathcal{X}^{\mathrm{Mix}}$. Since all test items have 44.1 kHz sampling rate, the frequency resolution is approx. 21.5 Hz and the temporal resolution is approx. 11.6 ms. We use a symmetric Hann window of size N for w. As a reference target, we take the same excerpt boundaries, apply the same zero-padding, but this time from the single track of each individual drum instrument, denoting the resulting STFT as $\mathcal{X}_c^{\mathrm{Oracle}}$. Subsequently, we define two different cases for the initialization of $(\mathcal{X}_c)^{(0)}$ as detailed in Table 1. Using these settings, we expect the inconsistency of the resulting $(\mathcal{X}_c)^{(0)}$ to be lower in case 1 compared to case 2. Knowing that there exists a consistent $\mathcal{X}_c^{\mathrm{Oracle}}$, we go through $L = 200$ iterations of both GL and our proposed TR method as described in Sec. 3.2.

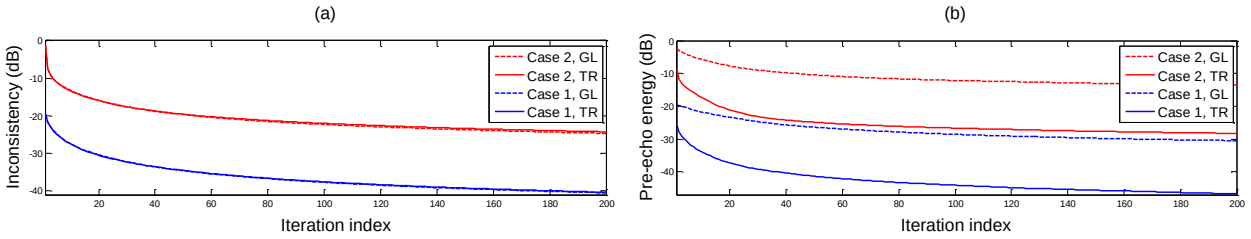


Figure 4: **(a):** Evolution of the normalized consistency measure vs. the number of iterations. **(b):** Evolution of the pre-echo energy vs. the number of iterations. The curves show the average over all test excerpts.

4.3. Quality Measures

We introduce $\mathrm{G}\left((\mathcal{X}_c)^{(\ell)}\right) := \mathrm{STFT}\left(\mathrm{iSTFT}\left((\mathcal{X}_c)^{(\ell)}\right)\right)$ to denote successive application of the iSTFT and STFT (core of the GL algorithm) on $(\mathcal{X}_c)^{(\ell)}$. Following [28], we compute at each iteration ℓ the normalized consistency measure (NCM) as

$$C\left((\mathcal{X}_c)^{(\ell)}, \mathcal{X}_c^{\mathrm{Oracle}}\right) := 10\log_{10}\frac{\left\|\mathrm{G}\left((\mathcal{X}_c)^{(\ell)}\right) - \mathcal{X}_c^{\mathrm{Oracle}}\right\|^2}{\|\mathcal{X}_c^{\mathrm{Oracle}}\|^2}, \tag{6}$$

for both test cases (see Table 1). As a more dedicated measure for the transient restoration, we compute the pre-echo energy as

$$E\left((x_c)^{(\ell)}\right) := \sum_{n=n_0-N}^{n_0}\left|(x_c)^{(\ell)}(n)\right|^2, \tag{7}$$

from the section between the excerpt start and the transient location in the intermediate, time-domain component signal reconstructions $(x_c)^{(\ell)} := \mathrm{iSTFT}\left((\mathcal{X}_c)^{(\ell)}\right)$ for both test cases (see Table 1).

4.4. Results and Discussion

Figure 4 shows the evolution of both quality measures from (6) and (7) with respect to ℓ. Diagram 4(a) indicates that, on average, the proposed TR method performs equally well as GL in terms of inconsistency reduction. In both test cases, the curves for TR (solid line) and GL (dashed line) are almost indistinguishable, which indicates that our new approach shows similar convergence properties as the original method. As expected, the blue curves (Case 1) start at much lower initial inconsistency than the red curves (Case 2), which is clearly due to the initialization with the mixture phase φ^{Mix}. Diagram 4(b) shows the benefit of TR for pre-echo reduction. In both test cases, the pre-echo energy for TR (solid lines) is around 15 dB lower and shows a steeper decrease during the first few iterations compared to GL (dashed line). Again, the more consistent initial $(\mathcal{X}_c)^{(0)}$ of Case 1 (blue lines) exhibit a considerable head start in terms of pre-echo reduction compared to Case 2 (red lines). From these results, we infer that it is sufficient to apply only a few iterations (e.g., $L < 20$) of the proposed method in cases where reasonable initial phase and magnitude estimates are available. However, we need to apply more iterations (e.g., $L < 200$) in case we have a good magnitude estimate in conjunction with a weak phase estimate and vice versa. In the following, we will assess if our preliminary findings obtained under laboratory conditions hold true in a more realistic scenario.

5. APPLICATION TO NMF-BASED AUDIO DECOMPOSITION

In this section, we describe how to apply our proposed transient restoration method in a score-informed audio decomposition scenario. As in Section 4, our objective is again the extraction of isolated drum sounds from polyphonic drum recordings with enhanced transient preservation. In contrast to the idealized laboratory conditions we used before, we now estimate the magnitude spectrograms of the component signals from the mixture. To this end, we employ NMFD [3, 4, 23] as decomposition technique. We briefly describe our strategy to enforce score-informed constraints on NMFD. Finally, we repeat the experiments described Section 4 under these more realistic conditions and discuss our observations.

5.1. Spectrogram Decomposition via NMFD

In this section, we briefly review the NMFD method that we employ for decomposing the TF-representation of x. As indicated in Section 2.3, a wide variety of alternative separation approaches exists. Previous works [3, 4, 23] successfully applied NMFD, a convolutive version of NMF, for drum sound detection and separation. Intuitively speaking, the underlying, convolutive model assumes that all audio events in one of the component signals can be explained by a prototype event that acts as an impulse response to some onset-related activation (e.g., striking a particular drum). In Figure 2(b), one can see this kind of behavior in the hi-hat component V_3. There, all instances of the 8 onset events look more or less like copies of each other that could be explained by inserting a prototype event at each onset position.

NMF can be used to compute a factorization $\mathrm{V} \approx \mathrm{W} \cdot \mathrm{H}$, where the columns of $\mathrm{W} \in \mathbb{R}_{\geq 0}^{K\times C}$ represent spectral basis functions (also called templates) and the rows of $\mathrm{H} \in \mathbb{R}_{\geq 0}^{C\times M}$ contain time-varying gains (also called activations). NMFD extends this model to the convolutive case by using two-dimensional templates so that each of the C spectral bases can be interpreted as a magnitude spectrogram snippet consisting of $T \ll M$ spectral frames. To this end, the convolutive spectrogram approximation $\mathrm{V} \approx \Lambda$ is modeled as

$$\Lambda := \sum_{\tau=0}^{T-1} \mathrm{W}_\tau \cdot \overset{\tau\rightarrow}{\mathrm{H}}, \tag{8}$$

where $\overset{\tau\rightarrow}{(\cdot)}$ denotes a frame shift operator. As before, each column in $\mathrm{W}_\tau \in \mathbb{R}_{\geq 0}^{K\times C}$ represents the spectral basis of a particular component, but this time we have T different versions of the component available. If we take lateral slices along selected columns of W_τ, we can obtain C prototype magnitude spectrograms as de-

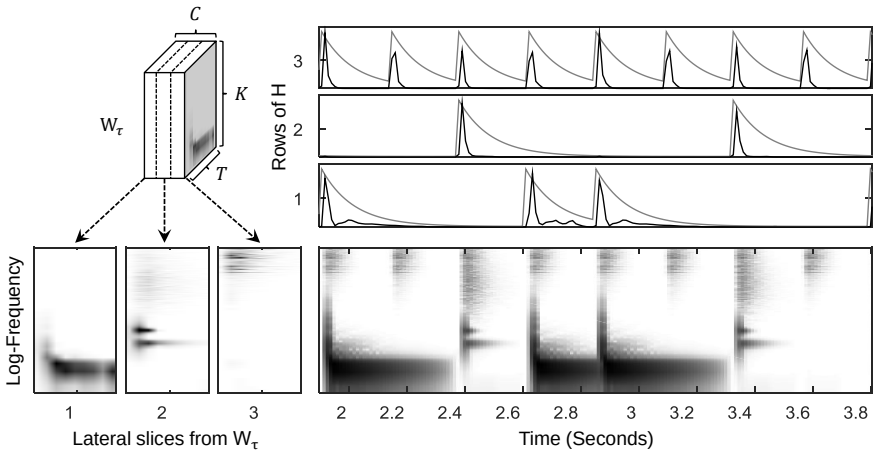


Figure 5: NMFD templates and activations computed for the example drum recording from Figure 2. The magnitude spectrogram V is shown in the lower right plot. The three leftmost plots are the spectrogram templates in W_τ that have been extracted via NMFD. Their corresponding activations in H are shown as black curves in the three top plots. The gray curves show the score-informed initialization $(\mathrm{H})^{(0)}$.

picted on the left hand side of Figure 5. NMFD typically starts with a suitable initialization of matrices $(\mathrm{W}_\tau)^{(0)}$ and $(\mathrm{H})^{(0)}$. Subsequently, these matrices are iteratively updated to minimize a suitable distance measure between the convolutive approximation Λ and V. In this work, we use the update rules detailed in [3], which we omit for brevity.

5.2. Score-Informed NMFD

Proper initialization of $(\mathrm{W}_\tau)^{(0)}$ and $(\mathrm{H})^{(0)}$ is an effective means to constrain the degrees of freedom in the NMFD iterations and enforce convergence to a desired, musically meaningful solution. One possibility is to impose score-informed constraints derived from a time-aligned, symbolic transcription [1]. To this end, the individual rows of $(\mathrm{H})^{(0)}$ are initialized as follows: Each frame corresponding to an onset of the respective drum instrument is initialized with an impulse of unit amplitude, all remaining frames with a small constant. Afterward, we apply a nonlinear exponential moving average filter to model the typical short decay of a drum event. The outcome of this initialization is shown in the top three plots of Figure 5 (gray curves).

In [19], best separation results were obtained by score-informed initialization of both the templates and the activations. For separation of pitched instruments (e.g., piano), prototypical overtone series can be constructed in $(\mathrm{W}_\tau)^{(0)}$. For drums, it is more difficult to model prototype spectral bases. Thus, it has been proposed to initialize the bases with averaged or factorized spectrograms of isolated drum sounds [21, 22, 4]. In this paper, we use a simple alternative that first computes a conventional NMF whose activations H and templates W are initialized by the score-informed $(\mathrm{H})^{(0)}$ and setting $(\mathrm{W})^{(0)} := 1$.

With these settings, the resulting factorization templates are usually a pretty decent approximation of the average spectrum of each involved drum instrument. Simply replicating these spectra for all $\tau \in [0 : T-1]$ serves as a good initialization for the template spectrograms. After some NMFD iterations, each template spectrogram typically corresponds to the prototype spectrogram of the corresponding drum instruments and each activation function corresponds to the deconvolved activation of all occurrences of that particular drum instrument throughout the recording. A typical decomposition result is shown in Figure 5, where one can see that the extracted templates (three leftmost plots) indeed resemble prototype versions of the onset events in V (lower right plot). Furthermore, the location of the impulses in the extracted H (three topmost plots) are very close to the maxima of the score-informed initialization.

In the following, we describe how to further process the NMFD results in order to extract the desired components. Let $\mathrm{H} \in \mathbb{R}_{\geq 0}^{C \times M}$ be the activation matrix learned by NMFD. Then, we define for each $c \in [1 : C]$ the matrix $\mathrm{H}_c \in \mathbb{R}_{\geq 0}^{C \times M}$ by setting all elements to zero except for the c^{th} row that contains the desired activations previously found via NMFD. We approximate the c^{th} component magnitude spectrogram by $\Lambda_c := \sum_{\tau=0}^{T-1} \mathrm{W}_\tau \cdot \overset{\tau\rightarrow}{\mathrm{H}_c}$.

Since the NMFD model yields only a low-rank approximation of V, spectral nuances may not be captured well. In order to remedy this problem, it is common practice to calculate soft masks that can be interpreted as a weighting matrix reflecting the contribution of Λ_c to the mixture V. The mask corresponding to the desired component can be computed as $\mathrm{M}_c := \Lambda_c \oslash \left(\epsilon + \sum_{c=1}^{C} \Lambda_c\right)$, where $\oslash$ denotes element-wise division and ϵ is a small positive constant to avoid division by zero. We obtain the masking-based estimate of the component magnitude spectrogram as $\mathrm{V}_c := \mathrm{V} \odot \mathrm{M}_c$, with $\odot$ denoting element-wise multiplication. This procedure is referred to as α-Wiener filtering in [29].

Test case	Initial phase estimate	Fixed magnitude estimate
Case 3	$(\varphi_c)^{(0)} := \varphi^{\mathrm{Mix}}$	$\mathcal{A}_c := \mathrm{V}_c^{\mathrm{T}}$
Case 4	$(\varphi_c)^{(0)} := 0$	$\mathcal{A}_c := \mathrm{V}_c^{\mathrm{T}}$

Table 2: Configuration of the test cases in the second experiment involving score-informed audio decomposition.

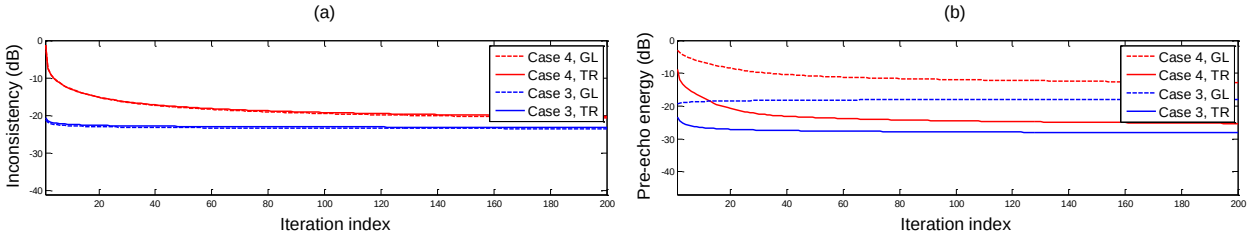


Figure 6: **(a):** Evolution of the normalized consistency measure vs. the number of iterations. **(b):** Evolution of the pre-echo energy vs. the number of iterations. The curves show the average over all test excerpts, the axis limits are the same as in Figure 4.

5.3. Evaluation Results

We now basically repeat the experiment from Section 4, keeping the STFT parameters and excerpt boundaries as described in Section 4.2. The component magnitude spectrograms are estimated from the mixture using $L = 30$ NMFD iterations and spectrogram templates with a duration of $T = 8$ frames (approx. 100 ms). Consequently, we introduce two new test cases as detailed in Table 2.

In Figure 6(a), we again observe that the inconsistency reduction obtained using TR reconstruction (solid lines) is indistinguishable from the GL method (dashed lines). The improvements are less significant compared to the numbers that can be obtained when using oracle magnitude estimates (compare Figure 4(a)). On average, the reconstructions in Case 3 (initialized with φ^{Mix}) seem to quickly get stuck in a local optimum. Presumably, this is due to imperfect NMFD decomposition of the onset related spectrogram frames, where all instruments exhibit a more or less flat magnitude distribution and thus show increased spectral overlap.

In Figure 6(b), we first see that pre-echo reduction with NMFD-based magnitude estimates $\mathcal{A}_c := \mathrm{V}_c^{\mathrm{T}}$ and zero phase (Case 4) works slightly worse than in Case 2 (compare Figure 4(b)). This supports our earlier findings, that weak initial phase estimates benefit the most from applying many iterations of the proposed method. GL reconstruction using φ^{Mix} (Case 3) slightly increases the pre-echno energy over the iterations. In contrast, applying the TR reconstruction decreases the pre-echo energy by roughly -3 dB, which amounts to approx. 15 % of the improvement achievable under idealized conditions (Case 1).

In Figure 3, different reconstructions of a selected hi-hat onset from our example drum loop is shown in detail. Regardless of the used magnitude estimate (oracle in (b) or NMFD-based in (c)), the proposed TR reconstruction (bottom) clearly exhibits reduced pre-echos in comparison to the conventional GL reconstruction (top). We provide example component signals from this drum loop and a few test items online[2]. By informal listening (preferably using headphones), one can clearly spot differences in the onset clarity that can be achieved with different combinations of MSTFT initializations and reconstruction methods. Even in cases, where imperfect magnitude decomposition leads to undesired cross-talk artifacts in the single component signals, our proposed TR method better preserves transient characteristics than the conventional GL reconstruction. Furthermore, usage of the mixture phase for MSTFT initialization seems to be a good choice since one can often notice subtle differences in the reconstruction of the drum events' decay phase in comparison to the oracle signals. However, timbre differences caused by imperfect magnitude decomposition are much more pronounced.

[2]Audio examples: `http://www.audiolabs-erlangen.de/resources/MIR/2015-DAFx-TransientRestoration/`

6. CONCLUSIONS

We proposed a simple, yet effective extension to Griffin and Lim's iterative LSEE-MSTFTM procedure (GL) for improved restoration of transient signal components in music source separation. The method requires additional side information about the location of the transients, which we assume as given in an informed source separation scenario. Two experiments with the publicly available "IDMT-SMT-Drums" data set showed that our method is beneficial for reducing pre-echos both under laboratory conditions as well as for component signals obtained using a state-of-the-art source separation technique. Future work will be directed towards automatic estimation of the required transient positions and application of this technique for polyphonic music recordings involving more than just drums and percussion.

7. ACKNOWLEDGMENTS

This work was supported by the German Research Foundation (DFG MU 268661). We would like to thank the colleagues from Fraunhofer IDMT for releasing the "IDMT-SMT-Drums" data set to the public.

8. REFERENCES

[1] Sebastian Ewert, Bryan Pardo, Meinard Müller, and Mark Plumbley, "Score-informed source separation for musical audio recordings," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 116–124, April 2014.

[2] Daniel W. Griffin and Jae S. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 32, no. 2, pp. 236–243, April 1984.

[3] Paris Smaragdis, "Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs," in *Proc. of the Intl. Conference for Independent Component Analysis and Blind Signal Separation (ICA)*, September 2004, pp. 494–499.

[4] Henry Lindsay-Smith, Skot McDonald, and Mark Sandler, "Drumkit transcription via convolutive NMF," in *Proc. of the Intl. Conference on Digital Audio Effects Conference (DAFx)*, York, UK, September 2012.

[5] Jonathan Le Roux, Nobutaka Ono, and Shigeki Sagayama, "Explicit consistency constraints for STFT spectrograms and their application to phase reconstruction," in *Proc. of the ISCA Tutorial and Research Workshop on Statistical And Perceptual Audition*, Brisbane, Australia, September 2008, pp. 23–28.

[6] Xinglei Zhu, Gerald T. Beauregard, and Lonce L. Wyse, "Real-time signal estimation from modified short-time fourier transform magnitude spectra," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 5, pp. 1645–1653, July 2007.

[7] Jonathan Le Roux, Hirokazu Kameoka, Nobutaka Ono, and Shigeki Sagayama, "Phase initialization schemes for faster spectrogram-consistency-based signal reconstruction," in *Proc. of the Acoustical Society of Japan Autumn Meeting*, September 2010, number 3-10-3.

[8] Nicolas Sturmel and Laurent Daudet, "Signal reconstruction from STFT magnitude: a state of the art," in *Proc. of the Intl. Conference on Digital Audio Effects (DAFx)*, Paris, France, September 2011, pp. 375–386.

[9] Nathanaël Perraudin, Peter Balazs, and Peter L. Søndergaard, "A fast Griffin-Lim algorithm," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, October 2013, pp. 1–4.

[10] Dennis L. Sun and Julius O. Smith III, "Estimating a signal from a magnitude spectrogram via convex optimization," in *Proc. of the Audio Engineering Society (AES) Convention*, San Francisco, USA, October 2012, Preprint 8785.

[11] Tomohiko Nakamura and Hiokazu Kameoka, "Fast signal reconstruction from magnitude spectrogram of continuous wavelet transform based on spectrogram consistency," in *Proc. of the Intl. Conference on Digital Audio Effects (DAFx)*, Erlangen, Germany, September 2014, pp. 129–135.

[12] Bernd Edler, "Codierung von Audiosignalen mit überlappender Transformation und adaptiven Fensterfunktionen," *Frequenz*, vol. 43, no. 9, pp. 252–256, September 1989.

[13] Jürgen Herre and James D. Johnston, "Enhancing the Performance of Perceptual Audio Coders by Using Temporal Noise Shaping (TNS)," in *Proc. of the Audio Engineering Society (AES) Convention*, Los Angeles, USA, November 1996, Preprint 4384.

[14] Oliver Niemeyer and Bernd Edler, "Detection and extraction of transients for audio coding," in *Proc. of the Audio Engineering Society (AES) Convention*, Paris, France, May 2006, Preprint 6811.

[15] Volker Gnann and Martin Spiertz, "Inversion of short-time fourier transform magnitude spectrograms with adaptive window lengths," in *Proc. of the IEEE Intl. Conference on Acoustics, Speech, and Signal Processing, (ICASSP)*, Taipei, Taiwan, April 2009, pp. 325–328.

[16] Jonathan Driedger, Meinard Müller, and Sebastian Ewert, "Improving time-scale modification of music signals using harmonic-percussive separation," *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 105–109, 2014.

[17] Chris Duxbury, Mike Davies, and Mark B. Sandler, "Improved time-scaling of musical audio using phase locking at transients," in *Proc. of the Audio Engineering Society (AES) Convention*, Munich, Germany, May 2002, Preprint 5530.

[18] Axel Röbel, "A new approach to transient processing in the phase vocoder," in *Proc. of the Intl. Conference on Digital Audio Effects (DAFx)*, London, UK, September 2003, pp. 344–349.

[19] Sebastian Ewert and Meinard Müller, "Using score-informed constraints for NMF-based source separation," in *Proc. of the IEEE Intl. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Kobe, Japan, March 2012, pp. 129–132.

[20] Umut Şimşekli and Ali Taylan Cemgil, "Score guided musical source separation using generalized coupled tensor factorization," in *Proc. of the European Signal Processing Conference (EUSIPCO)*, August 2012, pp. 2639–2643.

[21] Eric Battenberg, *Techniques for Machine Understanding of Live Drum Performances*, Ph.D. thesis, University of California at Berkeley, 2012.

[22] Christian Dittmar and Daniel Gärtner, "Real-time transcription and separation of drum recordings based on nmf decomposition," in *Proc. of the Intl. Conference on Digital Audio Effects (DAFx)*, Erlangen, Germany, September 2014, pp. 187–194.

[23] Axel Röbel, Jordi Pons, Marco Liuni, and Mathieu Lagrange, "On automatic drum transcription using non-negative matrix deconvolution and itakura saito divergence," in *Proc. of the IEEE Intl. Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, Australia, April 2015, pp. 414–418.

[24] Derry Fitzgerald, "Harmonic/Percussive Separation Using Median Filtering," in *Proc. of the Intl. Conference on Digital Audio Effects (DAFx)*, Graz, Austria, September 2010, pp. 246–253.

[25] Jonathan Driedger, Meinard Müller, and Sascha Disch, "Extending harmonic-percussive separation of audio signals," in *Proc. of the Intl. Conference on Music Information Retrieval (ISMIR)*, Taipei, Taiwan, October 2014, pp. 611–617.

[26] Estefanía Cano, Mark Plumbley, and Christian Dittmar, "Phase-based harmonic percussive separation," in *Proc. of the Annual Conference of the Intl. Speech Communication Association (Interspeech)*, Singapore, September 2014, pp. 1628–1632.

[27] Estefanía Cano, Jakob Abeßer, Christian Dittmar, and Gerald Schuller, "Influence of phase, magnitude and location of harmonic components in the perceived quality of extracted solo signals," in *Proc. of the Audio Engineering Society (AES) Conference on Semantic Audio*, Ilmenau, Germany, July 2011, pp. 247–252.

[28] Jonathan Le Roux, Hirokazu Kameoka, Nobutaka Ono, and Shigeki Sagayama, "Fast signal reconstruction from magnitude STFT spectrogram based on spectrogram consistency," in *Proc. of the Intl. Conference on Digital Audio Effects (DAFx)*, Graz, Austria, September 2010, pp. 397–403.

[29] Antoine Liutkus and Roland Badeau, "Generalized wiener filtering with fractional power spectrograms," in *Proc. of the IEEE Intl. Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, Australia, April 2015, pp. 266–270.

TOWARDS AN INVERTIBLE RHYTHM REPRESENTATION

AggelosGkiokas and Vassilis Katsouros

Institute for Language and Speech, Processing / R.C. Athena
Athens, Greece
{agkiokas,vsk}@ilsp.gr

Stefan Lattner and Arthur Flexer

The Austrian Research Institute for Artificial Intelligence,
Vienna, Austria
{stefan.lattner,arthur.flexer}@ofai.at

George Carayannis

National Technical University of Athens
Athens, Greece
carayan@softlab.ece.ntua.gr

ABSTRACT

This paper investigates the development of a rhythm representation of music audio signals, that (i) is able to tackle rhythm related tasks and, (ii) is invertible, i.e. is suitable to reconstruct audio from it with the corresponding rhythm content being preserved. A conventional front-end processing schema is applied to the audio signal to extract time varying characteristics (accent features) of the signal. Next, a periodicity analysis method is proposed that is capable of reconstructing the accent features. Afterwards, a network consisting of Restricted Boltzmann Machines is applied to the periodicity function to learn a latent representation. This latent representation is finally used to tackle two distinct rhythm tasks, namely dance style classification and meter estimation. The results are promising for both input signal reconstruction and rhythm classification performance. Moreover, the proposed method is extended to generate random samples from the corresponding classes.

1. INTRODUCTION

Invertible signal transformations play an essential role in the music processing field. From everyday use of music, like adjusting the equalizer of a stereo system up to the process of sound production and mixing, the transformation of audio is a crucial step for most of these applications. Most of them usually rely on the well-studied Short Time Fourier Transform (STFT) (a.k.a. Gabor Transform), which offers a very simple and intuitive way to edit audio signals. However, the main limitation of STFT is the linear spacing of frequency bins, which can be a drawback for music analysis systems, since in this case the energy concentrates on frequencies in a logarithmic scaling. The Constant Q Transform (CQT), firstly introduced by Brown [1], although it has been an alternative to STFT for almost three decades, it didn't get as much attention; not only due to its computational cost, but mostly because it is irreversible. In [2] authors presented a computational framework for a CQT with almost perfect reconstruction, and just one year later, a perfect reconstruction was achieved [3].

However, most of the signal analysis and transformation applications involve transformations based solely in time slices, i.e. observing the spectral content on a segment basis. Transformations in the other dimension (i.e. across time for each frequency bin) have not yet been studied. Such transformations would have an impact on the temporal organization of the input signal, which in the case of music is very closely related to rhythm.

The aim of this paper is to introduce a rhythm representation that can be exploited to tackle rhythm related tasks, and it is invertible, so that a rhythmically relevant signal can be reconstructed from it. Such a representation is useful, since it can provide a better insight of rhythm related features.

The rest of the paper is organized as follows. Section 2 will present current rhythm analysis systems, and will highlight the limitations of such methods towards developing an invertible rhythm representation. Section 3 will present an overview of the proposed method, while algorithmic details will be described in Section 4. An evaluation of the model's reconstruction ability and its rhythm classification performance will be presented in Section 5. Section 6 concludes this paper with a discussion and considerations for future work.

2. BACKGROUND

Most of the rhythm analysis methods involve a two step-process framework. Firstly, from a Time-Frequency representation (either STFT or CQT) of the input signal, the extraction of spectral characteristics through time (as for example frequency band energies), hereafter referred to as *accent features*, takes place. Accent features are then processed in a periodicity analysis step, such as the autocorrelation function [4-6], convolution with a bank of resonators [7-10], considering Inter-Onset-Interval histograms [11] or just by taking the Discrete Fourier Transform (DFT) of each feature [12]. The result of such an analysis is usually referred to as *spectral rhythm patterns* or *periodicity function* (PF).

Regarding the beat-tracking methods, most of them include an additional step, which combines spectral (periodicity function) with temporal (accent features) information to infer beat positions. A notable exception is the beat-tracking method presented in [5], where tempo is induced from the beat activation function.

Although the accent feature extraction step is a *lossy* transformation, it can be considered as being *lossless* in the context of

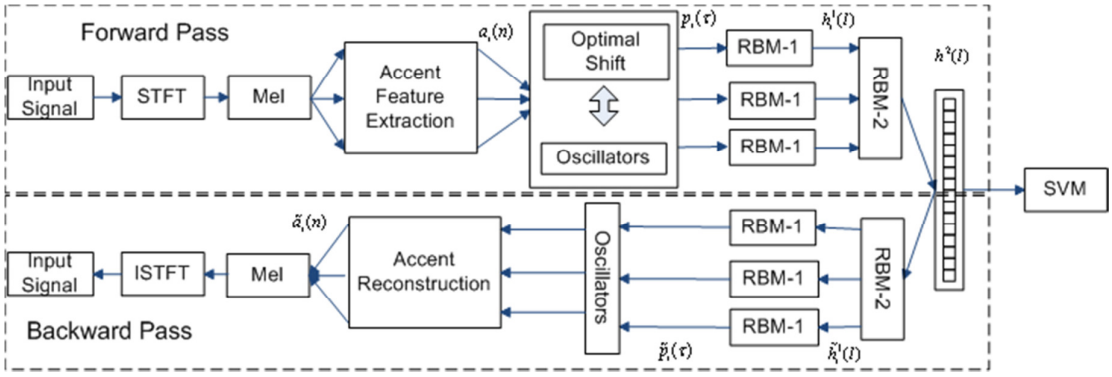


Figure 1. *Overview of the proposed method*.

rhythm analysis, as it is possible to reconstruct audio that preserves the rhythmic information of the original signal. Such a result was demonstrated in [7], where the envelopes of sub-band energies were used to modulate a noise signal. The derived signal had a rhythmic characteristic very similar to that of the original audio.

However, this is not the case for the periodicity analysis step methods. While most of them preserve spectral information of the accent features, phase or timing information gets lost, which makes it impossible to get any meaningful reconstruction of the accent features and consequently block the way to an invertible rhythm transformation.

Regarding beat-tracking methods, although beat-inference can be considered as an inverse process from the spectral to the time domain, it cannot be viewed as an inverse transformation. An exception can be found in [13], where a beat-tracking method was proposed based on the Non-Stationary Gabor Transform (NSGT) [14]. NSGT was applied to the accent features and the derived complex valued periodicity function was further processed by resampling and by peak-selection. The inverse NSGT was consequently applied to the processed periodicity function to get an estimate of the beat positions. Although this approach focuses on inferring the beats, it provides a framework for reconstructing rhythmically meaningful components of the accent features.

In this paper, we present a method for a lossy invertible rhythm transformation, which consists of two key features. Firstly, a periodicity analysis step (i.e. computing the PF) which allows for an imperfect but sufficient reconstruction of the accent features takes place. Second, we deploy a network of Restricted Boltzmann Machines (RBM) [15] on the periodicity function. RBMs are energy based, stochastic Neural Networks with a visible input layer and a hidden layer, which are able to learn the probability distribution over the training data. RBMs are generative models, able to reconstruct input data given the states of the hidden units, to sample from the learned distribution and to extract meaningful features from the data. The hidden layer provides a latent representation of the input data and can be used to tackle rhythm related tasks.

3. METHOD OVERVIEW

Figure 1 presents an overview of the proposed method. The input signal is decomposed into I frequency bands and the corresponding accent features $a_i[n]$ are extracted from each band. Next, a periodicity analysis takes place for each accent feature to extract a periodicity function $p_i[\tau]$ for each band. The periodicity analysis design has focused on preserving both amplitude and phase information of the target periodicities, such that it is possible to reconstruct the accent features from it. The periodicity functions derived from all accent features are then fed to train a single RBM (RBM-1). Actually, the RBM-1 learns a distribution over all periodicity functions of the training dataset, irrespectively from which frequency band of the signal the accent feature was computed. Afterwards, the outputs of the RBM-1, denoted by $h_i^1[l]$, are concatenated to a single vector, which is used as an input to the 2nd single-layer RBM (RMB-2). The motivation behind this architecture is that RBM-1 will learn a distribution over the individual PFs, while the RBM-2 will learn a distribution over combinations of PFs from the I frequency bands. The output of the RBM-2, denoted by $h^2[l]$, can be used as an input to a discriminative method such as a Support Vector Machine (SVM) to tackle the rhythmic task under consideration. The whole network is capable of reconstructing the accent features starting from the top-level RBM output $h^2[l]$, and then calculating sequentially $\tilde{h}_i^1[l]$, $\tilde{p}_i[\tau]$ and $\tilde{a}_i[n]$, as shown in the bottom part of Figure 1. Finally, from the reconstructed accent features $\tilde{a}_i[n]$, it is possible to derive an audio signal that preserves the most dominant rhythmic characteristics of the original audio signal.

4. METHOD DETAILS

4.1. Extracting Accent Features

The input signal is downsampled to 22.05 kHz and the STFT is computed with a sliding window of 1024 samples and half overlap between successive windows. From the amplitude spectrum $\mathbf{X}$, I band energies $e_i[n]$, $i = 1..I$ are computed with equally spaced triangular filters and half overlap in the mel-scale. Formally we can write

$$\mathbf{E} = \mathbf{X} \cdot \mathbf{M} \tag{1}$$

where $\mathbf{E} = [\mathbf{e}_1|\mathbf{e}_2|..\mathbf{e}_I]$ and $\mathbf{M}$ is the filter matrix. Then, the logarithm of the filter energies are differentiated to extract the accent feature sequence $a_i[n]$, $i = 1..I$, where n denotes the frame index. The use of logarithms followed by differentiation to extract the $a_i[n]$ is in line with [16], as it indicates relative changes w.r.t. the features' level. Each $a_i[n]$ is segmented by a sliding square window of N frames length with half hop size, resulting in approximately 12s of audio. Finally, the segments $a_i^s[n]$ are normalized w.r.t. their mean value and standard deviation.

To reconstruct an audio signal from the accent features $\tilde{a}_i^s[n]$ $\tilde{a}_i^s[n]$ the inverse pipeline is applied to $\tilde{a}_i^s[n]$, which can be summarized in the following equation:

$$\tilde{e}_i^s[n] = \exp\left(\sum_{m=1}^{n}(\tilde{a}_i^s[m]\cdot\sigma_{a_i^s} + \mu_{a_i^s})\right) \quad (2)$$

where $\mu_{a_i^s}$, $\sigma_{a_i^s}$ denote the mean and standard deviation of $\mathbf{a}_i^s$. Afterwards, the residual spectrogram $\tilde{\mathbf{X}}$ from is reconstructed from $\tilde{\mathbf{E}}$ as

$$\tilde{\mathbf{X}} = \tilde{\mathbf{E}}\mathbf{M}^{\mathbf{T}} \circ [e^{j\angle\mathbf{X}}] \quad (3)$$

where $\circ$ denotes the Hadamard product and $\angle\mathbf{X}$ denotes the phases of the initial spectrogram $\mathbf{X}$. It should be noted that if the number of bands $I \ll M/2$ where M is the FFT size, most of the harmonic content is truncated.

4.2. Periodicity Analysis

A periodicity function or a periodicity vector (PF) is an essential rhythmic representation of the accent features. Its domain is frequency of beats per minute or Hertz with typical values ranging from 0.5 Hz (30 b.p.m.) up to 5 Hz (300 b.p.m.) [17] and its value represents the salience of these periodicities. One of the most important contributions of the proposed method is to derive a periodicity function that (a) is able to reconstruct the accent features and (b) is an efficient representation which can be used to learn higher level rhythm features with Restricted Boltzmann Machines.

A typical family of periodicity analysis methods may comprise of the convolution of each accent feature sequence with a bank of resonators $\mathbf{o}_\tau$. Each $\mathbf{o}_\tau$ has an inherent oscillation frequency that corresponds to tempo τ. The maximum value of the convolution within a certain window for each accent-oscillator pair represents the salience of tempo τin this window for the accent feature $\mathbf{a}$, i.e. $p_a(\tau) = \max(\mathbf{o}_\tau * \boldsymbol{a})$, where $p_a(\tau)$ denotes the periodicity vector.

If $\mathbf{o}_\tau$ and $\mathbf{a}$ have the same length, an alternative calculation of a periodicity function $p_a(\tau)$ is given by

$$p_a(\tau) = \max_k(\mathbf{a}^{\mathbf{T}}\mathbf{o}_\tau^k) \quad (4)$$

where $\mathbf{o}_\tau^k$ denotes the circular shift of $\mathbf{o}_\tau$ by k samples. In other words, $p_a(\tau)$ corresponds to the value of the "best fit" between $\mathbf{o}_\tau$ and $\mathbf{a}$. The objective is to derive an invertible periodicity analysis step, i.e. it should be possible to reconstruct $\mathbf{a}$ *solely* by $p_a(\tau)$. For the oscillators we consider the derivative of the resonators proposed in [18], as follows

$$o_\tau(n) = d_L(n) * (1 + \tanh(\gamma\cdot(\cos(2\pi\omega_\tau n) - 1))) \quad (5)$$

where d_L denotes a non-causal differential filter of order L, ω_τ is the frequency corresponding to tempo τ and γ is called the output gain.

Let us denote $\mathbf{O^k} = [\mathbf{o}_{\tau_1}^{k_1}|\mathbf{o}_{\tau_2}^{2}|..|\mathbf{o}_{\tau_M}^{k_M}]$ where $\mathbf{k}$ is a vector containing the shifts of the oscillators of the target tempi and $\mathbf{O}^k = [\mathbf{o}_{\tau_1}^{k}|\mathbf{o}_{\tau_2}^{k}|..|\mathbf{o}_{\tau_M}^{k}]$ where k is a scalar, i.e. considering a constant shift for all $\mathbf{o}_\tau$. A naive choice to reconstruct the accent feature *solely* from $\mathbf{p}_a$ could be

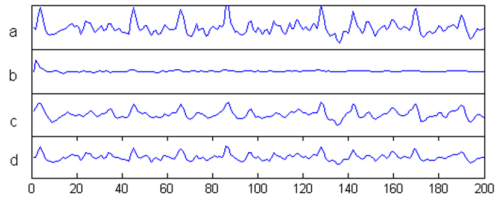


Figure 2. *Reconstruction of accent features from the periodicity vector.*

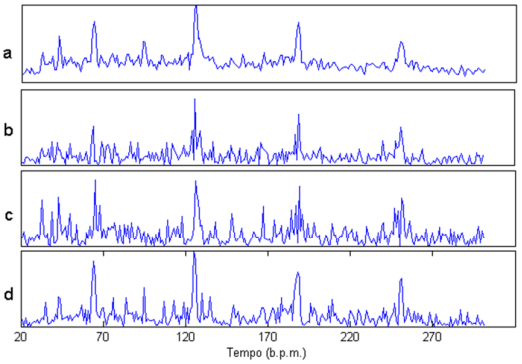


Figure 3. *The "ideal" PF (a) along with the derived PF with two random shifts k (b) and (c), and the PF computed with the proposed method (d).*

$$\tilde{\mathbf{a}} = \mathbf{O}^0\mathbf{p}_a^{\mathbf{T}} \quad (6)$$

that is to consider zero shifts for all oscillators. However, this approach has proven to result in a poor quality of accent features, because the phase, i.e. the time offsets k_τ which maximize $\mathbf{a}^{\mathbf{T}}\mathbf{o}_\tau^k$ ($k_\tau = \text{argmax}_k(\mathbf{a}^{\mathbf{T}}\mathbf{o}_\tau^k)$), are needed. To demonstrate this, Figures 2 (a) and (b) show the original and the reconstructed accent features $\mathbf{a}, \tilde{\mathbf{a}}$ respectively and it is clear that the temporal information of $\mathbf{a}$ has been lost. Figure 2(c), shows the reconstruction of the accent features, when the time offsets k_τ of each oscillator are considered in the calculation of Eq. 6.

On the other hand, if we compute the PF as $\bar{p}_a(\tau) = \mathbf{a}^{\mathbf{T}}\mathbf{o}_\tau$ instead as in Eq. 6, i.e. by setting k=0 for all oscillators, the reconstructed accent signal derived from the inverse operation (i.e. $\bar{\mathbf{a}} = \mathbf{O}\bar{\mathbf{p}}_a^{\mathbf{T}}$), is much closer to the original one, and the phase information is preserved, as shown in Figure 2 (d). The same result would hold for any constant shift k. In other words, the oscillator bank $\mathbf{O}^k$ is able to provide a rough reconstruction of $\mathbf{a}$ if all the individual shifts k_τ are constant. In other words, $\mathbf{O}^k$ can be considered as an approximate basis of the accent features. Note that although $\bar{p}_a(\tau)$ is used for the reconstruction of $\mathbf{a}$, the actual periodicity function representing the rhythm saliences is its absolute value $|\bar{p}_a(\tau)|$.

However, $\bar{p}_a(\tau)$ proves to be a poor periodicity estimator of the accent signal $\mathbf{a}$ if it is computed for an arbitrary shift k. Figure 3 (a) shows the periodicity function derived from Eq. 4 and Figure 3 (b) and (c) show $|\bar{p}_a(\tau)|$ for different time shifts k of the oscillators. It is clear that although $p_a(\tau)$ and $|\bar{p}_a(\tau)|$ exhibit peaks at the same positions, the amplitude of these peaks are very

different. Moreover, the derived PF is sensitive to time shifts of the oscillators, or equivalently to time shifts of the accent signal **a**. To sum up, we can deduce that there is a trade-off between a good periodicity function and a good reconstruction of the accent signal.

To overcome this limitation and have an efficient PF while at the same time keeping the reconstruction capabilities almost unaffected, we propose the following method. Let $a_i[n]$ denote the accent features of the $i = 1..I$ band energies. Firstly, an "ideal" periodicity function is computed as in Eq. 4, which is then averaged for all frequency bands *I*

$$\hat{p}_a(\tau) = \sum_i \left| p_{a_i}(\tau) \right| = \sum_i \left| \max_k (\mathbf{a}_i^{\mathrm{T}} \mathbf{o}_\tau^k) \right| \tag{7}$$

Next, the *actual* PF $p_a^k(\tau)$ is computed for a number of time shifts *k* as

$$p_a^k(\tau) = \sum_i \mathbf{a}_i^{\mathrm{T}} \mathbf{o}_\tau^k, \;\; \mathbf{p}_a^k = \sum_i \mathbf{a}_i^{\mathrm{T}} \mathbf{O}_\tau^k, \tag{8}$$

Note that *k* is the same for all the oscillators and for all accent bands *i*. Finally, the time shift k_0 that exhibits the higher cosine similarity between $\mathbf{p}_a^k$ and $\hat{\mathbf{p}}_a$ is chosen:

$$k_0 = \arg\max_k \left(\frac{|\mathbf{p}_a^k|^{\mathrm{T}} \hat{\mathbf{p}}_a}{\|\mathbf{p}_a^k\| \|\hat{\mathbf{p}}_a\|} \right) \tag{9}$$

The corresponding PF $p_a^{k_0}(\tau)$ can be considered as being the best approximation of $\hat{p}_a(\tau)$. The resulting periodicity function for each $\mathbf{a}_i$ is then computed by setting $k = k_0$ for all oscillators:

$$\mathbf{p}_{a_i} = \mathbf{a}_i^{\mathrm{T}} \mathbf{O}_\tau^{k_0} \tag{10}$$

Finally, the accent features are reconstructed as

$$\hat{\mathbf{a}}_i^{\mathrm{T}} = \mathbf{p}_{a_i} \left(\mathbf{O}_\tau^{k_0} \right)^{\mathrm{T}} \tag{11}$$

The choice of the same *k* on the calculation of Eq. 8 ensures a good balance between accent feature reconstruction and periodicity function approximation. Since the time shift k_0 is the same for all oscillators, the reconstruction accents will be close to $\mathbf{a}_i$ (as in Fig. 2d). In the case where only the periodicity functions $\mathbf{p}_{a_i}$ without the time shift k_0 are known, then the reconstructed accent features will differ by a time-shift value of k_0. Moreover, by choosing the same shift when computing the PF for the different bands *i*, there is no need to keep the phase information of the oscillators in order to represent the PF for all energy bands, and at the same time the derived PF is as similar as possible to the "ideal" PF. With this method, both frequency (rhythm analysis) capability and temporal (reconstruction) capability are preserved. Periodicity analysis is shift-invariant to the accent signal, while the residual signal is shifted by a constant value k_0 which is already known. Figure 3 (d) shows the PF derived with this method, which is closer to the "ideal" PF. Quantitative results of this analysis will be given in Section 5.

4.3. Learning Features with Restricted Boltzmann Machines

In a previous work [19] a Restricted Boltzmann Machine was trained on the periodicity function. The features learned by the RBM were used to successfully tackle a variety of rhythm analysis tasks. In this paper, we deploy a similar approach to exploit the periodicity function described in the previous Sections. The main difference is that instead of the absolute values of the PF (as mentioned in the previous Section), we consider the actual values of the PF. This choice relies on the reconstruction prerequisite of the proposed method. We expect that the salience of periodicities corresponding to large negative values of the PF will be preserved in the features learned by the RBM network.

The motivation of using RBMs instead of other methods such as the Auto-Encoder is firstly that RBMs are proved to derive better features, and secondly they are generative models, a property that is exploited in this paper as will be described later in Section 5.5.

If the reconstruction error of the RBM network is small enough, both periodicity and timing information are preserved. In the first step, the PF extracted for all excerpts denoted by $p_i^m[\tau]$, where *m*, *i* indicate the instance and the band respectively are all grouped into a single training dataset D_1 irrespectively of the band index *i*. D_1 is then used to train the first RBM. After RBM-1 is trained, for a target excerpt with periodicity functions $p_i^m[\tau]$, the corresponding RBM-1 outputs $\mathbf{h}_{i,m}^1$ of all bands *i*=1...*I* are concatenated to a single vector to form the training dataset D_2. If we denote the dimension of the hidden layer by N_1 the feature dimension of D_2 is $I \cdot N_1$. D_2 is subsequently used to train RBM-2, with $I \cdot N_1$ visible and N_2 hidden units. While RBM-1 is dedicated to learn the distribution of the individual accent features, RBM-2 learns an overall distribution across all *I* band energies. We denote the output of the RBM-2 as $\mathbf{h}_m^2$.

4.4. Rhythm Classification

To demonstrate the discriminative potential of the extracted features, we have deployed an SVM classifier with a Radial Basis Kernel for tackling meter estimation and dance style classification tasks. The LIBSVM [20] implementation was used. Given a dataset $\{\mathbf{a}_i^m\}_{m=1..M}$ with a predefined set of classes *C*, the corresponding RBM outputs denoted by $H = \{\mathbf{h}_m^2\}_{m=1..M}$ are computed and used as the feature space on which we employ SVM classifiers using the one-to-one multiclass approach. All datasets were split to 10 folds such that the distribution of the classes is the same for all folds and a 10-fold cross validation approach was used. 8 folds were used for training, one fold for testing and one for validation. SVM was trained on a segment basis and the classification decision was taken with a majority vote over each test excerpt.

5. EVALUATION

5.1. Evaluation Setup

The proposed method was evaluated on five datasets for two distinct tasks. The first task is related to Meter estimation, where experiments were conducted on the Essen Folk Song database [21] and Finish Folk Collections database [22] comprising of 6207and 7735 melodies in MIDI format respectively. MIDI files were synthesized to 22 kHz audio and ground-truth time signature information was extracted from the MIDI files. The second task is Dance Style Classification performed on the ballroom dataset [23], which consists of audio samples 30s long of 8 dance rhythm classes.

Apart from recognition performance, in order to demonstrate the inversion capabilities of the proposed model, the Speedo [24] and GTZAN [25] datasets which mainly consist of popular music audio, were used.

5.2. Network Training

The RBM network was trained on a subset of the Million Song Dataset (MSD) [26] consisting of 130.000 excerpts. This results in approximately 4.5 million instances for training the RBM-1 and 900K instances for training the RBM-2. The training instances for the RBM-1 were normalized to zero mean and unit standard deviation for each dimension and we used Gaussian Visible and Noisy Rectified Linear Hidden Units (NReLU) [27]. For the RBM-2, both layers consist of NReLUs. The number of accent bands was chosen $I = 5$, and the periodicity function was calculated for $\tau_{\min} = 20$, $\tau_{\max} = 300$ with a $\delta\tau = 1$step. The number of hidden units for RBM-1 and RBM-2 were chosen $N_1 = 300$ and $N_2 = 500$ respectively, resulting in an (281x300) architecture for RBM-1 and in an (1500x500) for RBM-2. Both RBMs were trained using Contrastive Divergence-1 [15].

5.3. Reconstructing Accent Signals from the Periodicity Vector

In the inverse pipeline presented in Figure 1 for reconstructing $\tilde{a}_i[n]$ from $\mathbf{h}^2$, the errors for each step are accumulated yielding the final reconstruction error. This Section will present a deeper insight into the reconstruction errors of the accent features, produced by the periodicity analysis step and the details of the method presented in Section 4.2 will be established experimentally. Moreover, PF approximation (Eq. 10) along with RBM and overall network reconstruction errors will be reported.

Let $\hat{a}_i[n]$ denote the reconstruction of accent feature $a_i[n]$ solely from the periodicity function as computed by Eq. 8, and let $\tilde{a}_i[n]$ denote the accent feature reconstructed from the whole network. The differences between $(\hat{a}_i[n], a_i[n])$, $(\tilde{a}_i[n], \hat{a}_i[n])$ and $(\tilde{a}_i[n], a_i[n])$ correspond to the reconstruction errors of the accent features introduced by the periodicity analysis step (RBMs are ignored), the RBM influence on accent signal reconstruction and the whole architecture respectively. Regarding periodicity function approximation, the difference of the "ideal" PF $\hat{p}_a(\tau)$ (Eq. 7) and $p_a(\tau) = \sum_i |p_{a_i}(\tau)|$ (Eq. 10) corresponds to the error introduced by the periodicity analysis step. If $\tilde{p}_{a_i}(\tau)$ denotes the *pf* derived from the RBM reconstruction, then the difference between $\tilde{p}_{a_i}(\tau)$ and $p_{a_i}(\tau)$ corresponds to the reconstruction error of the PF introduced by the RBM network, while the difference between $\hat{p}_a(\tau)$ and $\tilde{p}_a(\tau) = \sum_i \tilde{p}_{a_i}(\tau)$ can be viewed as a measure of the overall approximation of $\hat{p}_a(\tau)$ by the whole network.

To quantify the performance of the proposed method w.r.t. accent feature reconstruction and PF approximation we consider the cosine similarity $R_{\mathbf{x}\tilde{\mathbf{x}}} = \mathbf{x}^T\tilde{\mathbf{x}}/\|\mathbf{x}\|\|\tilde{\mathbf{x}}\|$ as an evaluation measure. The choice of the cosine measure instead of other conventional measures such as the L^2 norm relies on the fact that cosine measure is more intuitive with respect to the proposed context. For example, the cosine similarity between $a_i[n]$ and an amplified version of it $A \cdot a_i[n]$ will be 1, which is not the case for any norm. The same holds for comparing the PFs as well.

Table 1 summarizes the results for the accent feature and PF reconstruction for the periodicity analysis step, the RBM network and the whole architecture for all evaluation datasets described in previous section. Values correspond to mean values for each dataset. Regarding the approximation of the "ideal" periodicity function ($R_{\hat{p}p}$value) by finding a single optimal time shift for all oscillators as described in Eq. 7 and Eq. 10, a value around 0.9 for cosine similarity was achieved for all datasets. RBM's reconstruction of $\tilde{p}_{a_i}(\tau)$ is around 0.92 for all datasets while the similarity of $\tilde{p}_{a_i}(\tau)$ with the $\hat{p}_a(\tau)$ is above 0.87 for all datasets. In other words, the network learns a latent representation of the periodicities $\mathbf{h}^2$ from which we can reconstruct a PF that is very close to the "ideal PF". It is also noteworthy that the reconstruction rates of the PFs are almost the same for every dataset, even for those which stem from midi files.

Dataset	Periodicity		RBM Error		Overall Error	
	$R_{\hat{a}a}$	$R_{\hat{p}p}$	$R_{\tilde{a}\hat{a}}$	$R_{\tilde{p}p}$	$R_{\tilde{a}a}$	$R_{\tilde{p}\hat{p}}$
Essen	74.0	89.7	94.7	91.2	63.6	87.0
F-Folk	72.0	90.5	92.5	89.9	59.4	87.8
Genres	78.7	90.4	94.9	92.6	69.3	87.9
Speedo	79.5	90.4	95.1	92.8	70.4	87.9
Ballroom	78.6	89.8	95.2	92.9	69.9	87.6

Table 1. *Detailed reconstruction approximation results of the proposed method. Values correspond to* $100 \cdot R_{\mathbf{x}\tilde{\mathbf{x}}}$

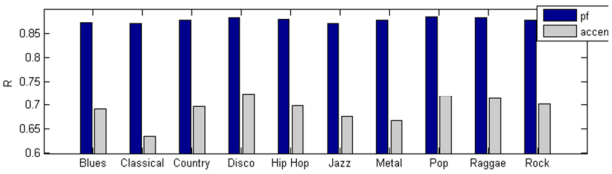


Figure 4. *Reconstruction approximation of accent features and periodicity vector across all genres on the TGZAN dataset.*

Regarding *accent feature* reconstruction rates of the proposed method, the 1st column of Table 1 indicates a larger reconstruction error of the accent features due to the inverse PF analysis step (Eq. 11). Although the accent feature reconstruction error introduced by the RBMs, i.e. the difference between accent features reconstructed from $p_{a_i}(\tau)$ and $\tilde{p}_{a_i}(\tau)$, the relatively small $R_{\hat{a}a}$ has an impact on the final reconstruction capabilities of the whole network $R_{\tilde{a}a}$ which is around 0.7 for audio and 0.6 for the midi datasets.

To get a better insight into the reconstruction errors, Figure 4 shows $R_{\tilde{a}a}$ and $R_{\tilde{p}\hat{p}}$ for each genre of the GTZAN dataset. As expected, the reconstruction error of the accent features is larger for some genres, such as classical, metal and jazz, while it is better for some genres with more steady rhythm, such as disco and pop. On the other hand, it is noteworthy that the periodicity function approximation value $R_{\tilde{p}\hat{p}}$ is almost the same for all genres. Audio examples of the reconstructions for all excerpts of the GTZAN can be found in[1]. For a direct comparison to the original tracks, the audio files reconstructed from the initial accent features (Eq. 3) are also provided.

5.4. Classifying to Rhythm Classes

5.4.1. Meter Estimation

To report comparable results with other methods [28] for the meter estimation task, 9 meter classes were considered, meters 2/4, 3/2, 3/4, 3/8, 4/1, 4/2, 4/4, 6/4, 6/8 were chosen for the Essen Collection and meters 2/4, 3/2, 3/4, 3/8, 4/4, 5/2, 5/4, 6/4, 6/8 for the Finish Folk Collection. The proposed method achieved a classification accuracy of 80.7% and 75% for the Essen and Finish Folk song collections on a track basis. For a better insight into

[1] http://mir.ilsp.gr/invertible_rhythm.html

	Predictions								
	2/4	3/2	3/4	3/8	4/1	4/2	4/4	6/4	6/8
2/4	83	0	4	1	0	0	12	0	1
3/2	0	**42**	9	0	0	31	18	0	0
3/4	7	0	**83**	0	0	0	10	1	0
3/8	36	0	12	**19**	0	0	2	0	31
4/1	0	0	0	0	**86**	14	0	0	0
4/2	0	1	0	0	2	**92**	4	0	0
4/4	5	0	3	0	0	1	**91**	0	0
6/4	0	0	61	0	0	0	13	**26**	0
6/8	5	0	6	2	0	0	1	0	**86**

Table 2. *Confusion matrix for the meter estimation task on the Essen Folk Song Collection. Values are percentages (%). Rows correspond to ground truth and columns to predictions.*

	Essen		Finish Folk	
Method	9 class	2 class	9 class	2 class
Proposed	80.7	89.2	75.0	93.8
Toivianen[28]	83.2	95.3	68	96.4
Eck [6]	-	90	-	93

Table 3. *Comparison with other reference methods for the 2-class and 9-class meter estimation.*

the classification performance, the confusion matrix of the classification results for the Essen dataset is presented in Table 2. Most of the meter classification errors are for similar meters, as for example in the case of 3/8 examples, where 43% of the cases were classified either as 6/8 or ¾.

If we consider only two broad meter categories, i.e. *duple* (e.g. 2/4, 4/4, 4/8 etc. meters) and *triple/compound* (e.g. 3/8, 6/8, 9/8, 3/2 etc. meters) the classification accuracy for the Essen dataset and the Finish Folk dataset become 89.2% and 93.8% respectively. Table 3 presents comparative results to other existing methods for the 2-class and 9-class classification problem. The proposed method achieves comparative results to state-of-the-art methods.

5.4.2. Dance Style Classification

Table 4 presents the confusion matrix of the classification results on the *Ballroom* dataset. The most correctly classified genre is the Quickstep (QS), with a classification accuracy of over 97%, while Tango (TA), ChaCha (CH) and Waltz (W) were correctly recognized with around 90% accuracy. On the other hand, many instances of Jive (JI) and Viennese Waltz (VW) were confused with Waltz. Table 5 presents the overall classification results of the proposed method (81.95%), compared to other methods. It is noteworthy that as in the case of meter estimation, without any prior knowledge about the task, the features learned from the network achieved a performance close to the state-of-the-art methods. Note that for the reference methods, the values in parenthesis correspond to results that were achieved when the ground-truth tempo was given. Consequently, they are not comparable to the proposed method.

5.5. Sampling from Rhythm Classes

To understand the features learned by the RBM network as well as the reconstruction capabilities of the proposed method,

	Predictions							
	CH	JI	QS	RU	SA	TA	VW	W
CH	**88.3**	0	2.7	3.6	0.9	4.5	0	0
JI	3.3	**60**	0	3.3	3.3	1.7	5	23.3
QS	0	0	**97.6**	2.4	0	0	0	0
RU	1	4.1	1	**77.6**	0	4.1	5.1	7.1
SA	1.2	1.2	4.7	8.1	**79.1**	1.2	1.2	3.5
TA	5.8	0	0	0	0	**89.5**	0	4.7
VW	0	1.5	0	1.5	0	0	**58.5**	38.5
W	0	2.7	0	1.8	0	0.9	4.5	**90**

Table 4. *Confusion matrix for the dance style classification task on the Ballroom dataset. Values are percentages (%). Rows correspond to ground truth and columns to predictions.*

Proposed	Gouyon[29]	Peeters[30]	Dixon [31]
81.95	79.6 (90.1)	81 (90.4)	84 (96)

Table 5. *Classification accuracies (%) on the Ballroom dataset of the proposed method along with three reference methods.*

we provide audio examples *sampled* from the overall architecture. To do so, we followed Hinton's approach in [32] to sample digit images from the ten digit classes of MNIST, as shown in Figure 5. A binary class vector $\mathbf{c}$ consisting of softmax units was concatenated to the network output vector $\mathbf{h}^2$ to form the visible vector of an additional RBM with N=3000 hidden binary units in order to learn a joint distribution of $\mathbf{c}$ and $\mathbf{h}^2$. The RBM was trained using Persistent Contrastive Divergence [33]. After training, samples for a given class where drawn by clamping $\mathbf{c}$ to a constant value corresponding to this class and running a Gibbs chain with $\mathbf{h}^2$ being randomly initialized. During Gibbs sampling, audio examples were reconstructed from $\mathbf{h}^2$.

We applied this method to draw samples from the eight *Ballroom* classes. In order to provide a quantitative measurement of the quality of the samples, we followed the following procedure. For each class, after 2000 initial Gibbs steps, one sample was drawn every 250 Gibbs steps. Afterwards the samples were filtered out, such that the value c of the class vector $\mathbf{c}$ on the negative phase of the Gibbs sampling for the corresponding class was over 0.7. With this procedure, 50 samples were finally drawn from each class. Afterwards, for each sample the 10 most similar training instances were retrieved. Similarity was computed as the cosine similarity measure of the corresponding periodicity functions. The retrieval matrix obtained with the above procedure is presented in Table 6. Rows correspond to samples drawn from the model and columns correspond to training instances retrieved from each class. The last row shows the mean value of c for each class computed over all Gibbs steps. Interestingly, for some classes, retrieval rates are very high, as for example for ChaCha and Samba. As expected, samples from Waltz and Viennese Waltz are similar and as a consequence many Waltz samples are closer to Viennese Waltz training instances and vice-versa. Rumba is confused with Samba, and similarly to Table 4, many Jive samples are close to Waltz instances. An interesting effect which should be further investigated is the case of Quickstep and Tango samples, which were closer to the Samba and the ChaCha training instances respectively. However it should be noted that this experiment corresponds only to a random snapshot; several trials indicated a high sensitivity of these results to both training epochs of the RBM and Gibbs sampling steps. Such an effect, should be investigated in the future. Nevertheless, the overall results of Table 6 indicate that in most of the cases the samples drawn from the model are closer to the actual training instances

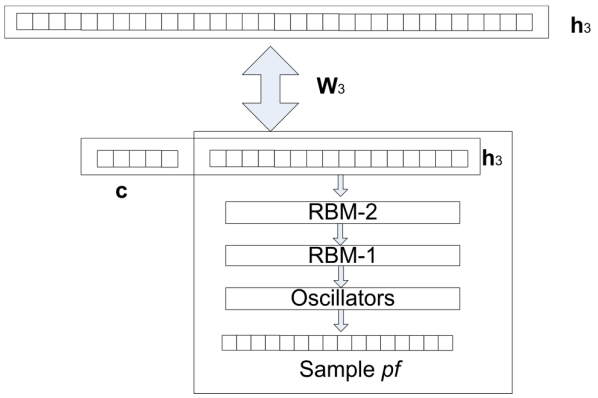


Figure 5. *Class sampling method overview.*

of the corresponding class, than to those of all other classes. Audio examples of the samples can be found in[1]

6. CONCLUSION AND FURTHER WORK

In this paper we presented a method for constructing a rhythm representation that is capable of reconstructing an audio signal that resembles the rhythmic properties of the original. Besides the reconstruction efficiency, the derived features proved to be successful for tackling two important rhythm analysis tasks, namely dance style classification and meter estimation. Without any prior knowledge of the tasks, the performance of the proposed method is comparable to state-of-the-art methods.

The accent feature reconstruction error introduced by the periodicity analysis step, should be investigated further in future work, as for example using other oscillator types. Another possible solution could be to increase the tempo analysis range. However, a balance between reducing this error and keeping the size of the network on a relative small size should be preserved.

The building block of the proposed network, the Restricted Boltzmann Machine, apart from being exploited as a generative model for reconstructing the input and as feature detector to tackle rhythm related tasks, has another important property. It provides a framework for generating samples from classes. Such an extension in the audio domain is a powerful tool since it will provide a deeper insight of what rhythmic features and classes are learnt.

Convolution plays an important role in rhythm processing, since many rhythm analysis methods involve a convolution step with a bank of template filters that correspond to certain frequencies, in order to compute a periodicity function. Instead of jointly learning rhythmic and timing information of the accent features by the periodicity analysis step, a Convolutional Restricted Boltzmann Machine acting directly on the time-domain accent features and representing rhythmic information could be explored as an alternative approach.

7. ACKNOWLEDGEMENTS

This research is funded by the Vienna Science and Technology Fund (WWTF) through project MA14-018, by the Future and Emerging Technologies (FET) Programme Lrn2Cr8 within the Seventh Framework Programme for Research of the European Commission, under FET grant number 610859 and by the Lang-TERRA project(FP7- REGPOT-2011-1)

	CH	JI	QS	RU	SA	TA	VW	W
CH	**100**	0	0	0	0	0	0	0
JI	0.8	**39.6**	0	3.3	0	0.2	11.8	44.3
QS	2	2	**40.2**	9.4	44.7	0.8	0.2	0.8
RU	0.8	2.9	4.5	**50**	38.4	0	0.6	2.7
SA	0	0	0.8	12.9	**86.3**	0	0	0
TA	33.1	5.7	0.2	1.2	0.2	**55.9**	1.6	2.2
VW	1	6.9	0	3.3	3.5	1	**55.9**	28.4
W	1.4	32.4	0.8	3.3	1.4	2.4	14.3	**44.1**
c	98.3	68.2	77.2	55.9	91.1	78.7	58.4	64.4

Table 6. *Samples to training instances retrieval precision for each class pair. Rows correspond to class samples and columns to class training instances. Values are in percentage.*

8. REFERENCES

[1] Brown, J. C., "Calculation of a constant Q spectral transform," *The Journal of the Acoustical Society of America*, *89*(1), pp. 425-434, 1991

[2] Schörkhuber, C., and Klapuri, A. ,"Constant-Q transform toolbox for music processing," in *7th Sound and Music Computing Conference,* Barcelona, Spain ,2010

[3] Velasco, G. A., Holighaus, N., Dörfler, M., & Grill, T. (2011). "Constructing an invertible constant-Q transform with non-stationary Gabor frames," in *Proceedings of DAFX11, Paris*.

[4] G. Percival and G. Tzanetakis. "Streamlined tempo estimation based on autocorrelation and cross-correlation with with pulses." *IEEE/ACMTrans. Speech Audio Process.* 22(12), pp. 1765-1776, Dec. 2014.

[5] S. Böck and M. Schedl. "Enhanced beat tracking with context-aware neural networks." In *Proc. DAFX,* 2011.

[6] D. Eck, and N. Casagrande, "Finding meter in music using an autocorrelation phase matrix and shannon entropy," in *Proc. ISMIR*, 2005, pp. 504-509.

[7] E. D. Scheirer, "Tempo and beat analysis of acoustic musical signals," *J. Acoust. Soc. Amer*., 103(1), pp. 588–601, 1998.

[8] Alonso, M. A., Richard, G., & David, B."TempoAnd Beat Estimation Of Musical Signals. In *Proc. ISMIR*, 2005

[9] A. Klapuri, A. Eronen, and J. Astola, "Analysis of the meter of acoustic musical signals*," IEEE Trans. Audio, Speech, Lang. Process*., 14(1), pp. 342–355, Jan. 2006.

[10] A. Gkiokas, V. Katsouros, G. Carayannis & T. Stafylakis, "Music tempo estimation and beat tracking by applying source separation and metrical relations," *in Proc. ICASSP*, pp. 421-424, 2012

[11] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *J. New Music Res*., 30(1), pp. 39–58, 2001.

[12] G. Peeters and J. Flocon-Cholet, "Perceptual Tempo Estimation using GMM-Regression," in *Proc. MIRIUM*, Nara, Japan, 2012

[13] A. Holzapfel, G.A. Velasco, N. Holighaus, M. Dörfler & A. Flexer, "Advantages of nonstationary gabor transforms in beat tacking," in *Proc. of the 1st international ACM workshop on Music information retrieval with user-centered and multimodal strategies*, pp. 45-50, 2011.

[14] Holighaus N., Dörfler M., Velasco G. A. and Grill T.: "A framework for invertible, real-time constant-Q transforms," *IEEE Transactions on Audio, Speech and Language Processing*, 21(4), pp. 775-785, 2013.

[15] G. Hinton, "Training products of experts by minimizing contrastive divergence." *Neural computation,* 14(8), pp. 1771-1800, 2002.

[16] A. Klapuri, "Sound onset detection by applying psychoacoustic knowledge," in *Proc. ICASSP*, 1999, pp. 3089-3092.

[17] L. van Noordenan D. Moelants, "Resonance in the perception of musical pulse," *Journal of New Music Research*, 28(1), 43-66, 1999

[18] Large E. and Kolen J., "Resonance and the Perception of Musical Meter", *Connection Science 6(1), pp 177-208,* 1994

[19] A. Gkiokas, V. Katsouros and G. Carayannis, "Towards Universal Spectral Rhythm Features: An Application to Dance Style, Meter and Tempo Estimation," Submitted to *IEEE/ACMTrans. Speech Audio Process.,* Feb. 2015, under review.

[20] C.C. Chang, and C.J. Lin. "LIBSVM: a library for support vector machines." *ACM Trans. on Intel. Systems and Tech.*, 2(3), pp. 27:1-27, 2011.

[21] H. Schaffrath and D. Huron. "The Essen folksong collection in the humdrum kern format," Menlo Park, CA: Center for Computer Assisted Research in the Humanities, 1995.

[22] T. Eerola and P. Toiviainen. "Digital archive of Finnish folk tunes." Computer Database, University of Jyvaskyla, 2004.

[23] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Trans. Audio, Speech, Lang. Process.*, 14(5), pp. 1832–1844, Sep. 2006

[24] M. Levy, "Improving perceptual tempo estimation with crowd-sourced annotations," in *Proc. ISMIR*, 2011, pp. 317–322.

[25] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," I*EEE Trans. Speech Audio Process.*, 10(5), pp. 293–302, Jul. 2002.

[26] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman and P. Lamere. "The million song dataset," in *Proc. ISMIR*, 2011, pp. 591-596.

[27] V. Nair and G. Hinton. "Rectified linear units improve restricted boltzmann machines," in *Proc. ICML*, 2010, pp. 807-814.

[28] P. Toiviainen and T. Eerola. "Autocorrelation in meter induction: The role of accent structure," *J. Acoust. Soc. Amer.*, 119(2), pp. 1164-1170, 2006.

[29] F. Gouyon, S. Dixon, E. Pampalk and G. Widmer. "Evaluating rhythmic descriptors for musical genre classification," in *Proc. AES* 2004, pp. 196-204.

[30] G. Peeters, "Rhythm Classification Using Spectral Rhythm Patterns," in *Proc.ISMIR*, 2005 pp. 644-647.

[31] S. Dixon, F. Gouyon and G. Widmer, "Towards Characterisation of Music via Rhythmic Patterns," in *Proc.ISMIR*, 2004.

[32] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, 18, pp. 1527-1554, 2006.

[33] T. Tieleman, "Training restricted Boltzmann machines using approximations to the likelihood gradient." In *Proceedings of the 25th international conference on Machine learning*, pp. 1064-1071. ACM, 2008.

Keynote 2

Julius Smith and Kurt Werner: Recent Progress in Wave Digital Audio Effects

Abstract The digital audio effects (DAFx) community has contributed significantly to advancements in "virtual analog" modeling of classic audio effects in software. Practicing musicians have enjoyed a growing list of classic analog gear available now as digital audio plugins as a result.

One competitive approach is the Wave Digital Filters (WDF) formulation pioneered by Alfred Fettweis. WDFs have been around since the early 1970s and have found much use in VLSI implementations of digital filters, where superior numerical robustness is especially important. Since the early 2000s, this framework has been applied increasingly to musical acoustic modeling and digital audio effects, and its range of applicability has been recently expanded considerably to include arbitrary circuit topologies and multiple nonlinear elements.

The closely-related Digital Waveguide Framework (DWF) similarly uses wave variables (traveling-wave components) because wave-propagation delay lines can be implemented super efficiently as circular buffers. As a result, it is straightforward to combine these two paradigms to yield wave-variable models of a mixture of distributed and lumped systems, such as a lumped hammer model striking waveguide string in the case of a piano model.

WDFs use wave variables in the context of lumped modeling where, by definition, wave propagation does not occur. How then can we understand the use of wave variables in WDFs? This talk includes a somewhat alternative development of WDF principles based on the way that wave variables can be used to resolve implicit relationships in modular discrete circuit models.

The complexity of audio circuitry has often stressed the state of the art of WDFs, especially for nonlinear models. The DAFx community has contributed many new techniques in response. This talk will review these contributions and point out remaining issues for future research in WDF theory.

In addition to their theoretical appeal, desirable energetic/numerical properties, and fine-grained modeling fidelity, WDFs are also attractive to virtual-analog algorithm designers as an elegant modular software framework. Implementing WDFs as a hierarchical object-orient tree in software such as C++ can yield readable and reusable code, with clear high-level descriptions of circuits and digital audio processing. We include examples of practical wave digital modeling, and demonstrations of real-time performance.

LOW-DELAY VECTOR-QUANTIZED SUBBAND ADPCM CODING

Marco Fink, Udo Zölzer

Department of Signal Processing and Communications,
Helmut-Schmidt University Hamburg
Hamburg, Germany
marco.fink@hsu-hh.de

ABSTRACT

Several modern applications require audio encoders featuring low data rate and lowest delays. In terms of delay, *Adaptive Differential Pulse Code Modulation* (ADPCM) encoders are advantageous compared to block-based codecs due to their instantaneous output and therefore preferred in time-critical applications. If the the audio signal transport is done block-wise anyways, as in Audio over IP (AoIP) scenarios, additional advantages can be expected from block-wise coding. In this study, a generalized subband ADPCM concept using vector quantization with multiple realizations and configurations is shown. Additionally, a way of optimizing the codec parameters is derived. The results show that for the cost of small algorithmic delays the data rate of ADPCM can be significantly reduced while obtaining a similar or slightly increased perceptual quality. The largest algorithmic delay of about 1 ms at 44.1 kHz is still smaller than the ones of well-known low-delay codecs.

1. INTRODUCTION

A major part of today's communication is based on packet-switched networks and especially IP-networks. The requirement for data rate compression techniques to transmit audio signals with a reduced data rate but good quality emerged in the very beginning of digital audio transmission and led to development of many specialized audio codecs, which are utilized in several applications. However, certain applications with a strong interactive characteristic, like wireless digital microphones, in-ear monitoring, online gaming, or Networked Music Performances (NMP) [1, 2], define the requirement of very small delays additionally. Several coding approaches, like the ultra-low delay codec (ULD) [3, 4], the aptX® codec family, and the *Constrained-Energy Lapped Transform* (CELT) part of the OPUS codec [5, 6], were presented to fulfill the additional low-delay requirement.

This study shall reveal the performance gain when a broadband ADPCM codec, as described in [7, 8, 9], is operating in subbands by extending it with a filterbank. Additionally, the scalar quantizer is replaced with a vector quantizer to create a vector-quantized subband ADPCM (VQSB-ADPCM). In contrast to the transform-based OPUS codec, the ULD and aptX® are also ADPCM codecs. However, the ultra-low delay codec is a broadband approach and an open loop implementation. In other words, no error feedback is used. Specific system differences to the aptX® codec family can't be asserted due to missing documentation. A subband ADPCM coding structure with a vector quantization was also proposed in [10]. However, the authors utilized known ADPCM predictors of the G.721 standard whereas in this work a prediction based on lattice filters, optimized to the proposed coding structure is used. In addition, the comparability of the codec from [10] is unfeasible due to non-compliant test methods and material. Robustness techniques against bit errors for ADPCM codecs as shown in [11] are not considered in this study.

This study is structured as follows. The proposed codec structure with all its modules is explained in Sec. 2, whereas the undertaken codec parameter optimization is explained in Sec. 3. The evaluation with automated measurements is presented in Sec. 4. Sec. 5 summarizes the findings of this study and depicts possible enhancements of the proposed codec structure.

2. SYSTEM OVERVIEW

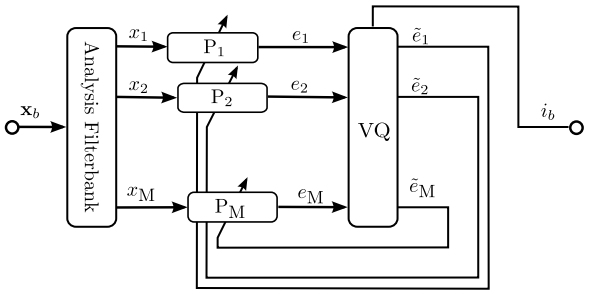


Figure 1: *Blocks scheme of encoder*

The proposed encoder is constructed block-wise as shown in Fig. 1. The b_th input signal block $\mathbf{x}_\mathrm{b}$ is fed to an analysis filter bank, resulting in M critically sampled subband signals, denoted as $x_1(b), \ldots, x_M(b)$. M is the number of bands and the block size of $\mathbf{x}_\mathrm{b}$. Every subband signal $x_1(b), \ldots, x_M(b)$ is individually processed by a distinct ADPCM encoder producing a residual signal $e_1(b), \ldots, e_M(b)$, which is massively reduced in entropy and amplitude by prediction. In a final step, a vector quantizer is applied to find a codebook entry $\mathbf{c}_b$ with index i_b describing the residual signal vector $\mathbf{e}(b) = [e_1(b), \ldots, e_M(b)]$ best.

At the decoder side (Fig. 2), the residual vector $\mathbf{e}$ is regained through a lookup operation in a codebook using i_b. The decoder uses the same predictors as the encoder to reconstruct the subband signals $\tilde{y}_1(b), \ldots, \tilde{y}_M(b)$. To reproduce a full-band signal with the original sampling a synthesis filter bank is used, which produces the final output block $\mathbf{y}_\mathrm{b}$, serialized in the continuous output signal $y(n)$.

The signals within the encoder are illustrated in Fig. 3. A monophonic tune played on a classical concert guitar is the exemplary input signal $x(n)$ in Fig. 3a). The corresponding fil-

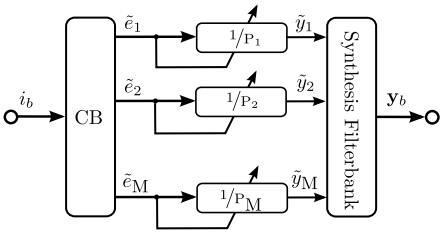


Figure 2: *Block scheme of decoder*

terbank output $x_1(b), \ldots, x_M(b)$ for $M = 8$ in Fig. 3b) shows that for stationary musical signals the energy of the input signal is mainly concentrated in the lower bands. Transient components like the plucking tend to show more energy in higher bands. The redundancy-reduced residual signals $e_1(b), \ldots, e_M(b)$ and the quantized residual signals $\tilde{e}_1(b), \ldots, \tilde{e}_M(b)$ are depicted in Fig. 3c) and Fig. 3d), respectively. Apparently, they can't be graphically distinguished when a 16 bit codebook is used. The corresponding codebook index $i(b)$ is plotted in Fig. 3e). When the codebook is sorted in order of ascending l_2-norm of the codebook entries $||\tilde{\mathbf{e}}||$, $i(b)$ tends to be small and hence the *most significant bit* (MSB) is mainly unused in contrast to the *least significant bit* (LSB).

All functional modules of the codec structure are explained in greater detail in the following.

2.1. Filter Bank

The authors considered two critically-sampled (nearly) perfect reconstruction filter banks:

2.1.1. Cosine-modulated FIR filter bank

A nearly perfect-reconstruction FIR filter bank was designed similar to [12]. First, a prototype filter $H_{\text{PT}}(e^{j\Omega})$ with a -3 dB cutoff at $\Omega = \frac{\pi}{M}$ is created iteratively using the window method with a Hamming window. Then its impulse response $h_{\text{PT}}(n)$ of length N is modulated for all subbands $m = 1, \ldots, M$

$$h_m(n) = 2 \cdot h_{\text{PT}}(n) \cdot \cos\left((-1)^m \tfrac{\pi}{4} + \tfrac{\pi}{M}(m + \tfrac{1}{2})(n - \tfrac{N-1}{2})\right) \quad (1)$$

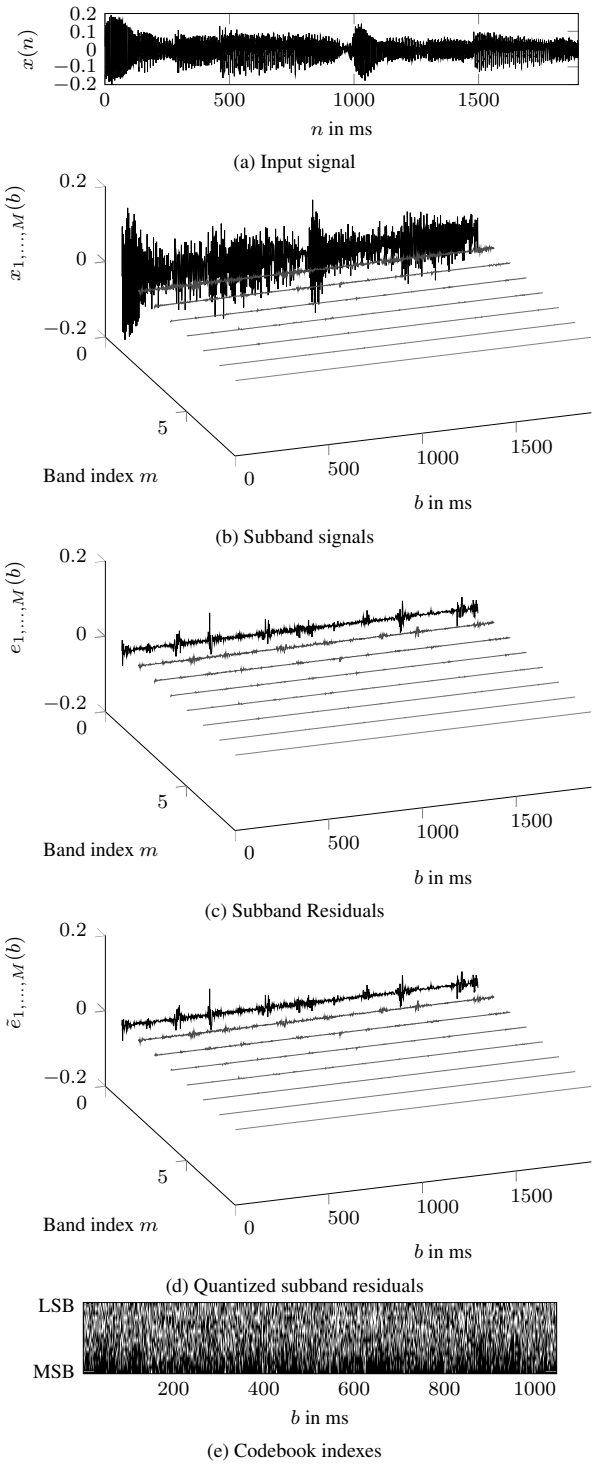
to obtain the impulse responses for the analysis filter bank. The corresponding synthesis filter bank impulse responses are simply time-reversed

$$g_m(n) = h_m(N - n). \quad (2)$$

The delay of a single FIR filter featuring a symmetric impulse response is known to be $\frac{N-1}{2}$ and hence the overall delay is $d_{\text{FIR}} = N - 1$. Since the study examines low-delay codecs the impulse response should be as small as possible to show a small delay. On the other hand, the side band attenuation scales with the filter length N.

2.1.2. Modified Discrete Cosine Transform (MDCT)

In addition to the FIR filterbank, a MDCT was chosen due to its well-known advantageous properties like low-frequency energy compression and time-domain aliasing cancellation (TDAC). Since it is a half-overlapped transform and applied twice, its algorithmic



(a) Input signal

(b) Subband signals

(c) Subband Residuals

(d) Quantized subband residuals

(e) Codebook indexes

Figure 3: *Signals throughout the encoder for a plucked nylon guitar sequence*

delay $d_{\text{MDCT}} = M$ is equal to the amount of bands. The impulse response for the m_{th} band is defined as

$$h_m(n) = \sqrt{\frac{2}{M}} \cdot w(n) \cdot \cos\left(\frac{(2n + M + 1)(2k + 1)\pi}{4m}\right). \quad (3)$$

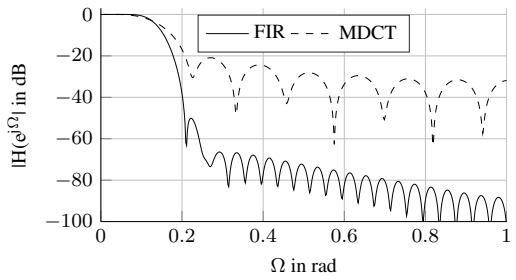


Figure 4: *Band $m = 1$ of FIR and MDCT filterbank for $M = 8$, $N = 51$*

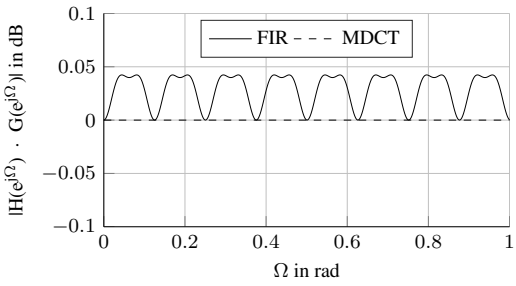


Figure 5: *Reconstruction error of FIR and MDCT filterbank for $M = 8$, $N = 51$*

Comparing the absolute frequency responses $|H_{\text{MDCT}}(e^{j\Omega})|$ and $|H_{\text{FIR}}(e^{j\Omega})|$ in Fig. 4 reveals that the stopband attenuation of the MDCT is significantly worse than in the FIR case. The reconstruction error of both filterbanks are depicted in Fig. 5. In contrast to the MDCT filterbank, the reconstruction of the FIR filterbank deviates up to $0.045\,\text{dB}$ from the perfect MDCT reconstruction. As demonstrated in [13], the design of a perfectly reconstructing FIR filterbank is possible when the impulse response length is restricted to a length of $2 \cdot M \cdot K$, where K is a positive integer. Nevertheless, the stopband attenuation is of predominant importance for the coding efficiency as derived later on.

2.2. Prediction

As previously mentioned, the idea of an ADPCM codec is to solely quantize prediction residuals. Therefore, M predictors $\text{P}_{1,\ldots,M}$ are utilized in every band to estimate the following sample $\hat{x}_m$ from prior values of x_m. Only the subband prediction error $e_m = x_m - \hat{x}_m$ is fed to the following quantization stage. At the decoder side, the same predictor the encoder utilizes is applied to predict the signal $\hat{y}_m$. The transmitted decoded prediction error signal $\tilde{e}_m$ is added to obtain the decoded subband signal $\tilde{y}_m$. To allow various signal conditions, the predictors filters are adapted using the *Gradient Adaptive Lattice* (GAL) technique [14]. To ensure a synchronous prediction filter adaption in encoder and decoder, the quantized error signal $\tilde{e}_m$ is utilized for the adaption step instead of the original error signal e_m.

The adaption of lattice filter coefficients is adaptive since the gradient weight of the lattice stage p

$$\mu_p(n) = \frac{\lambda}{\sigma_p(n) + v_{\text{min}}} \quad (4)$$

is computed using a base step size λ and normalizing it with an instantaneous power estimate $\sigma_p(n)$ at time instance n. v_{min} is a small constant avoiding a division by zero. The power estimate $\sigma_p(n)$ is computed recursively

$$\sigma_p(n) = (1 - \lambda)\,\sigma_p(n) + \lambda\,(f_{p-1}^2(n-1) + b_{p-1}^2(n-1)), \quad (5)$$

where $f_{p-1}(n-1)$ and $b_{p-1}(n-1)$ are the forward- and backward prediction error of the previous lattice stage at the previous time instance.

Since the signal characteristics differ drastically throughout the bands, they have to be individually adjusted. In this study, the prediction order N_m^p, base step size λ_m, and v_m of the filter adaption are carefully adjusted as described in Sec. 3.

2.3. Adaptive Vector Quantization

The process of vector quantization is defined as the mapping of an input vector to an equal-sized output vector, taken from a finite-sized codebook, resulting in the smallest distortion between the vectors. Due to the harmonic characteristics of sound and the imperfections of the applied filterbanks significant correlation between the subbands occur. Hence, individual scalar quantization would result in higher distortion than applying a vector quantization which can exploit the joint probability density function of the subbands. The euclidean distance between the current prediction error residual vector $\mathbf{e}$ and all codewords $\mathbf{C}_i$ in a codebook is minimized

$$\min_i \quad \left[(\mathbf{e} - \mathbf{C}_i)^T(\mathbf{e} - \mathbf{C}_i)\right] \quad (6)$$

to find the best-fitting match from the codebook and only the index i of the codebook table is transmitted.

The quantization is made adaptive by computing normalized subband residuals

$$\bar{e}_m = \frac{e_m}{v_m} \quad (7)$$

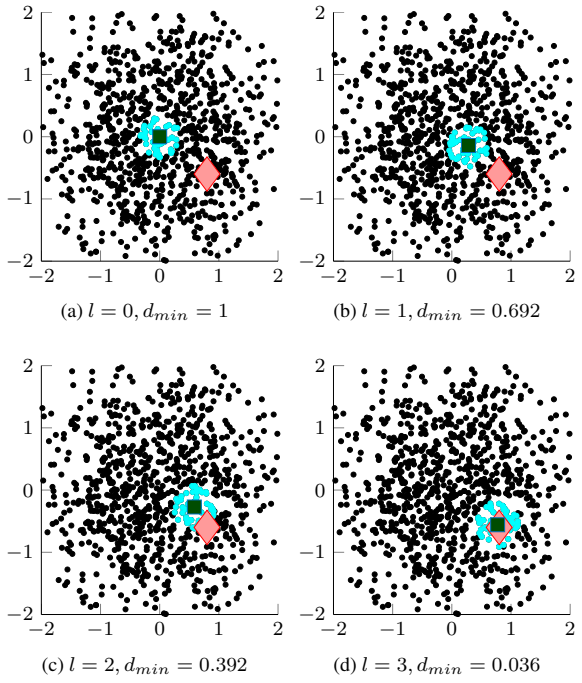
by dividing them by their recursively estimated envelope v_m to achieve a nearly constant signal variance [15] as described in [9]. Note that the computation the envelope estimation is signal-adaptive since different smoothing coefficients λ_{AT} and λ_{RT} are utilized for the attack and release case. Additionally, values of v_m smaller than v_{min} are clipped to v_{min}. Similar to [10] the distance function can be weighted

$$\min_i \quad \left[(\mathbf{e} - \mathbf{C}_i)^T \text{diag}(\mathbf{w})(\mathbf{e} - \mathbf{C}_i)\right] \quad (8)$$

using a weighting vector $\mathbf{w}$ to emphasize certain bands during the codebook search. In this study, the envelope estimation v_m is utilized to define $\mathbf{w} = v_1, \ldots, v_M$.

The codebook was designed using normally-distributed noise. The codebook size yields $M \times 2^{R_b M}$ entries. For $R_b = 2$ bits per sample and $M = 8$ subbands this corresponds to 8×65536 entries. Since a linear search in large codebooks is very costly, *Nearest*

Neighbour Search (NNS) was implemented. For that purpose, a second table $\mathbf{N_n}$ of size $K \times 2^{R_b M}$ containing the K indexes of the codewords with smallest euclidean distance has to be saved for every codeword.



(a) $l = 0, d_{min} = 1$ (b) $l = 1, d_{min} = 0.692$

(c) $l = 2, d_{min} = 0.392$ (d) $l = 3, d_{min} = 0.036$

Figure 6: *Illustration of the NNS search procedure for the 2D case and* 3 *iterations*

The search process is notated in Alg. 1 and illustrated in Fig. 6 for the 2D case, a codebook size of 2×1024 and $K = 50$. Initially, the euclidean distance between the normalized residual vector $\bar{\mathbf{e}}$ (marked in red) and the first entry of the codebook $\mathbf{C}_{i_{min}}$ (green), representing silence, is computed and set to the initial minimum distance d_{min}. Then the euclidean distance of all K nearest neighbours (cyan) to $\bar{\mathbf{e}}$ is computed and saved in $d(k)$. d_{min} is replaced by $d(k)$'s smallest value d_{nn} if it is smaller. Otherwise the search method terminates. This procedure is repeated $\mathrm{l}_{\max}$ times. Examining Fig. 6 shows, that the cloud of nearest neighbours is moving progressively in the direction of $\bar{\mathbf{e}}$. The codebook entry showing the smallest distance $d_{min} = 0.036$ is already found in the third iteration in this example and hence, only $4 \cdot 50 = 200$ computations of the vector distance had to be computed instead of 1024 for the linear codebook search.

Algorithm 1 *Nearest neighbor search*

$i_{min} = 1$
$d_{min} = \left[(\mathbf{e} - \mathbf{C}_{i_{min}})^T \mathrm{diag}(\mathbf{w})(\mathbf{e} - \mathbf{C}_{i_{min}})\right]$
for l = $[1, \ldots, \mathrm{l}_{max}]$ **do**
  for $k = [1, \ldots, K]$ **do**
    $i = \mathbf{N_n}(k, i_{min})$
    $d(k) = \left[(\mathbf{e} - \mathbf{C}_i)^T \mathrm{diag}(\mathbf{w})(\mathbf{e} - \mathbf{C}_i)\right]$
  end for
  $[d_{nn}, i_{nn}] = \min(d)$
  if $d_{nn} > d_{min}$ **then**
    break
  else
    $d_{min} = d_{nn}$
    $i_{min} = i_{nn}$
  end if
end for

3. OPTIMIZATION

As mentioned before, a variety of prediction parameters have to be optimized for this codec structure design. The optimization of codec parameters was done in a two-step approach and is restricted to a filterbank size of $M = 8$ in this study. At first, the authors wanted to roughly identify the order N_m^p and base step λ_m for all bands. For every band m, a chosen set of orders N_m^p, and both filterbanks a gradient descent algorithm is applied to find the base step size λ_m by minimizing the cost function

$$C_{\mathrm{gd}} = \frac{1}{F} \sum_{f=1}^{F} \sum_{n=0}^{N_f - 1} e_m^2(n) \tag{9}$$

describing the accumulated prediction error energy within a band m averaged over a large data set consisting of F tracks with a length of N_f samples, respectively. The actual computation of e_m was performed with a real-time C implementation of the encoder with disabled quantization. The utilized data set is the *Sound Quality Assessment Material* (SQAM) [16] dataset, consisting of 70 stereo tracks incorporating lots of different signal sources and mixes, which are recorded with a sampling rate f_s of 44.1 kHz. For all following evaluations the tracks were downmixed to a single channel.

In a second step, the optimization using simulated annealing using a perceptually motivated cost function similar to [17] was performed. Perceptual evaluation of audio quality is a tricky task due to the subjectivity of the nature of human hearing. Nevertheless, the *Perceptual Evaluation of Audio Quality* (PEAQ) [18] algorithm is established as standard tool for this purpose. A self-implemented tool, verified in [19], is fed with an original wavefile and a wavefile processed with encoder and decoder. The result of the PEAQ algorithm is the so-called *Objective Difference Grade*, ranging from -4 to 0 corresponding to the measured impact of coding artifacts describing the impairment as "very annoying" (-4) to "imperceptible" (0). The cost function

$$C_{\mathrm{ODG}} = \frac{1}{F} \sum_{f=1}^{F} \mathrm{ODG}_f^4 \tag{10}$$

emphasizes negative ODG scores since the fourth power of ODG scores is used. Hence, an optimization towards a globally good audio quality instead of excellent quality for many tracks but strong negative outliers can be expected. Additionally to N_m^p and λ_m, the minimum envelope amplitude $v_{\min}$, and the smoothing coefficients for the envelope estimation λ_{AT},λ_{RT} are optimized in a second step.

The results of the second optimization step are depicted in Tab. 1. The trend of the prediction order N_m^p over the bands m, shown in Tab. 1a), falls from about 120 to 20 for the MDCT and FIR case. In contrast, the base step sizes λ_m (Tab. 1b)) differ drastically for both filterbanks. While λ_m shows a concave trend ranging from 0.08 to 0.01 and back to 0.09 for FIR, the MDCT λ_m are

Table 1: *Overview of optimization results*

	(a) N_m^p		(b) $\lambda_m \cdot 1e^{-2}$		(c) $v_{\min} \cdot 1e^{-4}$	
Band m	FIR	MDCT	FIR	MDCT	FIR	MDCT
1	119	127	0.7945	0.4247	0.1182	0.1420
2	112	133	0.5950	0.9517	0.8280	0.0835
3	88	60	0.6225	0.9687	0.4174	0.0067
4	75	80	0.3590	0.9468	1.1363	0.1671
5	42	55	0.3138	0.9987	1.0234	0.1678
6	27	20	0.1339	0.9717	1.2486	0.0757
7	27	16	0.2826	0.9319	0.1754	0.1727
8	19	30	0.8991	0.9934	0.0052	0.0577

nearly constant 0.09 except for 0.04 at $m = 1$. Also the results for $v_{\min}$ in Tab. 1c) are very distinguishable. For $m = 2, \ldots, 6$ the minimum envelope amplitude is significantly higher in the FIR case. Unexpectedly, the envelope smoothing coefficients were optimized to very similar values ($\lambda_{\text{AT}} = 0.8, \lambda_{\text{RT}} = 0.1$) as in the broadband case [17] for both filterbanks.

4. EVALUATION

Using the optimized codec parameters, the evaluation of the codec structure can be discussed in the following. The evaluation is established using the PEAQ tool and the SQAM dataset as before. The proposed codec structured is compared to the optimized single-band ADPCM codec without noise shaping from [17] using $R_b = 3$ bits per sample, corresponding to 132.3 kbit/s for a sampling frequency $f_s = 44100$ Hz. The VQSB-ADPCM is configured with $R_b = 2$ bits, equivalent to 88.2 kbit/s, and hence every residual vector $\bar{\mathbf{e}}$ is encoded using 16 bit. A comparison with the same bitrate is not directly feasible due to the resulting huge codebook of size $8 \times 2^{3 \cdot 8}$. The ODG scores for the 3-bit reference and several codec configurations are shown in Fig. 7. The reference broadband codec achieves ODG scores in the range of $[-1.4, -0.075]$ and an average score of -0.5, indicating good quality throughout the test items. Only the castanets (27) and the accordion (42) sample produce scores below -1. The ODG scores for the VQSB-ADPCM with the FIR filterbank are similar to the reference codec for most items. Items (2-6), which are synthesized gong signals, horn (23), tuba (24), castanets (27), vibraphone (37), tenor (46), bass (47) show a massive degradation in audio quality in contrast to the reference. Apparently, low-pitched and percussive signals benefit from broadband encoding. Neglecting item (1-7) due to their minor importance for the applications mentioned in Sec. 1, the average ODG score constitutes -0.73. In other words, the outcome of the proposed codec structure is a relative bit rate reduction to 2/3 for the cost of 0.24 mean ODG score degradation. However, the standard deviation is rising from 0.21 to 0.50, indicating a larger variance of the determined audio quality.

Utilizing the MDCT filter bank and the related codec parameter performed significantly worse as apparent from Fig. 7. The scores for the FIR case can never be reached for the relevant items and is even less consistent. The average score of -2.54 clearly discloses the missing usability of the codec in this configuration.

So far, only the results for the linear codebook search were considered. The implemented NNS codebook search is not guaranteed to find the globally best codeword from the codebook and hence a degradation of performance in exchange of decreased computation time has to be expected. Fortunately, the curves for linear and NNS search only differ slightly throughout the test set except for bass (47) item. For $K = 100$ nearest neighbours, a mean deviation of 0.01 and 0.16 is identified for the FIR and MDCT case, respectively. The averaged computation time for encoding the SQAM test set with a mean item duration of 50 s with the non-optimized C implementation on a Desktop PC with i5-3570K CPU clocked with 3.4 GHz is depicted in Fig. 8. The broadband ADPCM reference required about 1.44 s and hence, is approximately 35 times faster than realtime. Applying the filterbank and the linear codebook search increases the encoding time up to 315.23 and 320.81 s for FIR and MDCT configuration, respectively. This results in an impracticable performance of 0.15 times realtime. Applying NNS decreases the encoding drastically to 7.02 and 7.52 s. The decoding times are clearly smaller since the inverse quantization is a simple table lookup operation. The reference broadband decoder required 1.30 s, whereas the MDCT and FIR VQSB-ADPCM took 2.5 s and 1.6 s. The described and some additional results are listed in Tab. 2.

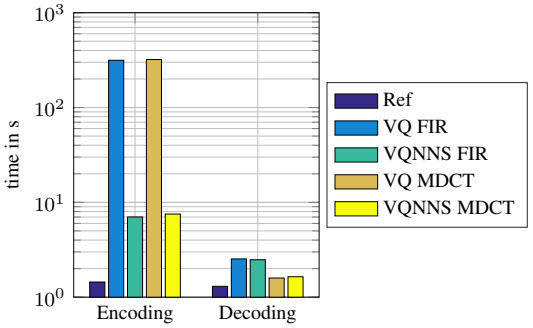


Figure 8: *Averaged encoding and decoding times*

5. CONCLUSION

The goal of this study was to develop a low-delay vector-quantized subband ADPCM codec structure offering a similar or slightly decreased performance at reduced bit rate in comparison to a broadband variant. The codec structure and all its modules were illustrated and optimized in a two-step approach to find meaningful codec parameters, especially for the subband predictors. The resulting initial codec based on a nearly perfectly reconstructing filterbank could reduce the data rate by 1/3 to 88.2 kbit/s for the cost of 1.1 ms delay and an average degradation of 0.28 ODG score in comparison to the broadband ADPCM codec without noise shaping from [17]. Significant performance gains are expected by further enhancements and extensions of the proposed codec.

For example, different filterbank designs can be considered and evaluated. Since transient signals tend to suffer from the proposed codec structure a transient detection could better control the adaption of the subband predictors. So far only a random codebook combined with nearest neighbor search was implemented but much advanced codebook design algorithms, like pyramid vector quantization [20], can be utilized. Considerable improvement can also be expected when the system is extended with pre- and post-filters to perform noise shaping.

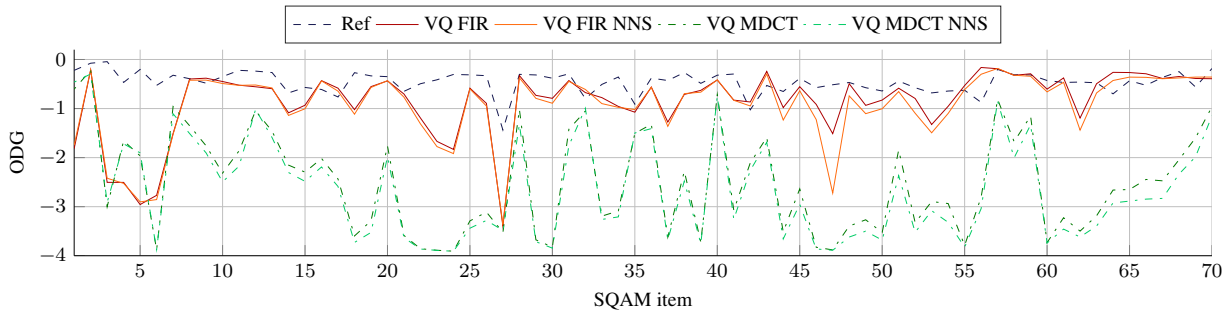


Figure 7: *ODG scores for SQAM items*

Table 2: *Overview of codec evaluation for different configurations for the relevant SQAM items (8-70)*

Configuration	Codebook search	Bit rate in kbit/s	Delay in ms	Mean ODG	Std. Dev. ODG	× realtime
Broadband ADPCM	None	132.3	0	−0.482	0.219	34.75
VQSB-ADPCM FIR	Linear	88.2	1.13	−0.728	0.497	0.159
VQSB-ADPCM FIR	NNS	88.2	1.13	−0.816	0.558	7.16
VQSB-ADPCM MDCT	Linear	88.2	0.18	−2.594	0.971	0.156
VQSB-ADPCM MDCT	NNS	88.2	0.18	−2.763	0.947	6.66

6. REFERENCES

[1] A. Carôt and C. Werner, "Network music performance-problems, approaches and perspectives," in *Proceedings of the Music in the Global Village*, Budapest, Hungary, 2007.

[2] A. Renaud, A. Carôt, and P. Rebelo, "Networked music performance: State of the art," in *Proceedings of the AES 30th International Conference*, Saariselkä, Finland, 2007.

[3] J. Hirschfeld, J. Klier, U. Kraemer, G. Schuller, and S. Wabnik, "Ultra low delay audio coding with constant bit rate," in *AES Convention 117*, San Francisco, USA, Oct 2004.

[4] J. Hirschfeld, U. Kraemer, G. Schuller, and S. Wabnik, "Reduced bit rate ultra low delay audio coding," in *AES Convention 120*, Paris, France, May 2006.

[5] J.M. Valin, G. Maxwell, T. Terriberry, and K. Vos, "High-Quality, Low-Delay Music Coding in the Opus Codec," in *AES Convention 135*, New York, USA, 2013.

[6] J.M. Valin, T. Terriberry, C. Montgomery, and G. Maxwell, "A High-Quality Speech and Audio Codec With Less Than 10-ms Delay," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 1, Jan. 2010.

[7] M. Holters, O. Pabst, and U. Zölzer, "ADPCM with Adaptive Pre- and Post-Filtering for Delay-Free Audio Coding," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, Honolulu, USA, April 2007.

[8] M. Holters and U. Zölzer, "Delay-free lossy audio coding using shelving pre- and post-filters," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008)*, Las Vegas, USA, March 2008.

[9] M. Holters, C.R. Helmrich, and U. Zölzer, "Delay-Free Audio Coding Based on ADPCM and Error Feedback," in *Proc. of the 11th Conference on Digital Audio Effects (DAFx-08)*, Espoo, Finland, 2008.

[10] D. Martinez, M. Rosa, F. López, N. Ruiz, and J. Curpián, "Sub-band/adpcm audio coder using adaptive vector quantization," in *4th WSES/IEEE World Multiconference on Circuits, Systems, Communications and Computers*, Vouliagmeni, Greece, July 2000.

[11] G. Simkus, M. Holters, and U. Zölzer, "Error resilience enhancement for a robust adpcm audio coding scheme," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, May 2014.

[12] F. Keiler, *Beiträge zur Audiocodierung mit kurzer Latenzzeit*, Cuvillier, 2006.

[13] Henrique S. Malvar, "Modulated qmf filter banks with perfect reconstruction," *Electronics Letters*, June 1990.

[14] Lloyd J. Griffiths, "A continuously-adaptive filter implemented as a lattice structure," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '77)*, Hartford, USA, May 1977.

[15] D. Cohn and J. Melsa, "The relationship between an adaptive quantizer and a variance estimator (corresp.)," *IEEE Transactions on Information Theory*, vol. 21, Nov 1975.

[16] European Broadcast Union, "EBU Tech. 3253-E: Sound quality assessment material," April 1988.

[17] M. Holters and U. Zölzer, "Automatic parameter optimization for a perceptual audio codec," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009)*, Taipei, Taiwan, April 2009.

[18] International Telecommunication Union, "ITU Recommendation ITU-R BS.1387: Method for objective measurements of perceived audio quality," November 2001.

[19] M. Fink, M. Holters, and U. Zölzer, "Comparison of Various Predictors for Audio Extrapolation," *Proc. of the 16th Int. Conference on Digital Audio Effects (DAFx-13)*, 2013.

[20] T. Fischer, "A pyramid vector quantizer," *IEEE Trans. Inf. Theor.*, vol. 32, no. 4, Sept. 2006.

SPARSE DECOMPOSITION OF AUDIO SIGNALS USING A PERCEPTUAL MEASURE OF DISTORTION. APPLICATION TO LOSSY AUDIO CODING

Ichrak Toumi, *

CNRS LMA
31 chemin Joseph-Aiguier
13402 Marseille Cedex 20
toumi@lma.cnrs-mrs.fr

Olivier Derrien,

Université de Toulon & CNRS LMA
31 chemin Joseph-Aiguier
13402 Marseille Cedex 20
derrien@lma.cnrs-mrs.fr

ABSTRACT

State-of the art audio codecs use time-frequency transforms derived from cosine bases, followed by a quantification stage. The quantization steps are set according to perceptual considerations. In the last decade, several studies applied adaptive sparse time-frequency transforms to audio coding, e.g. on unions of cosine bases using a Matching-Pursuit-derived algorithm [1]. This was shown to significantly improve the coding efficiency. We propose another approach based on a variational algorithm, i.e. the optimization of a cost function taking into account both a perceptual distortion measure derived form a hearing model and a sparsity constraint, which favors the coding efficiency. In this early version, we show that, using a coding scheme without perceptual control of quantization, our method outperforms a codec from the literature with the same quantization scheme [1]. In future work, a more sophisticated quantization scheme would probably allow our method to challenge standard codecs e.g. AAC.

Index Terms– *Audio coding, Sparse approximation, Iterative thresholding algorithm, Perceptual model.*

1. INTRODUCTION

Actually, state-of-the-art lossy audio coders (e.g. MP3, AAC, OGG) globally share the same structure [2]: The signal is first processed using a time-frequency (TF) transform, the transform coefficients are quantized according to a psycho-acoustic model and finally lossless binary coded. The TF transform is usually adaptive (i.e. it can switch between two pre-defined resolutions), perfectly invertible and introduces no redundancy. The most popular choice is the Modifed Discrete Cosine Transform (MDCT) [3]. Optimizing the quantization is performed in the frequency domain (i.e. independently in each time slot corresponding to one analysis/synthesis long window or a block of consecutive short windows). This process takes into account a perceptual weighting of frequency components computed by a psychoacoustic model.

However, this structure introduces two main limitations: 1) the invertible and non-redundant (i.e. orthogonal) TF transform takes advantage of an aliasing-cancelation property [3]. But when some spectral components are removed, because they are considered perceptually less relevant that others, some disturbing artifacts may become audible (called *birdies*). This occurs usually at low bitrate, where many components have to be removed. 2) Optimizing in the frequency domain does not allow to take into account the time-domain properties of audition (e.g. the temporal masking effect), which implies sub-optimal performance. Some propositions have been made to compensate for these drawbacks (e.g. [4, 5]), but it only resulted in marginal improvements.

The most important breakthrough in the last few years consists in performing a sparse approximation of the signal on an over-complete dictionary of waveforms instead of using an orthogonal transform [1]. This does not guarantee perfect reconstruction, but this property is not mandatory in a lossy codec. This approach was proved to significantly improve the audio quality on some audio files, especially at low bitrate. The proposed method relies on a Matching-Pursuit (MP) derived algorithm, which was modified to reduce artifacts called *pre-echo*. In its basic version, the MP is optimal with respect to the minimum Mean Square Error (MSE) criterion. Some improvements have been proposed in order to introduce perceptual weights in MP, but this was not selected in [1] because it significantly increases the complexity. Recently, a more sophisticated method for sparse audio signal decomposition was proposed [6]. It uses a *variational* algorithm and includes a perceptual weighting. However, the optimization is still performed only in the frequency domain and its implementation remains uncomplete (no quantization and binary coding were proposed).

In this paper, we describe a new method that is partially inspired by [1] and [6]. Our algorithm performs a sparse approximation on a over-complete dictionary, more precisely a union of MDCT bases, as in [1]. The optimization uses a variational algorithm, which appears to be more flexible than MP with respect to the distortion constraint. Our main contribution is that this algorithm performs the optimization in the TF plane, using a new TF audition model based on recent studies [7]. Then, we apply a quantization and binary coding scheme from the literature to evaluate the potential of this method in an audio-coding context. In [1], two codecs are described: #1 uses a simple *bit-plane* algorithm as a quantization and binary coding stage and #2 uses a so-called *Perceptual bit-plane* algorithm. It was shown that codec #1 usually does not perform as well as AAC, but codec #2 is able to outperform AAC, especially at low bitrate. In this paper, we only implement the simple bit-plane algorithm, because using the perceptual version is not straightforward in our coding scheme. For the same bitrates, we compare the audio quality for our codec and codec #1. In further works, we plan to improve our method by implementing a perceptual quantizer.

This paper is organized as follows: In section 2, we present the general method for sparse decomposition. In section 3, we motivate and describe our implementation. And in section 4, we compare our method to codec #1 described in [1].

* This work was supported by the joint French ANR and Austrian FWF project "POTION", refs. ANR-13-IS03-0004-01 and FWF-I-1362-N30.

2. THE METHOD FOR SPARSE DECOMPOSITION

Let's consider a block of audio samples noted as a vector $\mathbf{x}$ of size $1 \times N$, N being the number of samples. $\mathbf{x}$ can indifferently represent the whole signal, or a time-segment. We define the coding dictionary as a matrix $\mathbf{S}$ of size $M \times N$. Each row can be interpreted an elementary waveform, often called *atom*. The reconstructed signal is a linear combination of atoms, which can be written as:

$$\hat{\mathbf{x}} = \mathbf{a}\,\mathbf{S} \tag{1}$$

$\hat{\mathbf{x}}$ is the reconstructed signal and $\mathbf{a}$ is a vector of coefficients of size $1 \times M$, M being the number of atoms in the dictionary. This general scheme applies to state-of-the-art audio codecs: $\mathbf{S}$ is determined by the TF transform and $\mathbf{a}$ represents the transform coefficients to be quantized and coded. If we assume an invertible dictionary, $M = N$ and the coefficients can be easily computed using:

$$\mathbf{a} = \mathbf{x}\,\mathbf{S}^{-1}$$

This is the case for instance with TF transforms derived from the Fourier transform (DFT, DCT). An over-complete dictionary implies that $M > N$, which means that $\mathbf{S}$ is not invertible. In the general case, there is no $\mathbf{a}$ such that $\hat{\mathbf{x}} = \mathbf{x}$. Then, $\mathbf{a}$ must be computed using an iterative algorithm, with respect to a pre-defined distortion measure. For instance, the MP algorithm [8] finds $\mathbf{a}$ which minimizes the mean-square-error (MSE):

$$D(\mathbf{a}) = \|\mathbf{a}\,\mathbf{S} - \mathbf{x}\|^2$$

For audio signals, MSE is known to be poorly correlated to the auditory perception of distortion. In [6], a more efficient measure is proposed based on the concept of perceptual weights for each frequency component.

Here, we propose a more general formulation of the perceptual weighting technique. This comes from the observation that reasonably good perceptual distortion measures have already been proposed, e.g. the mean Noise-to-Mask Ratio (NMR) [9]. But these measures are usually associated to an analysis filterbank that follows the characteristics of human perception. Thus, we introduce a second matrix $\mathbf{P}$ of size $N \times K$ that represents a perceptual transform. We assume that $K > N$, i.e. this transform is redundant. In the general case, $\mathbf{P}$ and $\mathbf{S}$ are not directly related, because a good perceptual filterbank (with respect to the accuracy of perceptual modeling) is generally not so good for coding purposes (with respect to the efficiency of coding). The coefficients of this transform corresponding to the signal $\mathbf{x}$ can be written as:

$$\mathbf{p}_x = \mathbf{x}\,\mathbf{P}$$

We define the perceptual distortion measure as a weighted version of MSE computed in the perceptual-transform domain:

$$D_p(\mathbf{a}) = \|\,(\mathbf{p}_{\hat{x}} - \mathbf{p}_x)\,\mathbf{\Delta}_x\|^2 = \|\,(\mathbf{a}\mathbf{S} - \mathbf{x})\,\mathbf{P}\mathbf{\Delta}_x\|^2$$

Where $\mathbf{\Delta}_x = \mathrm{diag}\,(\mu_k(\mathbf{x}))$ is a diagonal matrix of size $K \times K$ containing perceptual weights μ_k associated to the components of $\mathbf{p}_x$. These weights depend on $\mathbf{x}$ and can be computed using a hearing model of masking. If we note $T_k(\mathbf{x})$ the masking threshold values corresponding to the signal $\mathbf{x}$, we choose $\mu_k(\mathbf{x}) = (T_k(\mathbf{x}))^{-\frac{1}{2}}$. Then, the perceptual distortion $D_p(\mathbf{a})$ can be interpreted as a mean NMR. Note that, if the perceptual transform performs a TF analysis with a sufficiently good time precision, both temporal and frequential masking can be modeled.

The general coding problem then consists in finding, for any input signal $\mathbf{x}$, the optimal vector of coefficients $\mathbf{a}$ which minimizes the perceptual distortion measure $D_p(\mathbf{a})$. The existence of a solution depends on the matrix $\mathbf{K} = \mathbf{SP}$ called the *mixture matrix*. It is quite easy to prove that the maximum value for the rank of $\mathbf{K}$ is N. The minimization problem would have a single minimum if $\mathbf{K}$ would have a full-rank (i.e. $\min\{M, K\}$), which is never the case here because $N < \min\{M, K\}$. In other words, there are many equivalent solutions. We must add a sparsity constraint on $\mathbf{a}$ to select the best solution, i.e. we add a regularization term Ψ to the distortion measure and then we minimize the following objective function:

$$\Phi(\mathbf{a}) = D_p(\mathbf{a}) + \Psi(\mathbf{a})$$

This class of problem can be solved using an iterative thresholding algorithm [10, 11]. However, when $\mathbf{K}$ has not a maximum rank (i.e. N), the convergence is not always satisfactory.

In the literature, Ψ is often assimilated to an ℓ_α norm on the coefficients, where α is usually equal to 1. Although ℓ_1 norm does not directly measure the amount of zero coefficients, one usually assume that the minimum ℓ_1 norm corresponds to a sparse solution [10]. Then,

$$\Phi(\mathbf{a}) = \|\,(\mathbf{a}\mathbf{S} - \mathbf{x})\,\mathbf{P}\mathbf{\Delta}_x\|^2 + \lambda\|\mathbf{a}\|_\alpha \tag{2}$$

where the λ parameter allows to set the tradeoff between the distortion constraint and the sparsity constraint.

3. OUR PROPOSED IMPLEMENTATION

Figure 1 shows a block-diagram of our coder. It is mainly composed of 3 parts: The adaptive analysis over a coding dictionary which is a union of MDCT bases, a psycho-acoustic model and a quantization and coding section. The main point consists in finding a suitable coding dictionary $\mathbf{S}$ and a perceptual transform $\mathbf{P}$ such that $\mathbf{K} = \mathbf{SP}$ has a maximum rank.

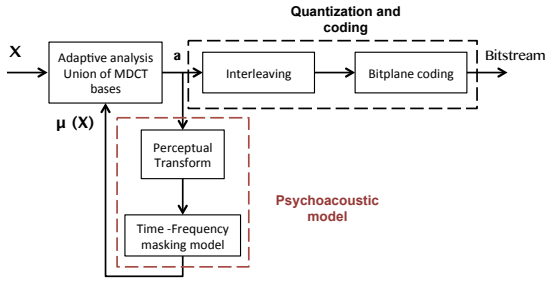


Figure 1: *Diagram of the proposed coder.*

3.1. Union of MDCT bases

First, we assume a coding scheme that splits the audio file in time-segments of reasonably long duration, called *macro blocks*, in order to be compatible with audio streaming applications (e.g. over the internet). Thus, we assume that the signal $\mathbf{x}$ corresponds to a macro-block. This approach requires an overlap between macro-blocks, and we wish that it does not introduce redundancy. Thus, a suitable choice for $\mathbf{S}$ is a union of MDCTs of different sizes, in the

same way as in [1]. In this paper, the authors proposed a union of 8 MDCTs: 64, 128, 256, 512, 1024, 2048, 4096 and 8192 bands. In this study, we restrained our dictionary to a union of 2 MDCTs with medium numbers of bands: 256 and 2048. In this implementation, one macro-bloc is composed of 16 short-windows (for 256-band) and 2 long-windows (for 2048-band). Choosing the number of MDCT sizes is tricky: More MDCTs, especially longer ones, would increase the efficiency of the decomposition, but would also degrade the efficiency of the coding stage: More MDCTs means more coefficients in $\mathbf{a}$, and even with the same number of non-zero coefficients, the additional zero coefficients require more coding bits.

The vector of MDCT coefficients $\mathbf{a}$ is made of two parts:

$$\mathbf{a} = [\mathbf{a_{256}}\ \mathbf{a_{2048}}]$$

where $\mathbf{a_{256}}$ stands for the coefficients of the short MDCT and $\mathbf{a_{2048}}$ for the long MDCT. The time-support corresponding to one macro-block is determined by the long MDCT, i.e. $N_{\text{mb}} = 3 \times 2048 = 6144$ samples. But practically, the optimization must be performed on an analysis-segment longer than a macro-block, in order to avoid sharp variations of coding parameters between successive macro-blocks. Thus, N corresponds to the length of one analysis segment. Here, we chose $N = 7 \times 2048 = 14336$ samples. This is illustrated on figure 2. Both $\mathbf{a_{2048}}$ and $\mathbf{a_{256}}$ are of size $1 \times M_{\text{as}}$, with $M_{\text{as}} = 6 \times 2048 = 12288$, and $\mathbf{a}$ is of size $1 \times 2M_{\text{as}}$, which means $M = 2M_{\text{as}} = 24576$. But practically, only 8192 coefficients (in the middle) are actually coded. The first and last 8192 coefficients are discarded. This part of signal will be analyzed and coded respectively in the previous and the next macro-blocks.

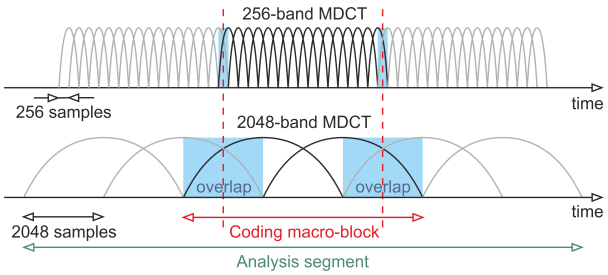


Figure 2: *MDCT windows in the coding dictionary for one macro-block, and corresponding analysis segment.*

3.2. Perceptual model

For the perceptual transform, we seek for a filterbank that follows the audition, but also with nice mathematical properties. The rank constraint on $\mathbf{K}$ requires that $\mathbf{P}$ is a full-rank matrix, i.e. its rank is N. A good choice is the ERB-MDCT, which is a near-orthogonal transform that follows the ERB frequency-scale [12]. In this transform, the time-resolution is adapted to the frequency resolution in each frequency band: Low frequencies have a high frequency-resolution and a low time-resolution, whereas high frequencies have a lower-frequency resolution and a higher time-resolution. We choose an ERB-MDCT with one band per ERB, which corresponds to $K = 15126$. One can check that we actually have $K > N$.

The square-value of ERB-MDCT coefficients is interpreted as the temporal and spectral density of energy. To obtain a masking threshold, we convolve the TF energy image with the TF masking kernel described in [7], and plotted on figure 3. We assume the additivity of masking patterns in the energy scale, which is known to sometimes under-estimate the masking level. Finally, we keep the minimum value of the masking threshold and the absolute threshold of hearing in quiet as described in MPEG #1 psycho-acoustic model [13], and get an estimation of the $\mu_k(\mathbf{x})$.

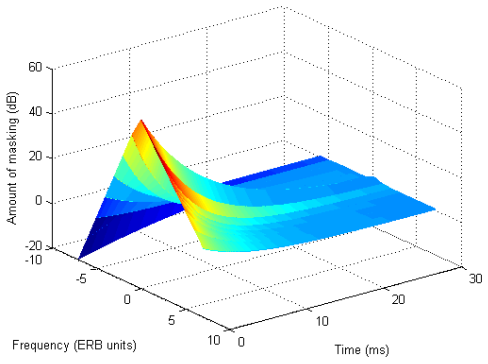


Figure 3: *Time-frequency masking kernel.*

3.3. Adaptive decomposition

For the optimization in the adaptive analysis step, we used the FISTA algorithm proposed by Beck et al. in [11]. The algorithm includes a gradient step followed by a shrinkage/soft-thresholding step. The general update rule in FISTA is

$$\mathbf{a}_i = \mathrm{T}_{\lambda\gamma}\left(G(\mathbf{y}_i)\right)$$

where i is the iteration index, γ is the gradient stepsize, $G(.)$ is the gradient of the distortion term D_p, $\mathbf{y}_i$ is a specific linear combination of the previous two iterations $\{\mathbf{a}_{i-1}, \mathbf{a}_{i-2}\}$ and $\mathrm{T}_{\lambda\gamma} : \mathbb{R}^M \rightarrow \mathbb{R}^M$ is the shrinkage operator:

$$\mathrm{T}_{\lambda\gamma}(\mathbf{a})_m = \text{sign}(a_m)\ \max\left(0, |a_m| - \lambda\gamma\right)$$

A typical condition which guarantees the convergence to a single minimiser $\mathbf{a}^\star$ is $\gamma \in \left[0, 1/\left\|\mathbf{K}^T\mathbf{K}\mathbf{\Delta}_x\right\|\right]$.

In the following, we assume for each macro-block:

- $\alpha = 1$, which amounts to perform a soft thresholding on the coefficients $\mathbf{a}$.
- $\gamma = 1/\left\|\mathbf{K}^T\mathbf{K}\mathbf{\Delta}_x\right\|$.
- For $i = 0$, $\mathbf{a}_0 = 0_{1\times M}$.

A crucial parameter for the optimization process is λ. We found out that a constant value for λ does not always produce the same sparsity ratio for $\mathbf{a}$. Thus, the suitable value for λ has to be set for each macro-block in order to get a constant sparsity ratio. Our goal was to get approximately the same rate of non-zero coefficients in $\mathbf{a}$ as with quantized MDCT coefficients in a MPEG-AAC bitstream. We measured that, for a monophonic AAC operating at 48 kbps, approximately 30 % of the MDCT coefficients

are equal to zero. Then, we target a sparsity ratio of 15 %, since our dictionary has a redundancy factor of 2.

Furthermore, we found out that setting the same value of λ for $\mathbf{a_{256}}$ and $\mathbf{a_{2048}}$ was not the optimal choice. Thus, we rewrite equation (2) as

$$\Phi(\mathbf{a}) = \| \left(\mathbf{aS} - \mathbf{x}\right) \mathbf{P} \mathbf{\Delta}_x \|^2 + \lambda_1 \|\mathbf{a_{256}}\|_\alpha + \lambda_2 \|\mathbf{a_{2048}}\|_\alpha \quad (3)$$

where λ_1 and λ_2 are respectively the regularization parameters corresponding to $\mathbf{a_{256}}$ and $\mathbf{a_{2048}}$. From informal listening tests, it appears that the best audio quality is obtained with $\lambda_1 > \lambda_2$, which means more non-zero coefficients in the long MDCT than in the short one.

In the literature, the choice of a suitable value for λ is known to be a difficult problem [14, 15], that would deserve a more specific study by itself. In this work, we use a simplified formula from [10] which is valid in the following conditions: (i) $\mathbf{K}$ is the identity operator and (ii) $\alpha = 1$:

$$\mathrm{T}_\lambda(\mathbf{x}) = \begin{cases} \mathbf{x} + \frac{\lambda}{2} & \text{if} \quad \mathbf{x} \leq -\frac{\lambda}{2}, \\ 0 & \text{if} \quad |\mathbf{x}| < \frac{\lambda}{2}, \\ \mathbf{x} - \frac{\lambda}{2} & \text{if} \quad \mathbf{x} \geq \frac{\lambda}{2}. \end{cases} \quad (4)$$

Then, assuming that $\forall k \in \{1 \cdots K\}$, $|\mathbf{p}_x(k)\mathbf{\Delta}_x(k)| \geq \frac{\lambda}{2}$, we get:

$$\lambda = 2Y_s \left\lfloor K(1 - sp) + 1 \right\rfloor ;$$

where sp is the amount of sparsity, Y_s are the values of $|\mathbf{p}_x \mathbf{\Delta}_x|$ sorted in increasing order, and $\lfloor . \rfloor$ denotes the rounding operator towards $-\infty$. Although $\mathbf{K}$ is not the identity operator in our case, this formula still gives a good approximation of the optimal λ in each macro-block. Thus, controlling sparsity only requires to a adjust the parameter sp.

3.4. Coefficients quantization and coding

When the coefficients $\mathbf{a}$ are computed, we carry out the quantification and binary coding preceded by an interleaving step which aims to group together the coefficients that are close in the time-frequency plane, and to make the coding algorithm more efficient. We chose the same binary coder as in [1]: a bit-plane coder, which is a special case of adaptive Rice-Golomb codes. It also performs an implicit quantization: the number of coding bits corresponds to the number of bit-planes used for coding. This method is particularly efficient when the sequence to be encoded exhibits long ranges of zeros, and thus is well suited for encoding sparse vectors. However, our interleaving scheme, as illustrated on figure 4, is different from the one described in [1]. The idea is to group the low frequencies at the beginning and the high frequencies at the end. We found out that our method reduces the bitrate, because it favors long ranges of zeros at the end.

4. RESULTS AND DISCUSSION

To test our method, we used the same audio material as in [1]. This collection of 4 musical pieces, sampled at 44.1 KHz, is described on table 1. Note that all the audio signals mentioned in this paper (unprocessed, re-synthesized and coded/decoded signals) can be downloaded from the companion webpage of this paper: http://potion.cnrs-mrs.fr/dafx2015.html.

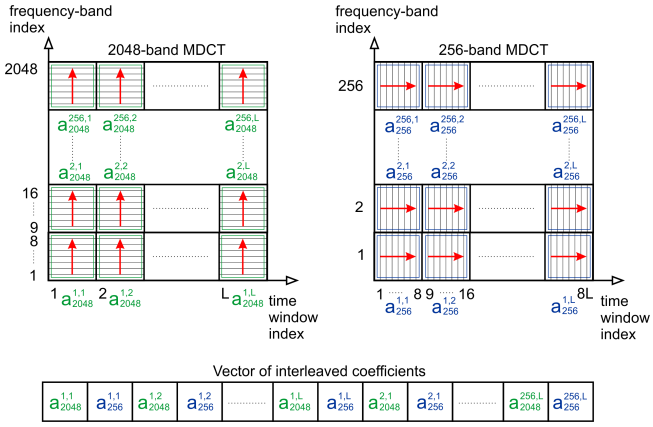


Figure 4: *Interleaving scheme for MDCT coefficients.*

Test signal	Description	Duration (s)
harp	Harpsichord	8.0
bagp	Bagpipes	11.1
orch	Orchestral piece	12.7
popm	Contemporay pop music	11.6

Table 1: *Test signals used for evaluation.*

4.1. Performance of the adaptive decomposition algorithm

In a first step, we analyze the results of the adaptive decomposition algorithm on the *harp* signal. For evaluation purpose, we re-synthesize the audio signal without quantization by applying equation (1). As explained in section 3.3, the sparsity ratio is about 10% on 256-band MDCT coefficients and 20% on 2048-band MDCT coefficients.

We can see on the spectrograms plotted on figure 5 that the original and re-synthesized signals are very similar. However, the reconstructed signal has less energy on the regions of the TF plane that are between the partials, which should correspond to masked regions. One can also notice that partials sometimes cross vertical structures corresponding to attacks, which can be associated to *pre-echo*. This is most probably due to the fact that our short MDCT has 256 bands, whereas in [1], the shortest MDCT has only 64 bands.

We also plot the *TF maps* of the coefficients $\mathbf{a_{256}}$ and $\mathbf{a_{2048}}$ on figure 6. In other words, we plot the magnitude of MDCT coefficients in the TF plane for both MDCT bases separately. First, one can see that the long MDCT exhibits a good frequency resolution, but a poor time-resolution. The short MDCT has opposite properties. One can also see that partials are essentially represented by long MDCT coefficients, and attacks by short MDCT coefficients. This proves that our optimization algorithm dispatches the energy between MDCT bases in an accurate way. However, this process is not perfect: some partials are represented in the short MDCT.

Finally, we plot on figure 7 the spectrograms of the re-synthesized signal after quantization and coding at 48 kbps and 24 kbps, for the same original test signal. At 48 kbps, one can see that the spectrogram of the re-synthesized signal is highly similar to the spectrogram before quantization and coding. However, at low

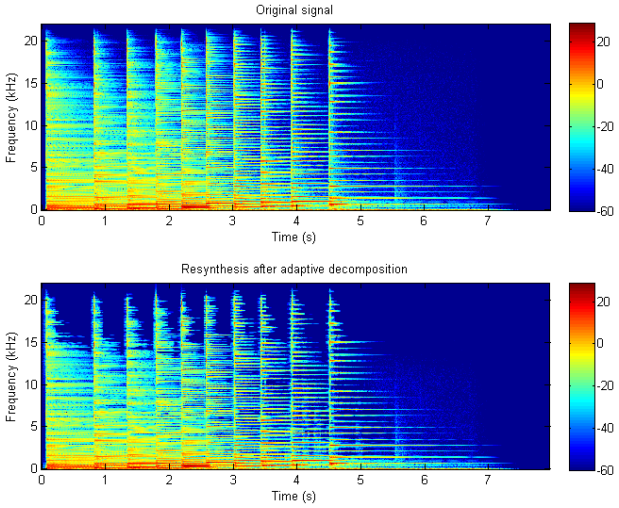


Figure 5: *Spectrograms of the* harp *signal*

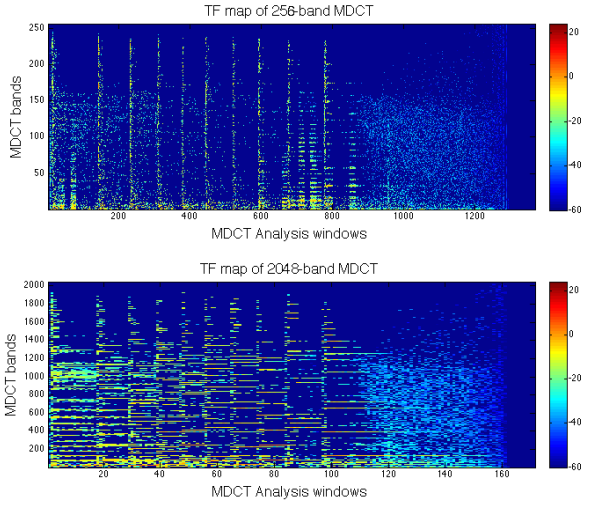


Figure 6: *TF maps of coefficients for the* harp *signal. Up: short MDCT. Down: long MDCT.*

bitrate (24 kbps), the spectrogram is somewhat different, with a slight degradation in high frequencies and less energy between partials.

4.2. Evaluation of audio quality

In order to evaluate the impact of our method on the audio quality and to compare our codec (denoted in section *codec A*) to the codec #1 proposed in [1], we organized a MUSHRA listening test [16] on the four audio signals described in table 1, with scores ranging from 0 (very bad quality) to 100 (excellent quality). Six versions of the test signals were evaluated by a total number of 16 listeners:

- A hidden reference,
- A 4 kHz low-pass anchor,
- Two coded versions with codec A at 24 and 48 kbps,
- Two coded versions with codec #1 at 24 and 48 kbps.

The listeners were post-screened according to the scoring of the reference: If the score attributed to the reference is lower than 80, the listener was judged unreliable and discarded. Finally, only 10 subjects were kept.

The results of MUSHRA listening tests are plotted on figure 8 and 9, respectively at 24 and 48 kbps (mean values and 95% confidence intervals). For both bitrates, one can see that results highly depend on the test signal. Signals with naturally sparse spectrums (*harp*, *bagp*) are associated to high audio quality at both bitrates. This can be interpreted as follows: The amount of perceptually relevant information in these signals is relatively low, and a good reconstruction in achievable at low bitrate. For *bagp*, the results obtained with both codecs are similar at 24 and 48 kbps but for *harp*, codec #1 is better rated. According to the listeners' feedback, this was mainly due to the fact that attacks are not as sharp with our codec as with codec #1, which is probably related to the length of the short MDCT. Signals with more dense spectrums (*orch*, *popm*) are associated to lower audio quality, especially a 24 kbps, probably because the amount of perceptually relevant information in these signals is higher. At 24 kbps, both codecs perform similarly on these two signals, but at 48 kbps, our codec is slightly better on *popm*, and much better on *orch*. Note that, at 24 kbps, both codecs were rated lower than the anchor, because a reduced band-pass was judged less disturbing than a high level of coding artifacts.

Finally, on average over the 4 test signals, our codec was rated slightly worse at 24 kbps, only because it does not perform as well on *harp*, and rated significantly better at 48 kbps, because it performs better on *orch* and *popm*.

5. CONCLUSION

In this paper, we described a method that performs an adaptive decomposition of audio signal on redundant coding dictionary (a union of 2 MDCT bases) using a variational algorithm, i.e. the optimization of a cost function taking into account both a perceptual distortion measure derived form a hearing model and a sparsity constraint. We applied a simple quantization and coding scheme from the literature (simple bit-plane coder) and compared the final audio quality to one achieved by codec #1 in [1], which uses the same quantization and coding scheme. We show that at low bitrate (24 kbps), our codec performs worse on signal with many sharp attacks, and performs similarly on other signals. This can be explained by the fact that our short MDCT has only 256 bands. At medium bitrate (48 kbps), our codec performs better on audio signals with a dense spectrum. This point is particularly interesting since in [1], codec #2 was able to outperform AAC but not on signals with dense spectrum (*orch* and *popm*), despite the use of a complex perceptual quantization scheme. Thus, these results are promising: As it is, our codec can not outperform codec #2 or AAC, but it may seriously challenge both if we design a suitable perceptual quantization scheme. This would also require to extend the coding dictionary by adding a shorter MDCT (128 or 64 bands).

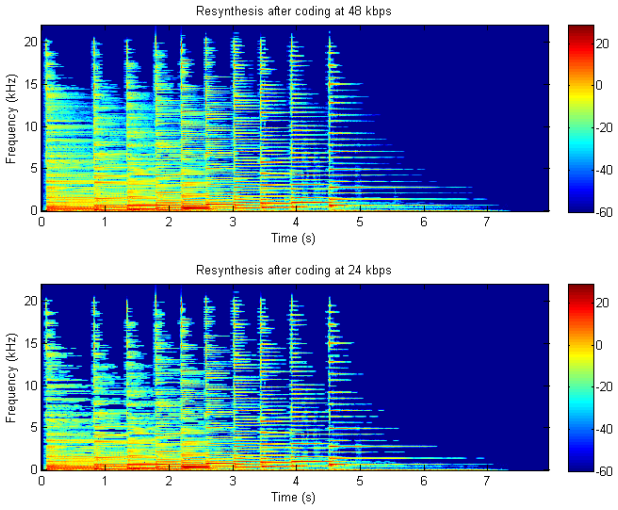


Figure 7: *Spectrograms of the* harp *signal after quantization and coding.*

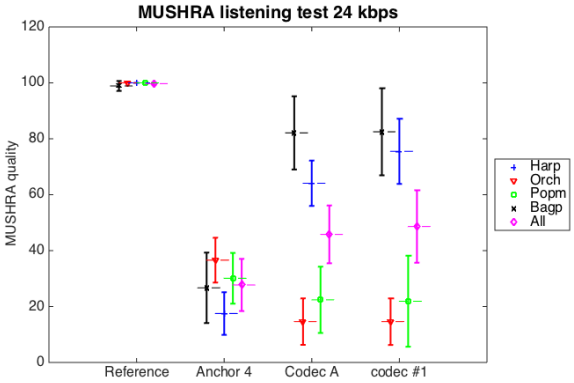


Figure 8: *Results of MUSHRA listening test at 24 kbps.*

6. REFERENCES

[1] E. Ravelli, G. Richard, and L. Daudet, "Union of MDCT bases for audio coding," *IEEE Tr. ASLP*, vol. 16, no. 8, pp. 1361–1372, Nov. 2008.

[2] A. Spanias, T. Painter, and V. Atti, *Audio Signal Processing and Coding*, Wiley, 2007.

[3] J. Princen and A. Bradley, "Analysis/synthesis filter bank design based on time domain aliasing cancellation," *IEEE Tr. ASSP*, vol. 34, no. 5, pp. 1153–1161, Oct. 1986.

[4] O. Derrien and L. Daudet, "Reduction of artefacts in mpeg-aac with mdct spectrum regularisation," in *Proc. 116th AES Convention*, Berlin, Germany, May 08-11, 2004.

[5] H. Najaf-Zadeh, H. Lahdidli, M. Lavoie, and L. Thibault, "Use of auditory temporal masking in the mpeg psychoacoustic model 2," in *Proc. 114th AES Convention*, Amsterdam, The Netherlands, March 22-25, 2003.

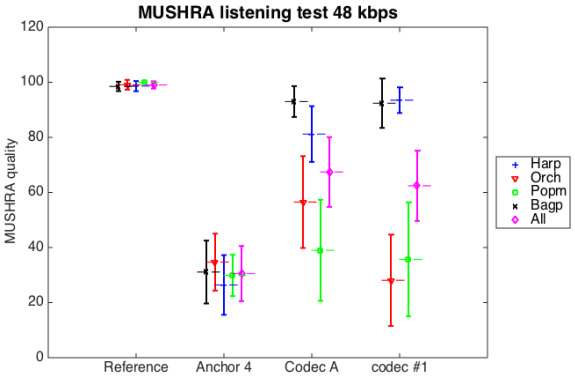


Figure 9: *Results of MUSHRA listening test at 48 kbps.*

[6] M.G. Christensen and B.L. Sturm, "A perceptually reweighted mixed-norm method for sparser approximation of audio signals," in *Proc. ASILOMAR*, Pacific Grove, California, Nov. 06-09, 2011.

[7] T. Necciari, *Time-Frequency Masking: Psychoacoustical Measures and Application to the Analysis-Synthesis of Sound Signals*, Ph.D. thesis, University of Aix-Marseille (in french), 2010.

[8] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Tr. SP*, vol. 41, no. 12, pp. 3397–3415, 1993.

[9] K. Brandenburg and T. Sporer, "NMR and Masking Flag : Evaluation of quality using perceptual criteria," in *Proc. 11th Int. Conf. of the AES*, 1992.

[10] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[11] Amir Beck and Marc Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[12] O. Derrien, T. Necciari, and P. Balazs, "A quasi-orthogonal, invertible and perceptually relevant time-frequency transform for audio coding," in *Proc. EUSIPCO*, Nice, France, Aug. 31 - Sept. 04, 2015.

[13] International Organization for Standardization, *ISO/IEC 11172-3 (Information Technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s) - Part 3: Audio*, 1993.

[14] P. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems*, Society for Industrial and Applied Mathematics, 1998.

[15] Robert Tibshirani, "Regression Shrinkage and Selection Via the Lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[16] ITU-R Recommendation BS.1534-1, "Method for the subjective assessment of intermediate quality levels of coding systems," Tech. Rep., International Telecommunication Union, Geneva, Switzerland, 2003.

APPROACHES FOR CONSTANT AUDIO LATENCY ON ANDROID

Rudi Villing,
Department of Electronic Engineering,
Maynooth University
Ireland
rvilling@nuim.ie

Victor Lazzarini,
Sound and Digital Music Research Group,
Maynooth University
Ireland
vlazzarini@nuim.ie

Dawid Czesak, Sean O'Leary
Department of Electronic Engineering,
Maynooth University
Ireland
dczesak@nuim.ie

Joseph Timoney
Sound and Digital Music Research Group,
Maynooth University
Ireland
jtimoney@nuim.ie

ABSTRACT

This paper discusses issues related to audio latency for realtime processing Android OS applications. We first introduce the problem, determining the difference between the concepts of low latency and constant latency. It is a well-known issue that programs written for this platform cannot implement low-latency audio. However, in some cases, while low latency is desirable, it is not crucial. In some of these cases, achieving a constant delay between control events and sound output is the necessary condition. The paper briefly outlines the audio architecture in the Android platform to tease out the difficulties. Following this, we proposed some approaches to deal with two basic situations, one where the audio callback system provided by the system software is isochronous, and one where it is not.

1. INTRODUCTION

The support for realtime audio on the Android platform has been shown to suffer from significant latency issues [1]. Although support for lower latency audio paths has been provided in later versions of the operating system, this provision can vary significantly from device to device. One of the difficulties here, as identified by its developers, is that there is no uniformity in the deployment of the system, which has to be able to adapt to a variety of vendor-designed hardware set-ups. Its competitor, iOS, on the other hand, has been built to run specific devices, which have much less variance, and, for this reason, it can deliver a better support for realtime audio to developers.

While low latency is critical for certain applications, for others, it is possible to design systems around a moderate amount of audio delays. In this case, the next requirement tends to be constant latency. In other words, we might not be worried if a given sound event is to be delivered a certain number of milliseconds in the future, but we would like to be able to predict with a certain degree of precision what this delay will be. Generally speaking, constant latency is guaranteed in a working full-duplex audio configuration, with respect to the input signal, since any modulation of this delay would be associated with sample dropouts.

The situation is, however, more complex when we are trying to synchronise audio output to external control inputs, operating asynchronously to the audio stream (fig.1). This situation typically arises, for instance, when trying to synchronise touch events, or responses to sensors, with specific parameter changes (e.g. onsets, pitch changes, timescale modifications). In this case, constant latency is not a given: it depends on the audio subsystem implementation. On Android, the audio subsystem is vendor-implemented, and, as noted, can vary significantly from device to device. For certain devices, achieving constant latency with regards to an asynchronous control is not much more than a matter of using a common clock, as the audio subsystem callback mechanism is tightly chained to the sample stream output (itself synchronised to a regular clock). On other devices, the situation is more complex, as there is no guarantee that the callback functions will occur with only small amounts of jitter (with regards to a common system clock). Furthermore, in these cases, stream time is not reported with a good degree of accuracy.

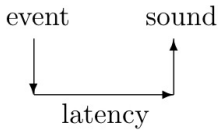


Figure 1. *Event-to-sound latency*

In this paper we will look at approaches to obtaining constant audio output latency with regards to external asynchronous control events. Constant latency is required when we need to estimate the exact time of the change of parameters in the sound stream in response to a control input. For instance, if we want to try and synchronise musical beats to a step detection algorithm, we will need to know how the time between the sending of a tempo adaptation command and the appearance of the effect at the output.

The paper is organised as follows: an overview of the Android audio architecture is provided, which will identify the difficulties with latency control, and identify the various sources of delays in the system. This will be followed by an examination of approaches to achieving constant latency in the two cases mentioned above: isochronous and anisochronous callback mecha-

nisms. The paper will demonstrate how it is possible to minimise latency jitter in these two situations, describing the algorithms employed and their results. The paper concludes with a brief discussion of the applications for the principles outlined here and some indications of future work.

2. ANDROID AUDIO SYSTEMS

The audio implementation in Android is built on a number of layers [2]. The lowermost components are not exposed to developers, and they can only be manipulated indirectly via the topmost application programming interfaces (APIs), of which there are a few. High-level development for Android uses a version of the Java language, and the simplest APIs are provided as part of the Software Developer Kit (SDK). The most flexible of these is AudioTrack, which provides a blocking-type audio writing functionality (synchronous), but it is still considered not enough low-level for any serious audio application.

For this, Android provides a C API based on the OpenSL ES 1.0.1 specification [3] in their Native Developer Kit (NDK). The NDK is a collection of C/C++ APIs that can be used to build dynamic modules for Java-based applications, whose functionality is accessed through the Java Native Interface. The NDK allows developers to port large portions of C/C++ code, and in the case of audio applications, to bypass the performance problems associated with Java garbage collection. However, it is important to note that both the Java AudioTrack and the NDK OpenSL ES APIs use the same underlying audio implementation. The documentation is clear in saying that using the latter does not guarantee better performance in terms of audio latency, although in some systems a "fast" audio mixer signal path might be available to NDK-based applications [4].

2.1. OpenSL ES

The OpenSL ES API provides the means to open audio streams for input and output, and to access these asynchronously via a callback mechanism. This method is generally the standard in audio APIs across platforms, where the developer provides the code to fill in audio buffers at the times requested by the system.

For the developer, this gets exposed via an enqueue mechanism: after the device is initialised and opened, the developer starts the process by enqueuing a single buffer, and registering a callback. This is invoked by the system, according to the specification, when a new buffer is required, and so the callback will include code to generate the audio and to enqueue it.

The specification does not say anything in relation to the timing of these callbacks. In others words, depending on the implementation, these can either be relied upon to happen at regular times (isochronous behaviour) or not (anisochronous). Depending on this, different strategies are required for the implementation of constant latency.

OpenSL also provides some means of polling the system for the current stream time. However, it is not clear from the specification how this should be implemented, and to which point in the audio chain it refers. In reality, we have observed that the information obtained is not reliable across different devices, so it is not always usable.

2.2. Lower levels

The implementation of OpenSL ES sits on top of the AudioTrack C++ library, which is also the backend of the Java AudioTrack API, as noted above. Although the two libraries share the name, they are actually distinct, the Java API occupying the same level as OpenSL on top of the C++ library. The following is a description of the lower-level implementation in Android version 5.0 (fig.2).

The actual code implementing the OpenSL exposed functionality is a thin layer. The enqueueing code simply places a pointer to the user-provided memory in a circular buffer. The data in this memory location is consumed by an AudioTrack function, which copies it into its own buffer. Once all supplied data is read, it issues the user-supplied callback, which should enqueue more data. The OpenSL specification asks for users to implement double buffering of audio data, ie. enqueueing one buffer while filling a second one. However, the implementation is clear: the enqueueing is only requested when the buffer data has been copied completely. Thus double buffering is not really a necessity.

AudioTrack shares its buffer memory with the next lower level service, AudioFlinger. This is responsible for mixing the audio streams in its own playback thread and feeding the Hardware Abstraction Layer (HAL), which is the vendor-implemented part of the code that communicates with the audio drivers running in the Linux kernel. Some devices implement a feature provided by AudioFlinger to reduce latency, a "fast mixer track", which, when available, is used if the developer employs a specific buffer size and sample rate dictated by the hardware, via OpenSL. The presence of this feature can be enquired via a Java call provided by the AudioTrack API, which also allows developers to obtain the native parameters for it. Surprisingly, it is not possible to do this via the NDK.

The consumption of audio data is ultimately linked to the HAL implementation, and so the timing behaviour of the callback mechanism is influenced by this. The HAL also is supposed to supply the information regarding stream time, which is exposed by the OpenSL ES, thus its reliability is related to how well this is implemented there.

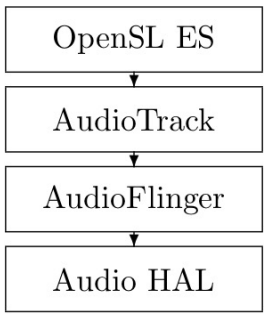


Figure 2. *The Android audio stack*

2.3. Achieving low and constant latency

It is clear from this discussion, that any approach to reducing latency is highly dependent on the hardware involved. Some recommendations have been provided, such as using the native buffer size and sample rates, but these are not guaranteed to have significant effect on all devices. Avoiding the interference of the

Java garbage collector is also important for constant latency, which in practice means migrating all relevant code from the SDK level to the NDK. In general, an analysis of a device's behaviour in relation to its callback timing appears to be needed for a full assessment of its particular latency improvement potential (both in terms of low and constant characteristics).

3. CONSTANT LATENCY APPROACHES

To achieve approximately constant latency, we first evaluate the most basic approach using OpenSL ES, namely enqueueing sound data in the OpenSL buffer queue as soon as possible after the request occurs. Thereafter, we examine approaches which instead estimate the play head position in the output stream and enqueue sound data at a fixed delay relative to that play head position.

All positions are defined relative to the start of the audio data stream. We define the enqueued position as the position within the data stream corresponding to the last sample enqueued to date. We define the true play head position as the position in the data stream corresponding to the sound that is currently being output from the device. In Android devices the delay between the true play head position and the enqueued position ranges from a few tens of milliseconds to more than 100 ms for some devices [1]. In general, if this delay is constant it should be easier to achieve constant latency between sound request and sound output.

To date, we have not encountered any method for measuring this latency in an absolute manner. To estimate the latency some authors measure the audio loopback delay, based on a round-trip path (input to output): an audio signal is fed to the input and the time it takes for it to appear at the output is measured. This can be achieved via the Larsen effect [5], or via a custom loopback audio connection. From this, it is estimated that the output latency is half of this time [6]. Nevertheless, it is not guaranteed that the audio processing path is symmetrical and therefore that the output latency is equal to the input latency.

Other approaches, which are more appropriate to the issue of constant latency, as discussed in sect. 1, include measuring the delay between a touch event on the Android device (or some external input) and the corresponding sound output (as illustrated by fig.1). However, even here there is a difficult to quantify (and usually variable) delay involved in receiving and processing the source event before a sound request is ever issued. Therefore, for this work, we developed an alternative approach based on the relative time at which sound requests were issued and the relative time at which sound outputs occurred.

3.1. Evaluation methodology, equipment and materials

Two different Android devices were used in this work: A Sony Xperia Z2 and a Motorola MotoG (first generation). Both devices were running Android 4.4.4. The most relevant audio properties of both devices are shown in Table 1. These devices were chosen as exemplars of a high end device (the Xperia Z2) and a mid-range device (the MotoG). As there are a great variety of Android devices and capabilities, this work with just two devices should be considered a preliminary examination of the problem space.

Table 1: *Key Audio Properties*

Device	Low latency	Native sample rate	Native buffer size (frames) / duration (ms)
Xperia Z2	No	48000	960 / 20
MotoG	No	44100	1920 / 43.5

A custom Android app was developed which included a C based NDK component and a java component. The NDK component implemented the sound engine as a thin layer on top of OpenSL ES and was responsible for all timing measurements (measured using clock_gettime specifying CLOCK_MONOTONIC). The Java component implemented test runs which consisted of a sequence of 500 sound requests issued at anisochronous intervals between 400–500 ms apart. (This interval was chosen so that even substantial latency jitter would not create any doubt about which sound request and which sound output correspond to one another.) The sound engine logged the precise time (in microseconds) at which each request was made and stored this in a log file along with other data required for subsequent processing.

Each sound request resulted in production of a short, 10 ms, 1000 Hz tone pip (with instant attack and release, limited only by the analogue hardware response). The audio output of the Android device was attached to the line input of a USB sound device (Griffin iMic) and recorded as a RIFF-Wave file (mono PCM, 16 bits, 44100 Hz) on a PC. This soundfile was then post-processed with threshold based onset detection to determine the times (within the audio file) at which tone pips occurred.

It is not possible to directly synchronize the clock used to measure sound request times with that used to measure sound onset times using standard Android devices. Therefore it was necessary to convert from raw times using these unsynchronized clocks to times that can be compared. To achieve this, all raw request times from the log file were made relative to the time of the first request (by subtracting the first request time from each of them). Similarly all raw sound onset times from the audio file were made relative to the time of the first sound onset (by subtracting the first sound onset time from each of them). After this calculation the first request and sound onset time are both zero and this allows subsequent request and onset times to be compared. The initial real world latency between the first request and the first sound onset cannot be measured (because the clocks cannot be synchronised) and appears as zero. If the latency is constant, all subsequent sound onsets occur at precisely the same time offsets as the corresponding requests and the time difference between them should be zero. In contrast, if there is latency jitter, the sound onset times will differ from the request times by an amount equal to the jitter.

In the subsequent evaluations it was noticed that there was a slight mismatch between the clock used to time requests on the Xperia Z2 and the sample clock in the USB microphone audio input to the PC. This mismatch caused the two clocks to drift apart by 1-2 ms over the duration of a test run. Therefore Xperia Z2 plots shown in the following sections have been de-trended to compensate for the drift which is unrelated to the latency algorithms being evaluated.

3.2. Latency using the Next Buffer approach

The most basic approach to approximating constant latency with OpenSL ES is to enqueue audio data as soon as possible after it is requested. In other words:

$$insertPos = enqueuedPos \quad (1)$$

where *insertPos* is the insertion position of the sound in the data stream and *enqueuedPos* is the end of data stream position after the most recent callback preceding the request has completed.

In general the Next Buffer approach can be expected to yield at least one buffer duration jitter as the request may occur just before a callback is due, resulting in a very short request to enqueue delay, or just after a callback has completed, resulting in a longer request to enqueue delay. For small buffer sizes (for example 20 ms) this jitter may be acceptable, but for larger buffer sizes (for example the 43 ms of the MotoG) the jitter can become audible.

Furthermore, if OpenSL callbacks occur at isochronous intervals, then two requests occurring within one buffer duration of each other will never be enqueued more than one buffer apart (even if they occur just before and just after a callback). If, however, callbacks do not occur at isochronous intervals (that is, some inter-callback intervals are shorter than others) then it is possible that two requests occurring within one buffer duration of each other in real time may be enqueued more than one buffer apart resulting in jitter that is even greater than one buffer duration in this case.

The results for the Xperia Z2 are shown in Figure 3.

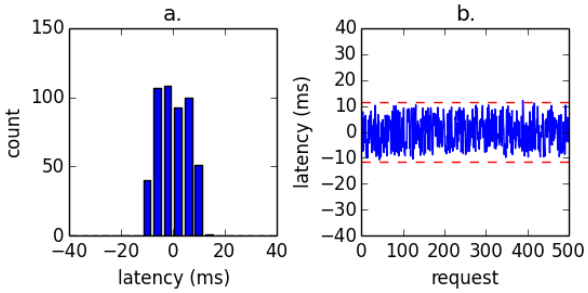


Figure 3 Relative latencies using Next Buffer on the Xperia Z2: (a) histogram, (b) values. Dashed lines show two standard deviation boundaries.

In this case the latency jitter is reasonable and limited to a range that is just larger than one buffer duration (-10.7 to 11.4 ms). Closer investigation showed that OpenSL callback times were nearly isochronous for this device.

Figure 4 shows the MotoG results and in this case the latency jitter is much larger than the Z2.

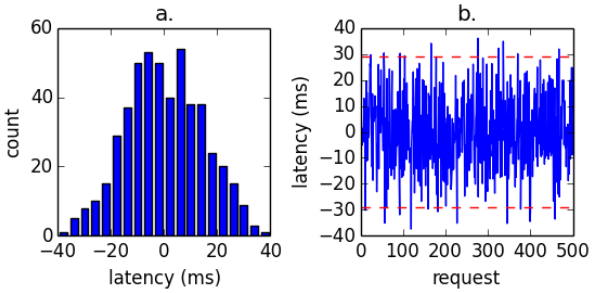


Figure 4 Relative latencies using Next Buffer on the MotoG: (a) histogram and (b) values. Dashed lines show two standard deviation boundaries.

In fact the latency jitter for the MotoG is even larger than one buffer duration (95% of the values were within a 58 ms range). This range is really too large and can result in audible jitter. The main cause for the jitter appears to be that the callback is not invoked at regular intervals on the MotoG. Instead a typical sequence of callback intervals measured in milliseconds might be: 40, 60, 20, 60, 40, 40, 80, 30, and so on.

Therefore it appears that the Next Buffer approach is only suitable if two conditions are satisfied by the device: (1) the native buffer size is relatively small, and (2) callbacks are invoked at isochronous intervals. For devices which do not satisfy these requirements, such as the MotoG, another approach is required.

3.3. Latency using the OpenSL position

Rather than attempting to enqueue a sound as soon as possible after the request and suffering the jitter that results it would be better if it was possible to enqueue the sound such that it was output a constant latency after the request. If the current play head position is known, then constant latency can be achieved by adding the sound to the data stream at some fixed delay relative to the play head position. This fixed delay must be large enough that the desired insert position of the sound is never earlier than the already enqueued position despite any callback jitter.

As discussed previously, the OpenSL ES API defines a function to read the current play head position and this is direct estimate of the play head position which may be used to calculate an appropriate insert position for the requested sound data as follows:

$$insertPos = requestSLPos + fixedDelay \quad (2)$$

where *requestSLPos* is the OpenSL position at the time of the request and the fixed delay (which was determined empirically for each device) guarantees that the insertion position is never earlier than the end of data already enqueued. This fixed delay does not appear in subsequent plots due to the relative nature of the times used.

Although the relative latencies for the Xperia Z2 using the Next Buffer approach were already quite good, we decided to evaluate the achievable latency jitter using the OpenSL position also. The results are shown in Figure 5.

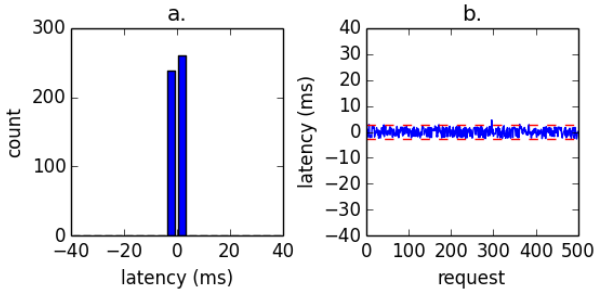


Figure 5 Relative latencies using OpenSL position on the Xperia Z2: (a) histogram, (b) values. Dashed lines show two standard deviation boundaries.

It is clear that inserting sounds a fixed delay after the OpenSL position (measured at the time of request) has substantially reduced the latency jitter (95% of the values are in 5.6 ms range) relative to the Next Buffer approach.

When this latency approach was applied to the MotoG the results improved somewhat relative to the NextBuffer approach but were still much worse than the Xperia Z2 as can be seen in Figure 6.

In this case, 95% of the latencies are contained within an 36.8 ms range (compared to a 58 ms range previously). This is still a relatively large range and, unlike the Xperia Z2, there are a number of outlier values which substantially depart from the typical range.

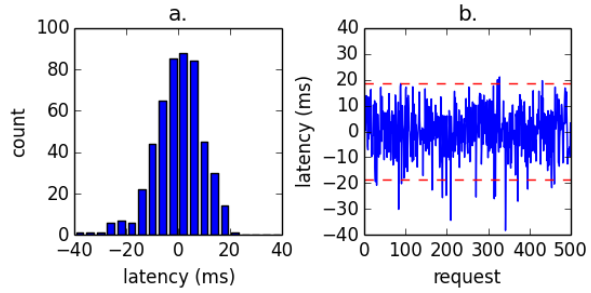


Figure 6 Relative latencies using OpenSL position on the MotoG: (a) histogram, (b) values. Dashed lines show two standard deviation boundaries.

Closer examination of OpenSL positions on the MotoG indicated that the intervals between OpenSL positions on consecutive requests did not match the intervals between the real time of those requests very well. Further investigation indicated that the API call to read the OpenSL position appears to read a cached position value which is not always up to date rather than reading the real position from the low level hardware or driver.

In some cases reading the OpenSL position twice 3 ms apart suggested that the position had not advanced at all whereas at other times a similar pair of requests separated by just 3 ms of real time indicated that OpenSL position had changed by as much as 30 ms. Therefore a better approach is still required for the MotoG and similar devices. As noted in section 2, the implementation of the position polling in Android devices cannot be relied on, and thus this approach is not general enough for all devices.

3.4. Latency using the Filtered Callback Time

Results to date indicated that the interval between callbacks on the MotoG was not even approximately constant. Nevertheless the mean callback interval precisely matches the buffer duration. This is to be expected: if the mean callback interval were any larger it would mean that the buffer level (the difference between the true play head position and the enqueued position) would gradually decrease and underflow as new buffers would be added more slowly than they were consumed. Similarly if the callback interval were any smaller than the buffer duration, the buffer level would increase until overflow occurred. Therefore, the fact that the callback intervals are distributed around a predictable mean value suggests a possible approach to achieving constant latency.

The callback intervals recorded for the MotoG are suggestive of a low level task which polls the state of the low level audio buffers approximately once every 20 ms and initiates a callback if the buffer level is less than some threshold. We assume that, for various reasons (including the mismatch between the polling rate and the buffer duration), the buffer level is different for consecutive callbacks and therefore there is variable latency between the true play head position and the next data enqueued by the callback. Our approach is to estimate the times at which callbacks would have been invoked if they were invoked at the same buffer level each time and therefore invoked at isochronous intervals.

The basic approach is to record the times at which callbacks occur and filter these to estimate the time at which a constant latency callback would have occurred.

The technique used to estimate the filtered callback time was double exponential smoothing [7] as defined by the following standard equations:

$$s(n) = \alpha x(n) + (1-\alpha)\left[s(n-1) + b(n-1)\right] \quad (3)$$

$$b(n) = \beta\left[s(n) - s(n-1)\right] + (1-\beta)b(n-1) \quad (4)$$

where $s(n)$ defines the smoothed output (the filtered callback time) at time n, $x(n)$ defines the input (the callback time), and b(n) defines the trend in the data (corresponding to the smoothed interval between callbacks in this case). The parameters α and β determine the smoothing factors and rate of convergence of the filter.

We initialize $s(0)$ to be $x(0)$ and $b(0)$ to be the native buffer duration. On each callback, (3) provides an estimate of the filtered callback time based on the linear trend up to that point. Thereafter (4) updates the value of the trend. The trend should not change much over time but using these equations allows the algorithm to adapt to slight mismatches between the system clock used by Android and the sample clock used by the audio hardware. (We previously used simple exponential smoothing [7] with a fixed estimated interval between callbacks but preliminary results showed that the two clocks soon drifted audibly apart using this approach.)

The filtered callback time is calculated at the start of each callback before any new data has been enqueued. Therefore the play head position at the start of the callback may be estimated (save for a constant offset) as

$$cbPlayPos = enqueuedPos + \left(cbTime - filteredCBTime\right) \quad (5)$$

where *enqueuedPos* is the end of data stream position after the preceding callback completed, *cbTime* is the time at which the current callback was invoked, and *filteredCBTime* is the time at which the current callback should have been invoked if it were invoked exactly when the buffer level drained to its mean level. This estimate of the play head position incorporates a constant offset (the mean buffer level) which should be subtracted to get a more accurate estimate of the true play head position. In our approach, this offset is incorporated as part of the fixed delay added later in the algorithm.

In the sound request a new estimate of the play position estimate is made as follows:

$$playPos = cbPlayPos + \left(reqTime - cbTime\right) \quad (6)$$

where *reqTime* is the time at which the request was made and other values are as already defined based on the most recent callback preceding the request. Therefore this step accounts for the time that has elapsed since the play position was last estimated. Thereafter, the insertion point may finally be determined as

$$insertPos = playPos + fixedDelay \quad (7)$$

Using the filtered callback time technique to achieve low latency on the Xperia Z2 yielded results that were essentially identical to those obtained using the OpenSL position. For that reason they are not shown here. The MotoG results, however, did differ from those obtained using the OpenSL position as shown in Figure 7.

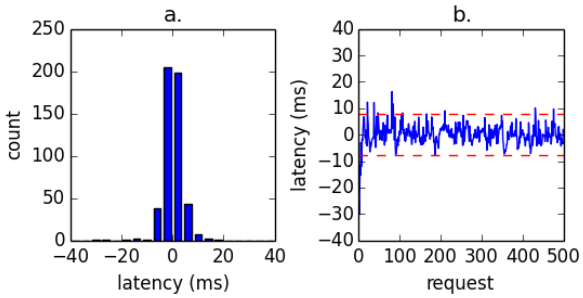


Figure 7 Relative latencies using Filtered Callback Time on the MotoG: (a) histogram, (b) values. Dashed lines show two standard deviation boundaries.

These results show a clear improvement over those obtained using the Next Buffer and OpenSL position techniques. In this case 95% of the relative latencies are within a 16 ms range resulting in jitter that should be inaudible to most, if not all, listeners. The results also indicate a number of outlier values, but these occurred at the start of the test run during the convergence period of the filter.

Despite the reduction in relative latency jitter that has been achieved there is still some residual jitter remaining. The detailed sources of this residual jitter are currently unknown but it is possible that at least some of it is due to scheduling jitter within the operating system since there are several interacting threads involved in the process of audio output.

4. CONCLUSIONS AND FUTURE WORK

The work described in this paper described the motivation for and difficulty in achieving constant latency on Android devices as many devices, even recent devices, do not implement the low latency feature now supported by the Android operating system.

We made two main contributions: we measured the relative latency jitter achieved by two Android devices using OpenSL ES and we proposed two schemes which specifically aimed to reduce this latency jitter and approach constant latency output of audio events.

The commonly used Next Buffer scheme was investigated and found to be sub-optimal in achieving constant latency for all devices (although for low latency devices it may be sufficient). Using the Open SL position to achieve constant latency worked well for one phone (the Xperia Z2) but badly for another (the MotoG) and we conclude that the success of this scheme is highly dependent on the quality of the OpenSL ES implementation on the device. Finally we proposed a novel scheme based on filtering the callback times to estimate the play head position and showed that this scheme worked quite well even when the underlying OpenSL ES implementation appeared to have several shortcomings (as was the case for the MotoG).

Notwithstanding the successes reported in this paper, preliminary investigations with additional Android devices have indicated that there seem to be quite a variety of different OpenSL ES implementations and buffer strategies used by different device manufacturers. Consequently, it appears that even the techniques described above must be supplemented by additional techniques if constant latency is to be achieved on all devices. In the future, as part of our ongoing work, we plan to investigate additional techniques and a method of selecting the best technique for a particular device (if no single technique works for all devices).

5. ACKNOWLEDGMENTS

This research was supported by BEAT-HEALTH, a collaborative project (FP7-ICT) funded by the European Union. For more information, please see http://www.euromov.eu/beathealth/.

6. REFERENCES

[1] G, Szanto and P. Vlaskovits, "Android's 10 Millisecond Problem: The Android Audio Path Latency Explainer" , Available at http:// superpowered.com/androidaudiopathlatency/ #axzz3bzkezdMg" Accessed June 3, 2015

[2] Android Open Source Project. Android Source Code. Available at https://source.android.com. Accessed April 29, 2015.

[3] Khronos Group Inc., The OpenSL ES 1.0.1 Specification. Available at https://www.khronos.org/registry/sles/specs/OpenSL_ES_Specification_1.0.1.pdf.. September 2009.

[4] "OpenSL ES for Android". Android NDK Documentation, "file://android-ndk-r10/docs/Additional%20library%20docs/opensles/index.html". Available from http://developer.android.com/ndk/downloads/index.html

[5] A. Larsen, “Ein akustischer Wechselstromerzeuger mit regulierbarer Periodenzahl für schwache Ströme”. Elektrotech. Z., ETZ 32, pp. 284–285, 1911.
[6] Android Open Source Project, "Audio Latency Measurements". Available at https://source.android.com/devices/audio/latency_measurements.html. Accessed June 11, 2015.
[7] NIST/SEMATECH e-Handbook of Statistical Methods, http://www.itl.nist.gov/div898/handbook/. Accessed June 11, 2015

GSTPEAQ – AN OPEN SOURCE IMPLEMENTATION OF THE PEAQ ALGORITHM

Martin Holters, Udo Zölzer

Department of Signal Processing and Communications,
Helmut Schmidt University
Hamburg, Germany
martin.holters|udo.zoelzer@hsu-hh.de

ABSTRACT

In 1998, the ITU published a recommendation for an algorithm for objective measurement of audio quality, aiming to predict the outcome of listening tests. Despite the age, today only one implementation of that algorithm meeting the conformance requirements exists. Additionally, two open source implementations of the basic version of the algorithm are available which, however, do not meet the conformance requirements. In this paper, yet another non-conforming open source implementation, GstPEAQ, is presented. However, it improves upon the previous ones by coming closer to conformance and being computationally more efficient. Furthermore, it implements not only the basic, but also the advanced version of the algorithm. As is also shown, despite the non-conformance, the results obtained computationally still closely resemble those of listening tests.

1. INTRODUCTION

Conducting listening tests is an expensive and time-consuming endeavor. It is therefore highly desirable to have an algorithmic alternative that estimates the outcome of a listening test at a fraction of the time and cost. For the task of judging the impairment introduced by an audio processing system that should ideally not introduce audible differences, like audio coding or transmission schemes, such an algorithm is specified in [1]. Specifically, it aims at predicting the outcome of a listening test performed according to [2], where the difference of two stimuli, the item to judge and a hidden reference, to a known reference is graded on a scale ranging from "imperceptible" (5.0) to "very annoying" (1.0). Assuming the listeners assign a better grade to the hidden reference than to the item under test, the difference between the grade for the item under test and for the hidden reference, called the Subjective Difference Grade (SDG) is negative and larger than -4. The PEAQ algorithm of [1] computes the Objective Difference Grade (ODG) which is meant to resemble the SDG obtained from listening tests. Two versions of the algorithm are specified, "basic" and "advanced", where the former is more suitable for real-time processing and the latter has higher computational demand but is supposed to yield results closer to listening tests.

As pointed out by P. Kabal [3], the standard is under-specified and in points inconsistent and the available reference data for conformance testing is insufficient to pinpoint the sources of any discrepancies. This explains why, even though the first version of the standard dates back to 1998, to this day, only one implementation meeting the conformance criteria specified therein is available. It is provided by OPTICOM[1], who were heavily involved in the development of the standard. Two other, open source implementations of the basic version of the algorithm exist, namely PQevalAudio by P. Kabal as part of the AFsp software package[2], and peaqb[3] by G. Gottardi. As will be detailed in section 4.1, neither one meets the conformance criteria of [1].

This disappointing situation motivated the authors to develop their own implementation, with the aims of

- conformance test results as close to the reference as possible
- computational efficiency
- inclusion of the advanced version
- flexible usage.

To come as close to conformance as possible, for aspects where [1] is ambiguous or [3] suggests an alternative interpretation, the authors systematically explored the possibilities and the one with the best result in terms of conformance was chosen. Efficiency was mainly achieved by exploiting the optimizations proposed in [3] and in general trying to avoid unnecessary (re-)computations wherever possible; no low-level optimizations like explicit use of SIMD instructions or the like were used, though. For flexibility in usage, the algorithm was implemented as a GStreamer[4] plug-in and the implementation therefore called GstPEAQ. GStreamer provides a framework for building graphs of processing elements for various multimedia tasks, similar to e.g. DirectShow[5], and is available for all major operating systems. Being able to include GstPEAQ in a signal processing chain allows e.g. to process the output of a codec or even measure an external device connected via an audio interface without the need of producing intermediate files. Nevertheless, for the common task of processing reference and test data stored in separate WAV files, a command line tool is provided.

2. OVERVIEW OF THE PEAQ ALGORITHM

Our aim in this section is to briefly summarize the PEAQ algorithm, while the details are left to [1]. The outline of the algorithm is depicted in Fig. 1 as a block diagram. The input signals are the reference and test signal. In case of stereo signals, the channels are processed independently in the first stages and are combined when computing the model output variables.

[1] http://www.opticom.de

[2] http://www-mmsp.ece.mcgill.ca/Documents/Downloads/AFsp/

[3] http://sourceforge.net/projects/peaqb/

[4] http://gstreamer.freedesktop.org/

[5] https://msdn.microsoft.com/en-us/library/windows/desktop/dd375454.aspx

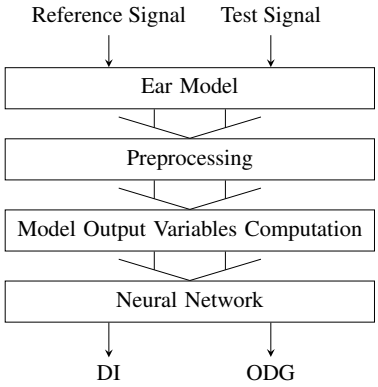


Figure 1: *Outline of the PEAQ algorithm.*

The first processing step comprises the ear model. In the basic version of the algorithm, only an FFT-based ear model is used, while in the advanced version, a filter bank-based model is used in addition. Although the details differ, both ear models decompose the input signals into auditory filter bands, apply a weighting corresponding to the outer and middle ear transfer function, add internal noise, and spread the signals both in time and frequency.

The outputs of the ear model are subject to a preprocessing step comprising level adaptation, loudness calculation and modulation processing. For the advanced version of the algorithm, only the output of the filter bank-based ear model is subject to preprocessing. The level adaptation tries to estimate and compensate level differences and linear distortions between reference and test signal. The loudness calculation determines the frame-wise loudness of the signals. The modulation processing computes a measure of the modulation of both reference and test signal.

The outputs of the ear model and the preprocessing step are then used to calculate various model output variables (MOVs). They capture different aspects of the reference and test signal and the difference between them, like the noise-to-mask ratio, linear distortions, differences in the modulation, and the harmonic structure of the error. For the basic version, a total of eleven MOVs are computed, while the advanced version employs only five MOVs.

The MOVs are then fed into a neural network with coefficients specified in [1]. It possesses one hidden layer comprising three nodes for the basic version and five nodes for the advanced version to obtain the Distortion Index (DI). The DI is finally mapped to the Objective Difference Grade (ODG) with one more node. While the ODG is intended to resemble the SDG using a five-grade impairment scale, the DI allows distinction of audio quality at the extremes where the ODG score saturates and is independent of the anchor points used to define the SDG/ODG [4].

3. GSTPEAQ IMPLEMENTATION

GstPEAQ is available under the GNU Library General Public License from `http://ant.hsu-hh.de/gstpeaq`. As mentioned, it is implemented as a plugin for the GStreamer framework, for both GStreamer version 0.10 and 1.0. In addition to the base functionality provided by GStreamer, it also uses the FFT provided by the GStreamer Base Plugins Libraries, but has no other direct library dependencies. GstPEAQ is implemented in plain C. It has been successfully used in many internal projects, e.g. [5, 6, 7].

In several aspects [1] is inconsistent or ambiguous or [3] suggests to deviate from [1] with good reason. Even though it will be solely of interest to readers with intimate knowledge of [1], in the following, the choices made for GstPEAQ are listed for the sake of completeness:

- In the frequency domain spreading of the filter bank-based ear model, the time-smoothing of the spreading slopes is performed with a and $b = 1 - a$ as given in the pseudo-code of [1], although only by swapping them one actually obtains the smoothing time constant mentioned in the textual description of [1], as observed in [3].
- For the calculation of the impact of missing components for the RmsNoiseLoudAsym MOV, the reference signal and test signal modulation patterns are swapped along with the excitation patterns as assumed in [3], although this is not explicitly mentioned in [1].
- In the calculation of the Average Distorted Block MOV, rounding towards zero as specified in [1] is used, although as remarked in [3], consistent rounding to the lower value might be more reasonable.
- A one-sided window is used in the calculation of the Error Harmonic Structure MOV, emphasizing middle frequencies, even though a window centered at DC seems to make more sense [3].
- As suggested in [3], the average is subtracted before applying the window in the calculation of the Error Harmonic Structure MOV, although this contradicts [1].
- The MOVs are not truncated to the range implied by the coefficients given for the Neural Network, as there is no mentioning of this in [1]. As the MOVs for the test items of the conformance test do not exceed this range, this setting has no impact on conformance.

With the exception of the last item, the above choices were made to minimize the RMSE obtained during conformance testing as detailed below.

4. EVALUATION

In this section, GstPEAQ version 0.6 is compared to the two other freely available, open source implementations of the PEAQ algorithm, peaqb version 1.0.beta and PQevalAudio as part of AFsp version 9r0. The comparison takes into account the conformance to [1] based on the criteria defined therein and the computational performance.

4.1. Conformance

To conform to [1], implementations are required to calculate a DI value within ± 0.02 of a given reference value for 16 selected test items. Unfortunately, neither GstPEAQ, nor peaqb or PQevalAudio fulfill this requirement. However, as can be seen in Fig. 2 for the basic version of the algorithm, all three implementations compute DI values that come reasonably close to the reference and to each other, as already observed in [7] for the ODG values. Only for three items, fcodtr2, fcodtr3, and lcodpip, the difference seems relevant. Overall, peaqb and GstPEAQ come a little closer to the reference than PQevalAudio for these 16 items, which is also

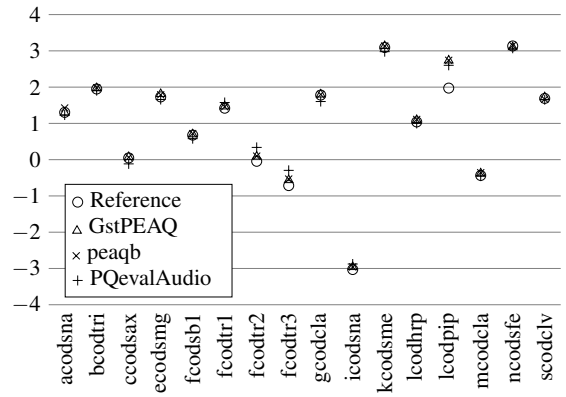


Figure 2: *Reference DI values (basic version) and DI values computed by the three implementations for the 16 test items.*

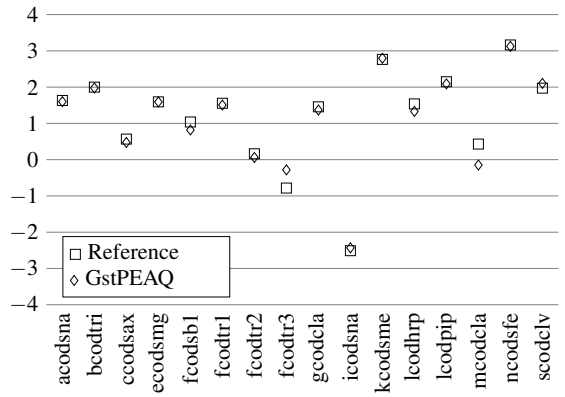


Figure 3: *Reference DI values (advanced version) and DI values computed by GstPEAQ for the 16 test items.*

reflected in the root-mean-square errors (RMSEs) between reference DI values and computed DI values listed in Table 1. However, due to the low number of test items and because the RMSEs are dominated by the few outliers, the differences between the implementations are not statistically significant.

Table 1: *Root-mean-square errors (RMSE) between reference DI values (basic version) and DI values computed by the three implementations for the 16 test items.*

Implementation	RMSE
GstPEAQ	0.2009
peaqb	0.2063
PQevalAudio	0.2329

For the advanced version of the algorithm, GstPEAQ is likewise unable to achieve conformance. The reference and computed values depicted in Fig. 3 reveal a similar trend as for the basic version, good general agreement with few exceptions, fcodtr3 and mcodcla, in this case. The resulting root-mean-square error of 0.2146 is also close to that for the basic version.

4.2. Perceptual Relevance

As none of the open source implementations, including GstPEAQ, fulfills the criteria for conformance, they should not be used for reference purposes, e.g. for stating quality metrics in a codec data sheet. This does not mean, however, that they fail to predict the outcome of listening tests. On the contrary, the relatively small differences observed in the conformance test makes one hope that the results obtained from one of the open source implementations is about as close to listening test results as those from a conforming implementation would be. To verify this, the ODG values computed by the different implementations, the reference ODG values from [1] and SDG values (read from Fig. 20 in [1]) for the same 16 test items are compared in Fig. 4.

Obviously, it is not possible to immediately judge which of the ODG data sets best fits the SDG values. What can be noted, though, is that for many items, all ODG values lie within the 95 % confidence interval of the SDG values and that the difference between the different implementations and references is smaller than the difference to the SDG value. To better quantify these findings, Table 2 gives the root-mean-square error between ODG and SDG values. Additionally, the correlation and the absolute error scores are listed, as these were criteria used in the development of the PEAQ algorithm. The absolute error score is defined in [1] as

$$AES = 2\sqrt{\frac{\sum_{i=1}^{N}((ODG_i - SDG_i)/\max(CI_i, 0.25))^2}{N}}, \quad (1)$$

where CI denotes the confidence interval and $N = 16$ the number of test items. Interestingly, the reference is outperformed in all three

Table 2: *Root-mean-square errors (RMSE), correlation, and absolute error score (AES) between SDG values and ODG values for the 16 test items.*

ODG source	RMSE	Correlation	AES
Reference (advanced)	0.2348	0.9755	1.2882
GstPEAQ (advanced)	0.1869	0.9835	1.0592
Reference (basic)	0.2713	0.9701	1.6166
GstPEAQ (basic)	0.2537	0.9711	1.5000
peaqb (basic)	0.2449	0.9728	1.4674
PQevalAudio (basic)	0.2631	0.9675	1.5786

metrics by GstPEAQ both in the basic and the advanced version and by peaqb for the basic version. This does not mean that they better predict listening test results in practice, as the differences are not statistically relevant for the same reasons as above. But it is a strong indicator that they at least should not do worse.

4.3. Performance

The performance of the three implementations was evaluated by measuring the time taken to sequentially process the 16 test items of the conformance test. All three implementations were compiled using gcc 4.9.2 with the `-O3` option. The evaluation was performed on an Intel Xeon E5-1520 CPU clocked at 3.7 GHz with 16 GiB RAM (easily enough to avoid any swapping). The achieved execution times are given in Table 3. GstPEAQ (basic version)

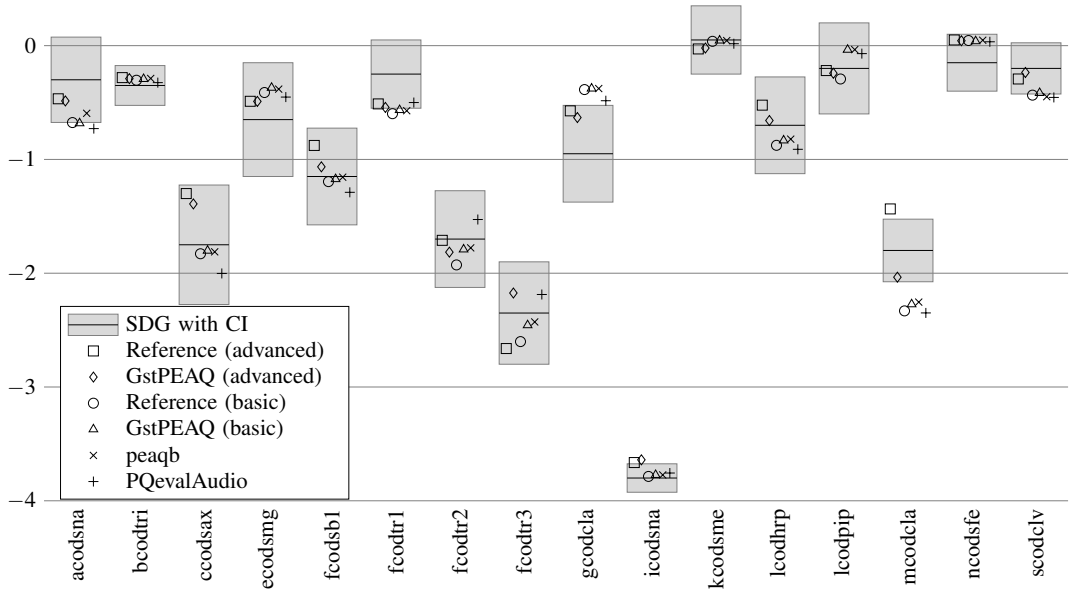


Figure 4: *SDG values with* 95 *% confidence interval (CI), reference ODG values, and ODG values computed by the three implementations for the 16 test items.*

Table 3: *Execution time.*

Implementation	Execution time
GstPEAQ (advanced)	35.4 s
GstPEAQ (basic)	6.78 s
peaqb (basic)	6660.14 s
PQevalAudio (basic)	7.04 s

and PQevalAudio show similar performance, while peaqb is substantially slower, taking about 1000 times longer. In comparison, switching GstPEAQ from basic to advanced version increases the execution time about five times.

5. CONCLUSIONS

An attempt was made to implement the PEAQ algorithm of [1]. Unfortunately, but not unexpectedly considering [3], conformance to [1] could not be achieved. However, the new implementation, GstPEAQ, comes closer to conformance than the existing implementations PQevalAudio and peaqb, while also being computationally more efficient. Furthermore, GstPEAQ is the only freely available implementation of the advanced version of the algorithm.

While lack of conformance limits the use for comparing results to others using conforming or other non-conforming implementations, GstPEAQ provides results in good accordance with the results of listening tests. Thus, it may still prove useful to judge audio quality while avoiding expensive listening tests, allowing e.g. optimization based on perceptual criteria as in [5] or even for the publication of results, provided they are marked to be computed with a non-conforming implementation.

As GstPEAQ is released as open source software under the GPL, other researchers can inspect and improve the code to better meet their demands. In fact, contributions by the community, especially if they improve conformance or performance, are very welcome.

6. REFERENCES

[1] ITU Radiocommunication Assembly, *RECOMMENDATION ITU-R BS.1387-1 – Method for objective measurements of perceived audio quality*, ITU, 2001.

[2] ITU Radiocommunication Assembly, *RECOMMENDATION ITU-R BS.1116-3 – Methods for the subjective assessment of small impairments in audio systems*, ITU, 2015.

[3] Peter Kabal, "An examination and interpretation of ITU-R BS.1387: Perceptual evaluation of audio quality," Tech. Rep., Department of Electrical & Computer Engineering, McGill University, 2003.

[4] Thilo Thiede, *Perceptual Audio Quality Assessment using a Non-Linear Filter Bank*, Dissertation, Technische Universität Berlin, 1999.

[5] Martin Holters and Udo Zölzer, "Automatic parameter optimization for a perceptual audio codec," in *Proceedings of the 34th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009, pp. 13–16.

[6] Gediminas Simkus, Martin Holters, and Udo Zölzer, "Error resilience enhancement for a robust ADPCM audio coding scheme," in *Proceedings of the 39th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Florence, Italy, 2014, pp. 3885–3689.

[7] Marco Fink and Udo Zölzer, "Low-delay error concealment with low computational overhead for audio over IP applications," in *Proceedings of the 17th International Conference on Digital Audio Effects DAFx-14*, Erlangen, Germany, 2014.

HARMONIC MIXING BASED ON ROUGHNESS AND PITCH COMMONALITY

Roman Gebhardt
Audio Information Processing,
Technische Universität München
Munich, Germany
roman.gebhardt@tum.de

*Matthew E. P. Davies**
Sound and Music Computing Group,
INESC TEC
Porto, Portugal
mdavies@inesctec.pt

Bernhard Seeber†
Audio Information Processing,
Technische Universität München
Munich, Germany
seeber@tum.de

ABSTRACT

The practice of harmonic mixing is a technique used by DJs for the beat-synchronous and harmonic alignment of two or more pieces of music. In this paper, we present a new harmonic mixing method based on psychoacoustic principles. Unlike existing commercial DJ-mixing software which determine compatible matches between songs via key estimation and harmonic relationships in the circle of fifths, our approach is built around the measurement of musical consonance at the signal level. Given two tracks, we first extract a set of partials using a sinusoidal model and average this information over sixteenth note temporal frames. Then within each frame, we measure the consonance between all combinations of dyads according to psychoacoustic models of roughness and pitch commonality. By scaling the partials of one track over $\pm$ 6 semitones (in 1/8th semitone steps), we can determine the optimal pitch-shift which maximises the consonance of the resulting mix. Results of a listening test show that the most consonant alignments generated by our method were preferred to those suggested by an existing commercial DJ-mixing system.

1. INTRODUCTION

The digital era of DJ-mixing has opened up DJing to a huge range of users, and also enabled new technical possibilities in music creation and remixing. The industry leading DJ-software tools (e.g., Native Instruments Traktor Pro 2[1], djay Pro[2] and Mixed in Key[3]) now offer users of all technical abilities the opportunity to rapidly and easily create DJ mixes out of their personal music collections, or those stored online. Central to these DJ-software tools is the ability to robustly identify tempo and beat locations, which, when combined with high quality audio time-stretching, allow for automatic "beat-matching" (i.e. temporal synchronisation) of music.

In addition to leveraging knowledge of the beat structure, these tools also extract harmonic information – typically in the form of an estimated key. Knowing the key of different pieces of music allows users to engage in so-called "harmonic mixing" where the aim is not only to align music in time, but also in key. Different pieces of music are deemed to be harmonically compatible if their keys exactly match or adhere to well-known relationships within the circle of fifths. When this information is combined with audio pitch-shifting functionality (i.e., the ability to transpose a piece of music by some number of semitones independent of its temporal structure) it provides a powerful means to "force" the harmonic alignment between two pieces of otherwise incompatible music.

While such a combination of robust music understanding and high quality music signal processing techniques is certainly effective within specific musical contexts – in particular for harmonically and temporally stable house music (and other related genres), we believe the key-based matching approach has several important limitations. Putting aside the primary issue that the key estimation itself might be error-prone, the most critical limitation is that a global property such as musical key provides no information regarding the musical composition which gives rise to that key nor how this might affect perceptual harmonic compatibility for listeners when two pieces are mixed. Similarly, music matching based on key alone provides no obvious means for ranking the compatibility between several different pieces of the same key. Likewise, assigning one key for the duration of a piece of music cannot indicate where in time the best possible mixes (or mashups) between different pieces of music might occur. Even with the ability to use pitch-shifting to transpose the musical key, it is important to consider the quantisation effect of only comparing whole semitone shifts. The failure to consider fine-scale tuning could lead to highly dissonant mistuned mixes between songs which still share the same key.

To attempt to address these limitations of key-based harmonic mixing, we propose a new approach based on the analysis of consonance. We base our approach on the well-established psychoacoustic principles of sensory consonance and harmony as defined by Ernst Terhardt [1, 2], where our goal is to discover the optimal, consonance-maximising alignment between two music excerpts. To this end, we first extract a set of frequencies and amplitudes

* MD is financed by National Funds through the FCT - Fundação para a Ciência e a Tecnologia within post-doctoral grant SFRH/BPD/88722/2012.

† BS is supported by BMBF 01 GQ 1004B (Bernstein Center for Computational Neuroscience Munich).

[1] http://www.native-instruments.com/en/products/traktor/dj-software/traktor-pro-2/

[2] http://www.algoriddim.com/djay-mac

[3] http://www.mixedinkey.com/

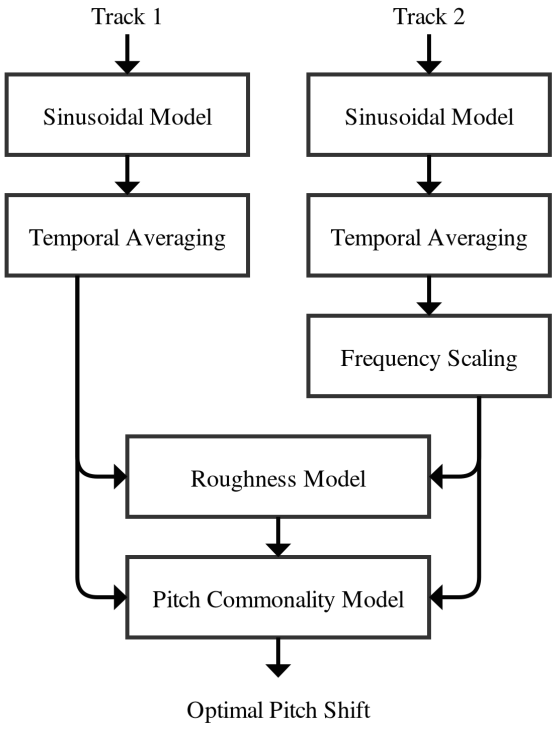


Figure 1: *An overview of the proposed approach for consonance-based mixing.*

using a sinusoidal model and average this information over short temporal frames. We fix the partials of one excerpt, and apply a logarithmic scaling to the partials of the other over a range of one full octave in 1/8th semitone steps. Through an exhaustive search we can identify the frequency shift which maximises the consonance between the two excerpts and then apply the appropriate pitch-shifting factor prior to mixing the two excerpts together. A graphical overview of our approach is given in Figure 1.

Searching across a wide frequency range in small steps allows both for a large number of possible harmonic alignments and the ability to compensate for differences in tuning. In comparison with an existing commercial DJ-mixing system, we demonstrate our approach is able to provide more consonant mixes which are also considered more pleasant by musically trained listeners.

The remainder of this paper is structured as follows. In Section 2 we review existing approaches for the measurement of consonance based on roughness and pitch commonality. In Section 3 we describe our approach for consonance-based music mixing driven by these models. We then address the evaluation of our approach in Section 4 via a listening test. Finally, in Section 5 we present conclusions and areas for future work.

2. CONSONANCE MODELS

In this section, we present the theoretical approaches for the computational estimation of consonance that will form the core of the overall implementation described in Section 3 for estimating the most consonant combination of two tracks. To avoid misunderstandings due to ambiguous terminology, we define consonance by means of Terhardt's psychoacoustic model [1, 2], which is divided into two categories: The first, *sensory consonance* combines *roughness* (and *fluctuations*, standing for slow beatings and therefore equated with roughness throughout), *sharpness* and *tonalness*. The second, *harmony* is mostly built upon Terhardt's virtual pitch theory and inherits *root relationship* and *pitch commonality*. We take these categories as the basis for our approach. To estimate the degree of sensory consonance, we use a modified version of Hutchinson & Knopoff's [3] roughness model. For calculating the pitch commonality of a combination of sonorities, we propose a model that combines Parncutt & Strasburger's [4] pitch categorisation procedure with Hofmann-Engl's [5] virtual pitch model. Both models take a sequence of sinusoids, expressed as frequencies, f_i, and amplitudes, M_i, as input.

2.1. Roughness Model

As stated above, the category of sensory consonance can be divided into three parts: roughness, tonalness and sharpness. While sharpness is closely connected to timbral properties of musical audio, we do not attempt to model or modify this aspect since it can be considered independent of the interaction of two pieces of music, which is the object of our investigation in this paper.

Parncutt & Strasburger [4] discuss the strong relationship between roughness and tonalness as a sufficient reason to only analyse one of the two properties. The fact that roughness has been more extensively explored than tonalness and that most sensory consonance models build exclusively upon it motivates the use of roughness as our sole descriptor for sensory consonance in this work. For each of the partials of a spectrum, the roughness that is evoked by the co-occurrence with other partials is computed, then weighted by the dyads' amplitudes and finally summed for every sinusoid.

The basic structure of this procedure is a modified version of Hutchinson & Knopoff's [6] roughness model for complex sonorities that builds on the roughness curve for pure tone sonorities proposed by Plomp & Levelt [7]. A function that approximates the graph estimated by Plomp & Levelt is proposed by Parncutt [8]:

$$g(y) = \begin{cases} (\exp(1)\frac{y}{0.25}\exp(-\frac{y}{0.25}))^2 & y < 1.2 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $g(y)$ is the degree of roughness of a dyad and y the frequency interval between two partials (f_i and f_j) expressed in the critical bandwidth (CBW) of the mean frequency $\bar{f}$, such that:

$$y = \frac{|f_j - f_i|}{\text{CBW}(\bar{f})} \quad (2)$$

and

$$\bar{f} = \frac{f_i + f_j}{2}. \quad (3)$$

Hutchinson & Knopoff's formula for the calculation of the critical bandwidth is often the subject of criticism (see, for example [8, 9]). Parncutt [8] states that better results can be obtained by using Moore & Glasberg's [10] equation for the equivalent rectangular bandwidth (ERB):

$$\text{ERB}(\bar{f}) = 6.23(10^{-3}\bar{f})^2 + 93.39(10^{-3}\bar{f}) + 28.52 \quad (4)$$

and hence we substitute $\text{CBW}(\bar{f})$ with $\text{ERB}(\bar{f})$ in eqn (2). The roughness values $g(y)$ for every dyad are then weighted by the

dyad's amplitudes (M_i and M_j) to obtain a value of the overall roughness D of a complex sonority with N partials:

$$D = \frac{\sum_{i=1}^{N} \sum_{j=i+1}^{N} M_i M_j g_{ij}}{\sum_{i=1}^{N} M_i^2}. \quad (5)$$

2.2. Pitch Commonality Model

As opposed to sensory consonance, which can be applied to any arbitrary sound, the second category of Terhardt's consonance model [1, 2] is largely specified on musical sounds. This is why the incorporation of an aspect based on harmony should be of critical importance in a system that aligns music according to consonance. However, the analysis of audio with a harmonic model of consonance is currently under-explored in the literature. Existing consonance-based tools for music typically focus on roughness alone [11, 12]. Relevant approaches which include harmonic analysis perform note extraction, categorisation in the octave-ranged chromagram and, as a consequence of this, key detection, but the psychoacoustic aspect of harmony is rarely applied. One of our main aims in this work is therefore to use the existing theoretical background to develop a model that estimates the consonance in terms of root relationship and pitch commonality and eventually to combine this with a roughness model.

The fundament of the approach lies in harmonic patterns in the spectrum. The extraction of these patterns is taken from the preprocessing stage of the pitch categorisation procedure of Parncutt & Strasburger's [4] tonalness model.

For a given set of partials, the audibilities of pitch categories in semitone intervals are produced. Since this corresponds directly to the notes of the chromatic scale, the degree of audibility for different pitch categories can be attributed to a chord. Hofmann-Engl's [5] virtual pitch model then will be used to compute the "Hofmann-Engl pitch sets" of these chords which will be compared for their commonality.

2.2.1. Pitch Categorisation

The first step of Parncutt & Strasburger's algorithm is the calculation of the pure-tone height, $H_p(f_i)$, for every frequency peak, f_i, in the spectrum using the analytic formula by Moore & Glasberg [10] that expresses the critical band rate in ERB:

$$H_p(f_i) = H_1 \log_e(\frac{f_i + f_1}{f_i + f_2}) + H_0. \quad (6)$$

As parameters, Moore & Glasberg propose H_1 = 11.17 erb, H_0 = 43.0 erb, f_1 = 312 Hz and f_2 = 14675 Hz. They also estimate the auditory level ΥL of each pure tone with the frequency f_i that is defined as its dB level above the threshold in quiet L_{TH}, which Terhardt [13] formulates as:

$$L_{TH} = 3.64 {f_i}^{-0.8} - 6.5 \exp\left(-0.6(f_i - 3.3)^2\right) + 10^{-3} {f_i}^4. \quad (7)$$

Then, the partial masking level $ml(f_i, f_j)$ which is the degree of how much every pure-tone in the sonority with the frequency f_i is masked by an adjacent pure-tone with its specific frequency f_j and auditory level $\Upsilon L(f_j)$ is estimated as

$$ml(f_i, f_j) = \Upsilon L(f_j) - k_m |H_p(f_j) - H_p(f_i)| \quad (8)$$

where k_m can take values between 12 and 18 dB (chosen value: 12 dB). The partial masking level is specified in dB. The overall masking level, $ML(f_i)$, of every-pure tone is obtained by adding up its partial masking levels, which are converted first to amplitudes and then, after the addition, back to dB levels:

$$ML(f_i) = \max(0, (20 \log_{10} \sum_{P \neq P'} 10^{(ml(f_i, f_j)/20)})). \quad (9)$$

In the case of a pure-tone with frequency f_i that is not masked, $ml(f_i, f_j)$ will take a large negative value. This negative value for $ML(f_i)$ is avoided by use of the the $\max$ operator when comparing the calculated value to zero.

The decision not to analyse pure-tone components in frequency, but in pitch categories is due to the need to extract harmonic patterns. The pitch categories, P, are defined by their centre frequencies in Hz:

$$P(f_i) = 12 \log_2(\frac{f_i}{440}) + 57 \quad (10)$$

where the standard pitch of 440 Hz (musical note A_4) is represented by pitch category 57.

Following this procedure for each component, we can now obtain its audible level $AL(P)$ (in dB) by subtracting its overall masking level from its auditory level $\Upsilon L(f)$:

$$AL(P) = \max(0, (\Upsilon L(P) - ML(P))). \quad (11)$$

To incorporate the saturation of each pure-tone with increasing audible level, Parncutt & Strasburger [4] estimate the audibility $A_p(P)$ for each pure-tone component:

$$A_p(P) = 1 - \exp(\frac{-AL(P)}{AL_0}). \quad (12)$$

where they follow Hesse [14] who sets AL_0 = 15.

Once every pure-tone component has been assigned to its corresponding pitch category and its audibility estimated, a template is used to detect partials of harmonic complex tones shifted over the spectrum in a step size of one semitone, i.e., one pitch category. One pattern's element is given by the formula:

$$P_n = P_1 + \lfloor 12 \log_2(n) + 0.5 \rfloor \quad (13)$$

where P_1 represents the pitch category of the lowest element (corresponding to the fundamental frequency) and P_n the pitch category of the n^{th} harmonic.

Whenever there is a match between the template and the spectrum for each semitone-shift, a complex-tone audibility $A_c(P_1)$ is assigned to the template's fundamental. To take the lower audibility of higher harmonics into account, they are weighted by their harmonic number, n:

$$A_c(P_1) = \frac{1}{k_T} \left(\sum_n \sqrt{\frac{A_p(P_n)}{n}} \right)^2. \quad (14)$$

Parncutt & Strasburger [4] set the free parameter $k_T = 3$. To estimate the audibility, $A(P)$, of a component which considers both the spectral- and complex-tone audibility of every category, the overall maximum is taken as the general audibility, as Terhardt et al. [13] state that only either a pure or a complex tone can be perceived at once:

$$A(P) = \max(A_p(P), A_c(P)). \quad (15)$$

2.2.2. Pitch-Set Commonality

The resulting set of pitch categories can be understood as a chord with each pitch category's note sounding according to its audibility. With the focus on music, we set a limit of the three notes of the sonority with the highest audibility as the triad – which is seen as the most important chord in Western culture [15]. On this basis we expect it to give a meaningful representation of the harmonic structure.

To compare two chords according to their pitch-commonality, Hofmann-Engl proposes to estimate their similarity by the aid of the pitch-sets that are produced by his virtual pitch model [16]. The obtained triad is first inserted into a table similar to the one Terhardt uses to analyse a chord for its root note (see [2]), with the exception that Hofmann-Engl's table contains one additional subharmonic. The notes are ordered from low to high along with their corresponding different subharmonics. A major difference to Terhardt's model is the introduction of two weights w_1 and w_2 to estimate the strength β_{note} for a specific note to be the root of the chord with $Q = 3$ tones for all 12 notes of an octave:

$$\beta_{\text{note}} = \frac{\sum_{q=1}^{Q} w_{1,\text{note}}\, w_{2,q}}{Q} \tag{16}$$

where the result is a set of 12 strengths of notes, or so-called "Hofmann-Engl pitches" [16]. The fusion weight, $w_{1,\text{note}}$, is based on note similarity and gives the subharmonics more impact in decreasing order. This implies that the unison and the octave have the highest weight, then the fifth, the major third and so on. The maximum value of $w_{1,note}$ is $c = 6$ Hh (Helmholtz, unit set by Hofmann-Engl). The fusion weight is decreased by the variable b, which is $b = 1$ Hh for the fifth, $b = 2$ Hh for the major third, $b = 3$ Hh for the minor seventh, $b = 4$ Hh for the major second and $b = 5$ Hh for the major seventh. All other intervals take the value $b = 6$ and are therefore weighted zero, according to the formula:

$$w_{1,note} = \frac{c^2 - b^2}{c}. \tag{17}$$

The weight according to pitch order, w_2, adds more importance to lower notes, assuming that a lower note is more likely to be perceived as the root of the chord than a higher one and is calculated as:

$$w_{2,q} = \sqrt{\frac{1}{q}} \tag{18}$$

where q represents the position of the note in the chord. For the comparison between two sonorities (e.g. from different tracks), the Pearson correlation $r_{\text{set}_1\,\text{set}_2}$ is calculated for the pair of Hofmann-Engl pitch sets, as Hofmann-Engl [16] proposes to determine chord similarity and therefore consonance, C, in the sense of harmony as:

$$C = r_{\text{set}_1\,\text{set}_2}. \tag{19}$$

3. CONSONANCE-BASED MIXING

Based on the models of roughness and pitch commonality presented in the previous section, we now describe our approach for consonance-based mixing between two pieces of music.

3.1. Data Collection and Pre-Processing

We first explain the necessary pre-processing steps which allow the subsequent measurement of consonance between two pieces of music. For the purpose of this paper, which represents our first investigation into consonance-based mixing, we make several simplifications concerning the properties of the musical audio we intend to mix.

Given that our motivation is to compare our approach to key-based matching methods in DJ-mixing software (see Section 4), we currently only consider electronic music (e.g. house music) which is both harmonically stable and typically has a fixed tempo. From a collection of recent electronic music we manually annotated the tempo and beat locations and extracted a set of musical excerpts, each lasting precisely 16 beats (i.e., 4 complete bars).

In order to focus entirely on the issue of harmonic alignment without the need to address temporal alignment, we force the tempo of each excerpt to be exactly 120 beats per minute. For this beat quantisation process, we use the open source pitch-shifting and time-stretching utility, Rubberband[4], to implement any necessary tempo changes. Accordingly, our database of musical excerpts consists of a set of 8 s (i.e., 500 ms per beat) mono .wav files sampled at 44.1 kHz.

To provide an initial set of frequencies and amplitudes, we use a sinusoidal model, namely the "Spectral Modeling Synthesis Tools" Python software package by Serra[5], with which we extract sinusoids using the default window size and hop sizes of 4096 and 256 samples respectively. In order to focus on the harmonic structure present in the musical input, we extract the partials with the highest amplitude under 5 kHz. Through informal experimentation, we set $I = 20$ partials as we found this was able to provide a sufficient harmonic representation for our consonance-based mixing application. However, we intend to explore the effect of this parameter in future work.

For our chosen genre of electronic music, we can assume that the harmonic structure remains largely constant over the duration of each 1/16th note (i.e., 125 ms). Therefore, to strike a balance between temporal resolution and computational complexity, we summarise the frequencies and amplitudes by taking the frame-wise median over the duration of each 1/16th note. Thus, for each excerpt we obtain a set of frequencies and amplitudes, $f_{\gamma,i}$ and $M_{\gamma,i}$, where i indicates the partial number (up to $I = 20$) and γ each 1/16th note frame (up to $\Gamma = 64$).

3.2. Consonance-Based Alignment

For two input musical excerpts, $T^{\mathbf{1}}$ and $T^{\mathbf{2}}$ with corresponding frequencies and amplitudes $f^{\mathbf{1}}_{\gamma,i}$,$M^{\mathbf{1}}_{\gamma,i}$ and $f^{\mathbf{2}}_{\gamma,i}$,$M^{\mathbf{2}}_{\gamma,i}$ respectively, we seek to find the optimal consonance-based alignment between them. To this end, we fix all information regarding $T^{\mathbf{1}}$ and modify $T^{\mathbf{2}}$.

Our approach centres on the calculation of consonance as a function of a frequency shift, s, and is based on the hypothesis that under some frequency shift applied to $T^{\mathbf{2}}$ the consonance between $T^{\mathbf{1}}$ and $T^{\mathbf{2}}$ will be maximised, and this, in turn, will lead to the optimal mix between the two excerpts.

In total we create $S = 97$ shifts which cover the range of $\pm$ 6 semitones in 1/8th semitone steps (i.e., 48 downward and 48 upward shifts around a single "no shift" option). We scale the

[4] https://bitbucket.org/breakfastquay/rubberband
[5] https://github.com/MTG/sms-tools

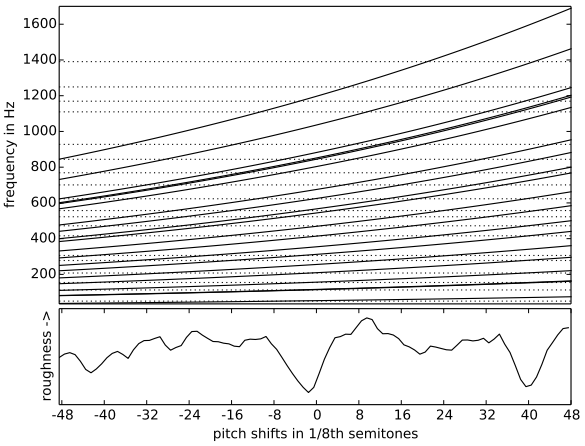


Figure 2: *(upper plot) Frequency scaling applied to the partials of one track (solid lines) compared to the fixed partials of the other (dotted lines) for a single temporal frame. (lower plot) The corresponding roughness as function of frequency scaling over that frame.*

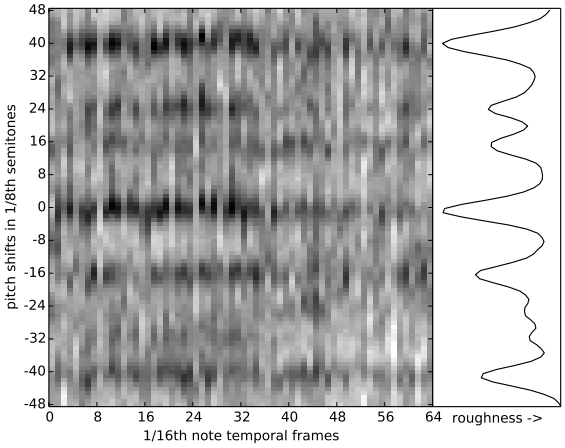


Figure 3: *Visualisation of roughness, $D_\gamma[s]$, over 64 frames for the full range of pitch-shifts. Darker regions indicate lower roughness. The subplot on the right shows the the average roughness curve, $\bar{D}[s]$ as a function of pitch-shift, where the roughness minima point to the left.*

frequencies of the partials $f^{\mathbf{2}}_{\gamma,i}$ as follows:

$$f^{\mathbf{2}}_{\gamma,i}[s] = 2^{\log_2(f^{\mathbf{2}}_{\gamma,i})+\frac{s-48}{96}} \quad s = 0,\ldots,S-1. \tag{20}$$

For each 1/16th note temporal frame, γ, and per shift, s, we then merge the corresponding frequencies and amplitudes between both tracks (as shown in Figure 2) such that:

$$f_\gamma[s] = \left[f^{\mathbf{1}}_\gamma \;\; f^{\mathbf{2}}_\gamma[s]\right] \tag{21}$$

and

$$M_\gamma[s] = \left[M^{\mathbf{1}}_\gamma \;\; M^{\mathbf{2}}_\gamma[s]\right]. \tag{22}$$

We then calculate the roughness, $D_\gamma[s]$ according to eqn (5) in Section 2.1 with the merged partials and amplitudes as input. Then, to calculate the overall roughness, $\bar{D}[s]$, as a function of frequency shift, s, we average the roughness values $D_\gamma[s]$ across the temporal frames:

$$\bar{D}[s] = \frac{1}{\Gamma}\sum_{\gamma=0}^{\Gamma-1} D_\gamma[s], \tag{23}$$

for which a graphical example is shown in Figure 3.

Having calculated the roughness across all possible frequency shifts, we now turn our focus towards the measurement of pitch commonality as described in Section 2.2. Due both to the high computational demands of the pitch commonality model, and the rounding which occurs to due to the allocation of discrete pitch categories, we do not calculate the harmonic consonance as a function of all possible frequency shifts. Instead we extract all local minima from $\bar{D}[s]$, label these frequency shifts, s^*, and then proceed with this subset. In this way we use the harmonic consonance, C, as a means to filter and rank the set of possible alignments (i.e., minima) arising from the roughness model.

While the calculation of $D_\gamma[s]$ relies on the merged set of frequencies and amplitudes from eqns (21) and (22), the harmonic consonance compares two individually calculated Hoffman-Engl pitch sets. To this end, we calculate eqns. (6) to (16) independently for $f^{\mathbf{1}}_\gamma$ and $f^{\mathbf{2}}_\gamma[s^*]$ to create set$^{\mathbf{1}}_\gamma$ and set$^{\mathbf{2}}_\gamma[s^*]$ and hence $C_\gamma[s^*]$ from eqn (19). The overall harmonic consonance $\bar{C}[s^*]$ can then be calculated by averaging across the temporal frames:

$$\bar{C}[s^*] = \frac{1}{\Gamma}\sum_{\gamma=0}^{\Gamma-1} C_\gamma[s^*]. \tag{24}$$

Since no prior method exists for combining the roughness and harmonic consonance we adopt a simple approach to equally weight their contributions to give an overall measure of consonance based on roughness and pitch commonality:

$$\rho[s^*] = \widehat{D}[s^*] + \widehat{C}[s^*] \tag{25}$$

where $\widehat{D}[s^*]$ corresponds to the raw roughness values $\bar{D}[s^*]$ which have been inverted (to reflect sensory consonance as opposed to roughness) and then normalised to the range [0,1], and $\widehat{C}[s^*]$ similarly represents the [0,1] normalised version of $\bar{C}[s^*]$. The overall consonance $\rho[s^*]$ takes values that range from 0 (minimum consonance) to 2 (maximum consonance), as shown in Figure 4. The maximum score of 2 is achieved only if the roughness and harmonic consonance detect the same pitch-shift index as most consonant.

3.3. Post-Processing

The final stage of the consonance-based mixing is to physically implement the mix between tracks $T^{\mathbf{1}}$ and $T^{\mathbf{2}}$ under the consonance-maximising pitch shift, i.e., $\arg\max_{s^*}(\rho[s^*])$. As in Section 3.1, we again use the Rubberband utility to undertake the pitch-shifting on $T^{\mathbf{2}}$. To avoid loudness differences between the two tracks prior to mixing, we normalise each audio excerpt to a reference loudness level using the Replay Gain method [17].

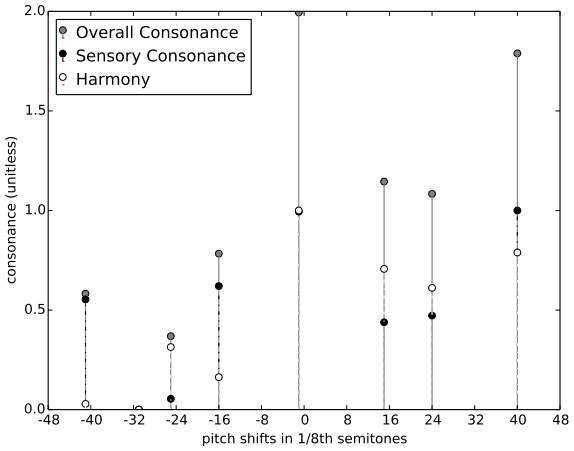


Figure 4: *Values of consonance from the sensory consonance model, $\widehat{D}[s^*]$, the harmonic consonance, $\widehat{C}[s^*]$, and the resulting overall consonance, $\rho[s^*]$. Pitch shift index -1 (i.e., -0.125 semitones) holds the highest consonance value and is the system's choice for the most consonant shift.*

4. EVALUATION

4.1. Listening Test

For the objective evaluation of our consonance-based mixing approach, we conducted a listening test. In this test we asked musically trained participants to rate short mixes created according to different outputs of our system, as well as those derived from the DJ-mixing software Traktor from Native Instruments, for their consonance and pleasantness. In total we created five conditions which are summarised as follows:

- **A** No Shift: we made no attempt to harmonically align the excerpts, instead simply aligned them in time by beat-matching.
- **B** Key Match (Traktor): we ran the key detection algorithm inside Traktor on each excerpt individually to determine the smallest pitch shift required to enable a harmonically compatible match based on the circle of fifths.[6]
- **C** Dissonant: we pitch-shifted according to the highest roughness from the roughness model without considering harmony in terms of pitch commonality.
- **D** Consonant (Sensory): we pitch-shifted according to lowest roughness from the roughness model without considering harmony.
- **E** Consonant (Sensory + Harmony): we pitch-shifted according to the result of the proposed combination of both models of roughness and pitch commonality.

Using a set of 20 excerpts (each 8 s in duration) as described in Section 3.1 we calculated the pitch-shifts required for all possible combinations between excerpts. From this complete set, we extracted a subset of 10 mixes (each made from different source excerpts) for which each of the 5 conditions yielded a unique pitch-shift. In total this gave a set of 50 musical stimuli for use in our experiment. The corresponding pitch-shifts for each mix for each of these stimuli and conditions are shown in Figure 5. Sound examples of some stimuli used in the listening test are available at the following website[7].

In total we recruited 28 participants whose musical training was determined by them being: music students, practicing musicians, or active in DJing. When listening to each mix, the participants were asked to rate two properties: first, how consonant the mixes sounded, and second they were asked to rate pleasantness of the mixes.

Both conditions were rated on a discrete six-point scale using a custom patch developed in Max/MSP. The order of the 50 stimuli was randomised for each participant. After every sound example, the ratings had to be entered before proceeding to the next example. To guarantee familiarity with the experimental procedure and stimuli, a training phase preceded the main experiment. This was also used to ensure all participants understood the concept of consonance and to set the playback volume to a comfortable level.

While the main goal was to assess the ability of our method to measure consonance, the pleasantness question was included to take into account the fact that musical consonance cannot be trivially equated with pleasantness of the sound [18], and furthermore to ensure that the definition of musical consonance was not confused with personal taste.

Regarding our hypotheses on the proposed conditions, we expected condition **C** (Dissonant) to be the least consonant, followed by **A** (No Shift). However, without any harmonic alignment, its behaviour was not predictable. Of the remaining conditions which attempted to find a good harmonic alignment, we expected the following order of consonance: **B** (Traktor) followed by **D** Consonant (Sensory) and finally our proposed combination **E** Consonant (Sensory + Harmony) the most consonant.

While the results of the sensory model have been explored in existing work [19, 11, 20], this experiment is, to the best of our knowledge, the first listener assessment of a combined roughness and harmonic model.

4.2. Results

Inspection of Figure 6, which shows the average ratings per excerpt across all conditions and criteria, reveals a wide range of ratings with some mixes considered very high in terms of consonance and pleasantness, while others were rated very low. In fact, the ratings across the two criteria of consonance and pleasantness were very strongly related with a correlation coefficient of .94. This supports our underlying assumption that a high level of consonance can be seen as a major factor for creating a good sounding mix.

By looking at the difference between different conditions in Figure 6 we can observe that in 8 of 10 cases (mixes), condition **D** (Consonant (Sensory)) - was rated more consonant than condition **A** (No Shift) and condition **C** (Dissonant). Regarding pleasantness, this was the case for every mix. The two mixes that showed the unexpected result of **D** being rated less consonant than **B** were mixes 2 and 3. Both had the lowest average ratings for consonance for all conditions (1.74, respectively 1.89, overall average 2.43). This might suggest that either one or both of the individual input

[6] `http://www.djprince.no/site/camelot_easymix_system.aspx`

[7] `http://telecom.inescporto.pt/~mdavies/dafx15/`

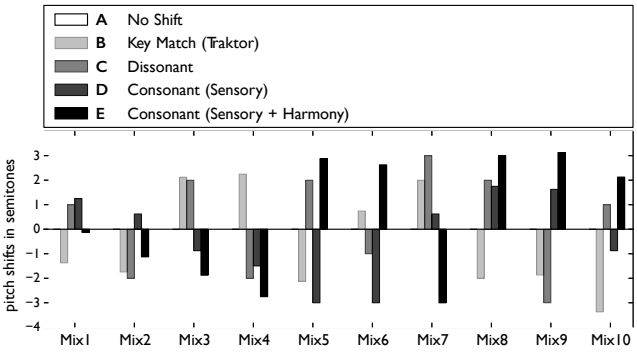


Figure 5: *Comparison of suggested pitch shifts under each condition for the listening experiment, where pitch-shifts are expressed in semitones. Note, the "No Shift" condition is always zero.*

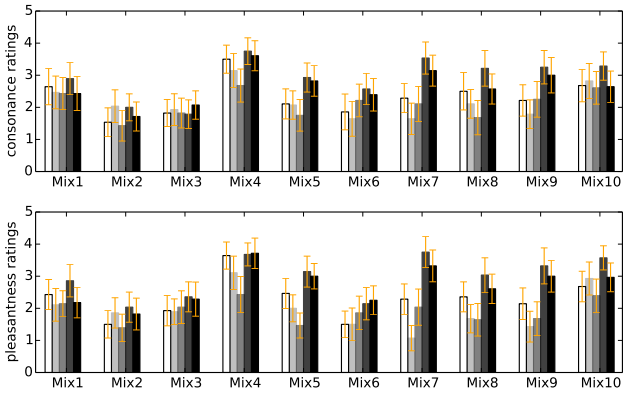


Figure 6: *Average ratings from participants of the listening experiment for consonance (upper plot) and pleasantness (lower plot). The error bars indicate the 95% confidence intervals. The shading to indicate the different conditions is as per Figure 5.*

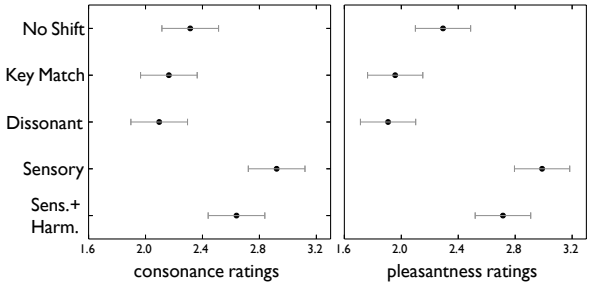


Figure 7: *Summary of multiple comparisons between conditions (with the Bonferroni correction) for consonance and pleasantness ratings. Error bars without overlap indicate statistically significant differences in the mean.*

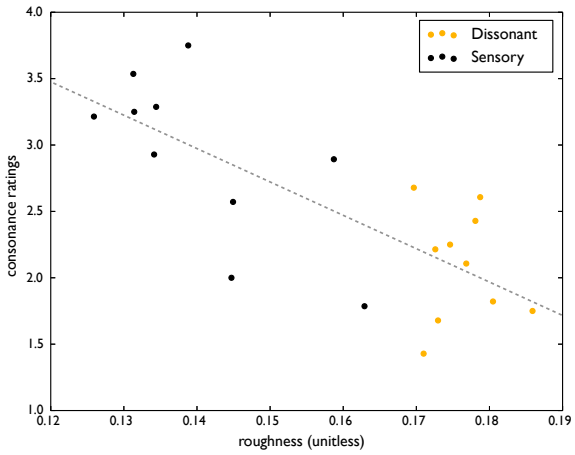


Figure 8: *Scatter plot of roughness versus consonance ratings for the stimuli of conditions* **C** *and* **D**.

tracks (prior to mixing) already contained dissonant sounds and was therefore always understood as dissonant by the participants, no matter what it was mixed with.

Comparing conditions **D** with **E** shows that the addition of the harmonic model, in general, did not improve the consonance or pleasantness ratings. In fact, the harmonic approach (**E**) was only rated more consonant once and more pleasant twice. However, it was still preferred over **A** and **C** for consonance eight times and in terms of pleasantness nine times. Therefore, our simple linear combination of roughness and pitch commonality does not seem obligatory to maximise the consonance. The inclusion of the harmonic model did appear to provide good alternative pitch-shifts, and hence expand the range of "good" harmonic alignments (see Figure 5).

Perhaps the most interesting result found in the listening test was the fact that both developed models (**D** and **E**) were rated more consonant than the mixes from condition **B** (Key Match Traktor) in 8 of 10 cases. These results were even better for pleasantness, where **D** was always preferred over **B**. These observations support the hypothesis that a consonance-based system can produce better harmonic alignments than those using the key detection method of Traktor aligned using the circle of fifths.

In addition to making direct observation of the ratings across conditions and mixes from Figure 6, we also conducted a statistical test to determine if these differences were significant. To this end, we performed a multiple comparison of means test which included the Bonferroni correction to adjust for variance between mixes. The mean ratings per condition with error bars are shown in Figure 7. For both consonance and pleasantness, condition **D** was rated significantly higher than conditions **A**, **B** and **C** ($p < .0001$ comparing **D** to **A** – the highest rated among the three), however there was no significant difference between **D** and **E**. As shown in Figure 7 condition **B** is among the lowest rated and has no significant difference even from **C** which we expected to be rated lowest. A possible explanation for this may be the failure of the key induction algorithm in Traktor to cope with such short music excerpts (each just 8 s in duration). We intend to explore this result and conduct comparisons with other key-based DJ-mixing software systems in future work.

The fact that roughness seems to have a major effect on the

rating of consonance (and hence pleasantness) motivates a closer investigation into their relationship. To this end, the roughness values of all mixes for conditions **C** and **D**, which represent the global extrema of the calculated roughness curves, were compared with their associated consonance ratings, as shown in shown in Figure 8. From the scatter plot we can observe a strong negative correlation (with coefficient of $-.75$) between the two. This relationship further supports the idea that roughness provides a meaningful perceptual scale for harmonic alignment of music signals.

5. CONCLUSIONS

In this paper we have presented a new method for harmonic mixing targeted towards addressing some of the limitations of commercial key-based DJ-mixing systems. Our approach centres on the use of psychoacoustic models of roughness and pitch commonality to identify an optimal harmonic alignment between different pieces of music across a wide range of possible pitch-shifts. Via a listening experiment with musically trained participants we were able to demonstrate that, within the context of the musical stimuli used, mixes based on roughness were considered significantly more consonant than those aligned according to musical key. Furthermore, the inclusion of the harmonic consonance model provided alternative pitch-shifts which were also significantly more pleasant than those of a commercial system.

In terms of future work, we intend to further explore how to weight the contribution of the roughness and harmonic consonance models. We also plan to extend the model to allow it to search across the temporal dimension of music to identify the most consonant temporal alignment between two musical excerpts. To this end, we will investigate more computationally efficient solutions to enable real-time interactive consonance-based music mixing, as well as experimentation with different musical genres.

6. REFERENCES

[1] E. Terhardt, "The concept of musical consonance: A link between music and psychoacoustics," *Music Perception: An Interdisciplinary Journal*, pp. 276–295, 1984.

[2] E. Terhardt, *Akustische Kommunikation (Acoustic Communication)*, Springer, Berlin, 1998, in German.

[3] W. Hutchinson and L. Knopoff, "The significance of the acoustic component of consonance of western triads," *Journal of Musicological Research*, vol. 3, pp. 5–22, 1979.

[4] R. Parncutt and H. Strasburger, "Applying psychoacoustics in composition: "harmonic" progressions of "non-harmonic" sonorities," *Perspectives of New Music*, vol. 32, no. 2, pp. 1–42, 1994.

[5] L. Hoffman-Engl, "Virtual pitch and pitch salience in contemporary composing," in *Proceedings of VI Brazilian Symposium on Computer Music*, Rio de Janeiro, Brazil, 1999.

[6] W. Hutchinson and L. Knopoff, "The acoustic component of western consonance," *Interface*, vol. 7, pp. 1–29, 1978.

[7] R. Plomp and W. J. M. Levelt, "Tonal consonance and critical bandwidth," *Journal of the Acoustical Society of America*, vol. 38, pp. 548–560, 1965.

[8] R. Parncutt, "Parncutt's implementation of Hutchinson & Knopoff (1978)," Available at http://uni-graz.at/ parncutt/rough1doc.html, accessed May 11, 2015.

[9] D. Huron, "Music 829B: Consonance and Dissonance," Available at http://www.musiccog.ohio-state.edu/Music829B/tonotopic.html, accessed May 11, 2015.

[10] B. Moore and B. Glassberg, "Suggested formulae for calculating auditory-filter bandwidths and excitation patterns," *Journal of the Acoustical Society of America*, vol. 74, pp. 750–753, 1983.

[11] J. MacCallum and A. Einbond, "Real-time analysis of sensory dissonance," in *Computer Music Modeling and Retrieval. Sense of Sounds*, R. Kronland-Martinet, S. Ystad, and K. Jensen, Eds., vol. 4969 of *Lecture Notes in Computer Science*, pp. 203–211. Springer Berlin Heidelberg, 2008.

[12] P. N. Vassilakis, "SRA: A Web-based Research Tool for Spectral and Roughness Analysis of Sound Signals," in *Proceedings of Sound and Music Computing Conference*, 2007, pp. 319–325.

[13] E. Terhardt, M. Seewan, and G. Stoll, "Algorithm for extraction of pitch and pitch salience from complex tonal signals," *Journal of the Acoustical Society of America*, vol. 71, pp. 671–678, 1982.

[14] A. Hesse, "Zur Ausgeprägtheit der Tonhöhe gedrosselter Sinustöne (Pitch Strength of Partially Masked Pure Tones)," in *Fortschritte der Akustik*, 1985, pp. 535–538, In German.

[15] W. Apel, *The Harvard Dictionary of Music*, Harvard University Press, Cambridge, 2nd edition, 1970.

[16] L. Hoffman-Engl, "Virtual pitch and the classification of chords in minor and major keys," in *Proceedings of ICMPC10*, Sapporo, Japan, 2008.

[17] D. Robinson, *Perceptual model for assessment of coded audio*, Ph.D. thesis, University of Essex, 2002.

[18] R. Parncutt, *Harmony: A psychoacoustical approach*, Springer, Berlin, 1989.

[19] G. Bernardes, M. E. P. Davies, C. Guedes, and B. Pennycook, "Considering roughness to describe and generate vertical musical structure in content-based algorithmic-assisted audio composition," in *Proceedings of ICMC/SMC*, Athens, Greece, September 2014, pp. 318–324.

[20] B. Hansen, "Modeling sensory dissonance in space: Revelations in sonic sculpture," M.S. thesis, University of California Santa Barbara, 2012.

[21] H. von Helmholtz, *Die Lehre von den Tonempfindungen als physiologische Grundlage für die Theorie der Musik (On the Sensations of Tone)*, Vieweg, Braunschweig, 1863, in German.

FLUTTER ECHOES; TIMBRE AND POSSIBLE USE AS SOUND EFFECT

Tor Halmrast,

1) Statsbygg, Oslo, Norway,
2) University of Oslo, Musicology
th@statsbygg.no

ABSTRACT

Flutter echoes are usually regarded as a defect we want to avoid by simple treatments like absorbents or angling/tilting of surfaces. Since the treatments are so simple to understand, flutter has not been investigated much. The physics of flutter echoes is, however, not simple, and flutter sounds quite interesting, so perhaps we can use it intentionally, as a *Digital Audio Effect?*

Repetitive reflections with Δt [s] between each reflections give a perceived tone with a frequency of $fo=1/\Delta t$ [Hz] and multiples of this; 2*fo*, 3*fo* etc. Such a repetive sound will be perceived as *fo*, (if higher than some 20 Hz). Often this "Repetition Tonality" is used to explain the "tonal" character of a flutter echo in common (not too big) rooms with two parallel, reflecting surfaces and all other surfaces almost totally absorbing. However, $fo=1/\Delta t$ is in the low frequency range (for a distance of 3.43 m, *fo*=100 Hz, assuming a velocity of sound of 343 m/s), but the perceived, "almost tonal" character of a flutter echo is of mid/high frequency, typically around 2 kHz.

This paper describes several methods of investigating flutter echoes. None of them give answers to all aspects of flutter, but together they show that the resulting mid/high frequency timbre of flutter in common rooms is not really a "tone", but band pass filtering of the broad banded impulsive signal. This filtering is a combination of two filtering effects: a Low Frequency Filtering (due to the increasing source distance and diffraction, which gives that the sound field is transferred from spherical to plane waves), and a High Pass Filtering due to air absorption. The sound pressure level of a plane wave is reduced only by air absorption and the absorption at the surfaces, while a spherical wave is reduced by additional 6 dB per doubling of distance. Together these two main filtering effects give the mid/high frequency "almost tonal" character of flutter, which we will call the "Flutter Band Tonality", as a distinction from the "Repetition Tonality" mentioned above. Depending on the amount of bass in the signal, its duration and especially the position of the sender/receiver with respect to the resonance peaks and nodes of the standing wave pattern of the room resonances between the surfaces (called $f_{res,o}$ etc), the "Repetition Tonality" (*fo. 2fo….*) will appear, but for most positions between the reflecting surfaces, the Flutter Band Tonality "tail" in mid/high frequencies will last longer.

In addition, there is also a third tonality that we will call the "Fresnel Diffraction Tonality". The last part of the paper shows that the combination of these "tonal" effects of the flutter echo can be simulated in Max/Msp and used as a *Digital Audio Effect* for speech and music.

1. INTRODUCTION

The tonal effect of repetitive short sound events is well known from acoustic literature and electroacoustic music; in the Karplus-Strong algorithm and in Stockhausen´s Kontakte [1] which incorporates a transform from tone to pulse/rhythm when the repetition rate gradually decreases to below some 20 Hz. In daily life, rhythmic reflections are common when an impulsive sound is "trapped" between two parallel, reflecting surfaces in a room with otherwise absorbing surfaces; Flutter Echoes. The signal must be shorter than, or at least comparable to the time for the sound to travel the path between the reflecting surfaces, for instance handclaps. (See Appendix regarding coloration of longer signals).

In Part 2 we look at some measurements of flutter echoes, the "tail" around 2 kHz and the relationship between the Repetition Tonality (*fo*) and the standing wave/room resonances between the surfaces. We will see that the latter is of importance only for some positions of sender and receiver in a room. In Part 3 we see the transition from a spherical wave to a plane wave, due to the diffraction from the edges of the surfaces in a simulation (Odeon). In part 4 we look at a calculation method from the literature, and in Part 5 and 6 we will look into calculations of diffraction using Fresnel Zones and Edge Diffraction Toolbox for MatLab, and compare this with the actual measurements. Each of the methods, by itself, does not give answers to all aspects of flutter, but together they give interesting view on what is happening.

Part 8 shows that the timbre of flutter can be used as an audio effect. In Appendix we will see that there is also a third tonality that we might call the "Fresnel Zone Tonality". This is, however, of minor importance in common rooms.

2. MEASUREMENTS OF FLUTTER ECHOES

2.1. Flutter Tonality/"Tail"

A typical measurement of a flutter echo in a foyer with absorbent ceiling and two reflecting, parallel walls is shown in fig. 1. (Taken from an Impulse Response measurement. For details and other similar measurements, see Halmrast[2]).

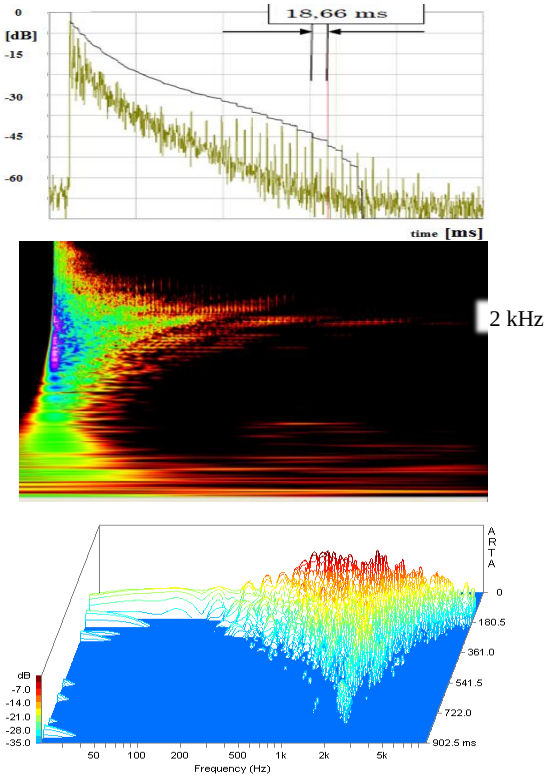


Figure 1: *Decay, Spectrogram (Wavelet) and Waterfall of a typical flutter echo*

We see that the decay ends up in a "tail" around 2 kHz. Halmrast [2] shows several similar examples of such a mid/high frequency "tail", almost like a gradual subtractive synthesis. There we also find that when the surfaces are somewhat absorbing for high frequencies, this "tail" appears at a somewhat lower frequency. The influence of such a small amount of absorption at the surfaces of and of the actual geometry will be further discussed in Part 4.

2.2. Repetition Tonality/Room Resonances//Standing Waves

Two parallel surfaces with a distance l [m] give an axial resonance at $f_{res,o}=c/2\pi l$ [Hz] and multiples of this; $2f_{res,o}$, etc. This means that the Repetition Pitch (fo) is twice the Resonance frequency, (like even partials: 2,4,6 etc of the lowest room resonance). The sound pressure distribution of the corresponding standing waves is shown in the lower part of fig. 2. The upper part of fig. 2 shows the decay at fo for the measurements with two parallel surfaces in an anechoic chamber, with constant sender position and varying receiver position (1, 2 and 3). (For details regarding the measurement, see [2]). We see that the level and decay highly depends on receiver position; slowest decay for positions close to pressure maximum, closest to the wall. Flutter is most commonly perceived when clapping at positions not very close to walls. Also, the result is reciprocal for sender/receiver, and in practice, both sender and receiver must be positioned at points of maximum sound pressure levels of $2f_{res,o}$ etc. in order for the resonances between the surfaces to be of importance. Unfortunately we could not measure exactly in the centre, but generally: Even though $2f_{res,o}$ etc. represents "standing waves", we shall see in Part 7 that the Repetition Tonality Band "stands" even longer. For signals with little energy in the bass (handclaps), the impact of room resonances/standing waves is even smaller.

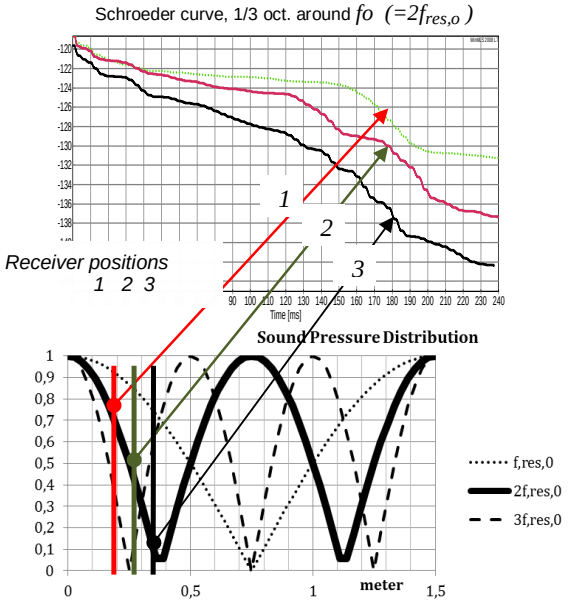


Figure2: *Schroeder curves showing the decay at different receiver positions between the surfaces Distance between surfaces:1.5m, fo=114 Hz*

For bigger rooms, fo (=2 $f_{res,o}$) will be much lower than the frequency of most common signals. Repetition Tonality might appear for $2fo$, $3fo$ etc., but their corresponding room resonances are generally weaker than for fo.

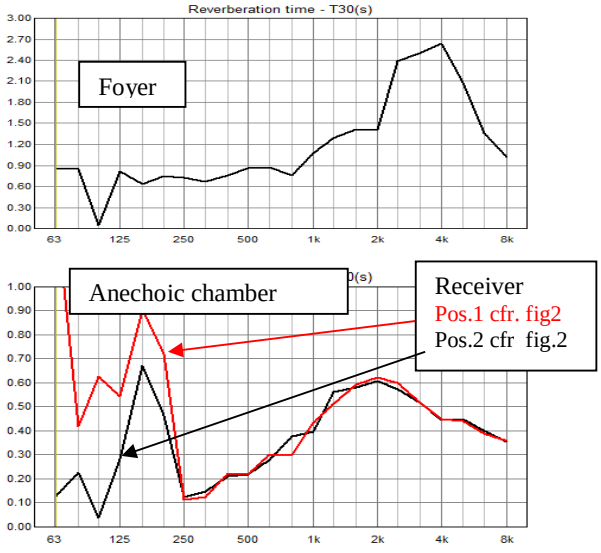


Figure 3: Reverberation Times of flutter.

The T30 curves fig. 3 are in the standard 1/3 octave. We see the characteristic peak at mid/high frequencies, but, compared to the spectrograms and waterfall curves (1/12 octave), the peak is not shown to be as sharp in T30. In fig.3, lower pane, it is interesting to notice that the reverberation time for the 1/3 oct. around *fo* changes with receiver position, which confirms the findings in fig. 2.

The set up in the anechoic chamber (see [2] and [3]) allowed for additional measurements of reverberation times (T30) for different angles between the two surfaces. In fig. 4 we see that, for constant sender/receiver positions, just a small angling of the surface give reduction in the flutter, and that the lower frequencies (*fo, 2fo*) are not influenced by such small angles, because the changes in geometry due to the angling are much smaller than the actual wavelength.

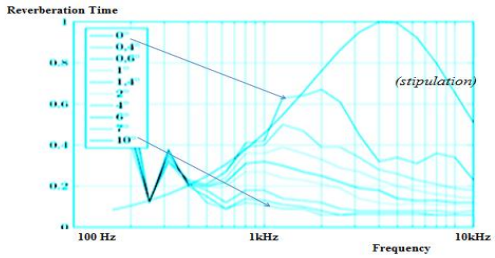


Figure 4: *Rev. time in anechoic chamber. Reduction of flutter for increasing angle between the surfaces[3]*

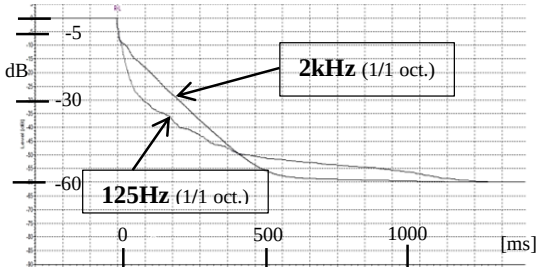


Figure 5: *Decay of sound between two surfaces.*

Fig. 5 shows that for the measurement in the anechoic room, the mid/high frequencies (2kHz) have a linear decay (which indicates a plane wave), but for the octave around *fo* (125 Hz) we see the decay of a spherical wave. We need to look further into aspects of spherical and plane waves.

3. ROOM ACOUSTICS MODEL

Spherical to plane waves

A very simple Odeon [4] room acoustics model with two parallel, reflecting surfaces was prepared (with all other surfaces totally absorbing). Figure 6 shows the radiation from a point source (spherical wave). The dimensions are as for the measurement in section 2.2. The sender is positioned almost on the centre line between the surfaces, and closer to the bottom of the surfaces, giving the possibility to inspect the situation both for a small surface (in the upper part of each figure) and a bigger surface (in the lower part of each figure).

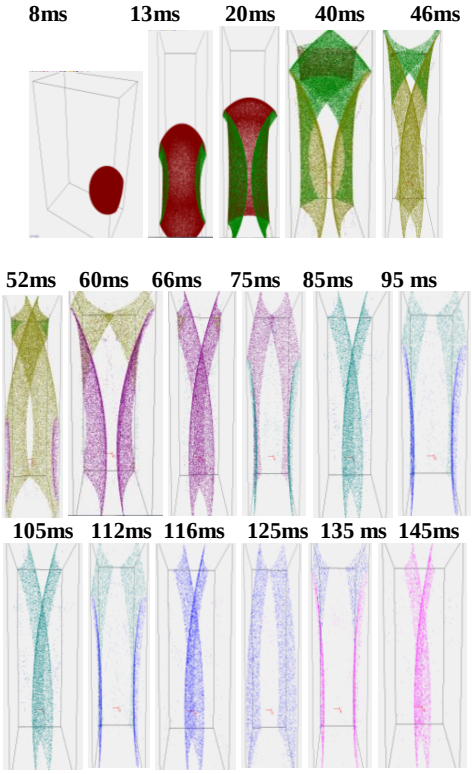


Figure 6: *Snapshots of flutter between two surfaces (Odeon simulation)*

We see that the propagation changes from a spherical to almost a plane wave after just a few reflections. It is interesting to notice that from 9th reflections and onwards to maximum for the Odeon program, we lose almost no "particles" (as they are called in Odeon) in this simulation. In general, the sound pressure level of a spherical wave is reduced by 6 dB pr. doubling of distance, while a plane wave is affected only by air absorption (and possibly of absorption at the surfaces). The reason for the transformation from spherical to plane waves is that the distance from the mirror source to the corresponding reflecting surfaces grow very fast. If we call the distance from source to surface a_1, and surface to receiver a_2 it is visualised in [1] that a_2 will remain constant, but a_1 will grow very quickly as the mirror source moves further and further away from the reflecting surface for each "flutter-reflection". For the n^{th} reflection $a_{1,n} = (2n-1)a_{1,0}$. The transition from spherical to plane is frequency dependent, as shown in fig.5. More studies on the transformation from spherical to plane waves are shown in [2].

4. INFLUENCE ON DIMENSIONS AND ABSORPTION. KUHL´S EQUATION

Flutter was investigated by Maa [5], Krait et al. [6] and Kuhl [7]. Both [6] and [7] states that for a plane wave between two surfaces of S [m2] with distance l [m], the wave is dampened only by the absorption coefficients α (in general frequency dependent, but for simplicity kept frequency independent and equal for both surfaces), and the air absorption, m (frequency dependent). Halmrast [2] includes the background for Kuhl´s

equations. The frequency content of flutter can be looked upon as the combined effect of three reverberation "asymptotes" for the reverberation time versus frequency, *f*: (*c*=velocity of sound).

1. Low Frequency damping due to finite surface areas:

$$T_1 = \frac{0.041\ x\ 2fS}{c} \quad (1)$$

2. Damping due to absorption on the surfaces:

$$T_2 = \frac{0.041xl}{\alpha} \quad (2)$$

3. Damping in the air (dissipation):

$$T_3 = \frac{0.041}{m} \quad (3)$$

The total reverberation time T_{FL} can be written as:

$$\frac{1}{T_{FL}} = \frac{1}{T_1} + \frac{1}{T_2} + \frac{1}{T_3} \quad (4)$$

Fig. 7 shows how these three "asymptotes" work together to give the total maximum reverberation for a mid/high frequency band, and how the different parameters influence on the position of the "peak" and, to a certain degree, how narrow this "tail" will be, (the "Q-factor" of the total combined filter).

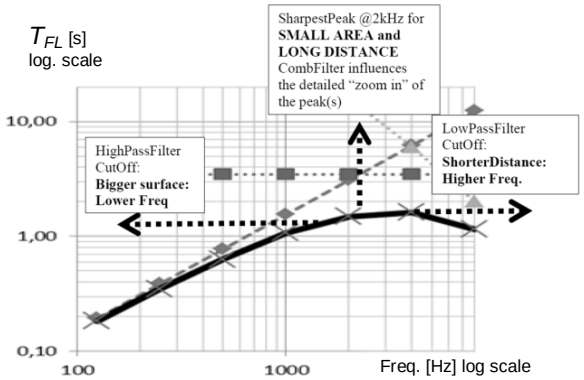


Figure 7: *Illustration of Kuhl´s equation, showing how the different parameters influence the reverberation time of flutter echoes*

For the understanding of flutter, T_1 and T_3 are the most important, and, for simplicity, the absorption coefficient α is set frequency independent and equal for both surfaces. Compared to the measurements in 2.1 (and several measurements in [2]), fig. 7 shows that Kuhl [7] gives a good explanation of what is happening, and we can see how the "tonal" characteristic of the flutter changes with different geometry and minor changes in surface absorption, but the method uses reverberation time only as a parameter, the equation for the effect of non-infinite surfaces is empirical, and the results do not give as sharp "tail"/Flutter Band Tonality as measured in actual rooms.

5. CALCULATING DIFFRACTION

5.1 Approximation of Fresnel/Kirchhoff

The behaviour of a physical reflector lies somewhere in between two extremes: Low frequency sound is not affected by a small surface (smaller than the wavelength), and if the reflector is really large, it reflects (almost) all frequencies. Between these extremes, diffraction from the edges influences the frequency response. Before we go into Fresnel Zones more in detail in the next section, we will look at an approximation of edge diffraction for two parallel surfaces. We will start with just one single surface. Fig. 8 shows a typical situation for the diffraction from the edge.

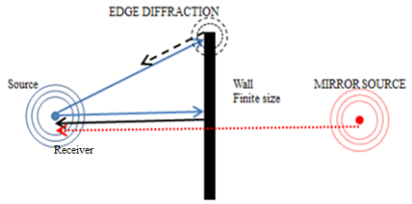


Figure 8: *Mirror source and Diffraction from the edge of a finite surface.)*

Rindel [8] has developed an approximation of Fresnel/Kirchhoff. (See also [2]). The method was developed for a single reflection and for situations where *Source-Surface* distance and *Surface-Receiver* distance are about the same size. For our investigation of flutter, we will disregard these assumptions, and investigate if this method (described in [8] and [2]) gives reasonable results also for repetitive reflections when *Source-Surface* distance quickly grows much longer as the mirror source moves longer and longer away from the reflecting surface(s) (and the wave is transformed from a spherical wave to plane wave). A typical result from such a calculation for the same small dimensions as in section 2.3 is presented in fig. 9, showing the gradual reduction in the bass as a function of the number of flutter reflections.

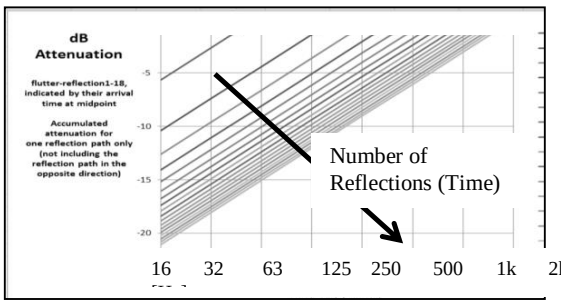


Figure 9: *Attenuation [dB] versus frequency for increasing number of flutter reflections. Fresnel/Kirchhoff approximation. (small surfaces)*

We see that this Fresnel/Kirchhoff approximation shows reasonably good agreement with the measurements for this small surface (typical dimension 1.5m). Similar tests for bigger surfaces however, give that this method does not show the large high pass filtering measured.

5.2 Fresnel zones

The peaks and dips in the frequency response due to diffraction can be investigated by looking at the Fresnel zones (see Halmrast [2]), which are shown as circles in fig. 10, left. In rooms with flutter echoes, the surface is often a rectangle, not a circular plate. Then we need to plot the rectangle and the Fresnel Zones for the given source/receiver positions and a given frequency (see fig. 10, left), and see which zone "most of edges" will fall into, in order to find if the edge diffraction will be in-phase or out-of-phase. Fig. 10 (right) shows a typical frequency response due to diffraction from the edges of a single, non-infinite surface.

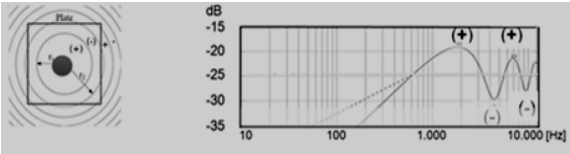


Figure 10: *Fresnel zones and their influence on frequency response [2]*

In [2] it is shown that for repetitive flutter echoes, the gradually increasing (mirror)source distance gives that the Fresnel radius gradually increases for each flutter repetition, but only up to some 10-12 reflections, as shown in fig. 11. After that, the Fresnel radii are almost constant. This is another way of showing how fast the wave is transformed from spherical to plane. In addition, the Fresnel zones actually give a "Fresnel tonality" of minor importance which is described in Appendix.

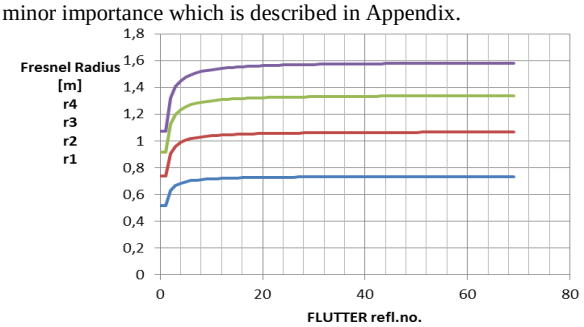
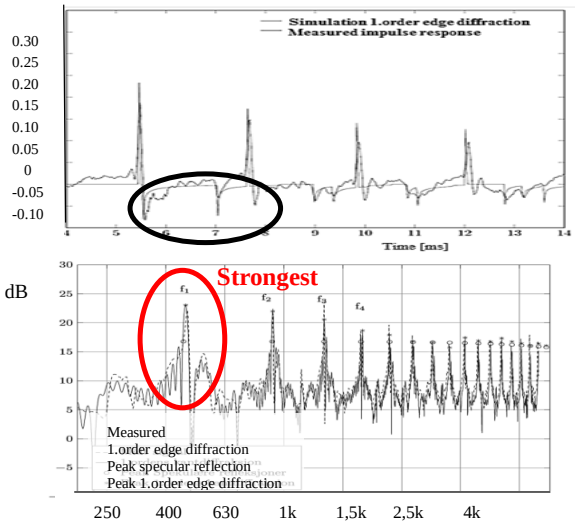


Figure 11: *Fresnel zones radii after repetitive flutter reflections*

6. CALCULATING DIFFRACTION

Flutter was investigated [3] using the EDB (Edge Diffraction toolBox) from Peter Svensson [11]. One typical comparison of measured and simulated impulse responses is shown in fig. 12. (Surfaces 1.5m x 1.5m). We see the edge diffraction (black circle).



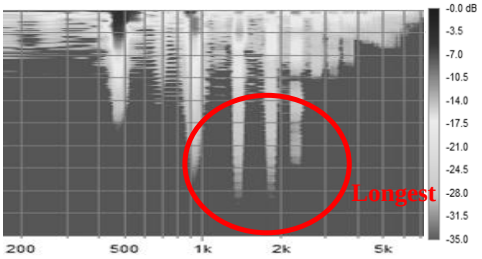


Figure 12: *Comparing simulation and measurement. Simulation: Imp.Resp. (partly) and Frequency analysis [2], Lower: Measurement, Waterfall (T.H.)*

The last two panes of fig. 12 show simulated and measured frequency response of flutter. We see that the frequency peaks align pretty well, but, because the maximum number of reflections in the simulation is only 13, the simulation does not include the last of part of the decay of these peaks. Comparing these two last panes of fig.12, we see that the peaks around 1-2 kHz are not the strongest ones in the simulation, but they last longer in the actual measurement, (red circles). (Lower part of fig.12 shows the same measurement as the waterfall in fig. 4). As a conclusion: It would have been nice to be able to simulate all aspects of flutter in MatLab, but the method available was not able to simulate longer time stretches than 13 specular reflections.

7. LINKS BETWEEN THE TWO "TONALITIES" OF FLUTTER

The waterfall curves in fig. 13 show the two main "tonalities" of a flutter echo. The lowest "hill" (marked 2, dotted ellipses) indicates the Repetition Tonality ($fo=1/\Delta t$) between the surfaces. For gradually higher frequencies we see the "harmonics" of this resonance (*2fo, 3fo* etc.). We see that the mid/high band (marked 1, solid ellipes) last longer and one of these "overtones" will of course "win" in the competition of lasting the longest. The fact that a mid/high frequency band last longer than the fundamental of the Repetition Pitch (*fo*), is therefore not a direct result of the *fo*-resonance itself, but as we only have the multiples of *fo* to choose from towards the "tail", there is of course a certain link between the two main "tonalities" of flutter. It is like a subtractive synthesis, a gradual formant shaping like in harmonic (overtone) chant.

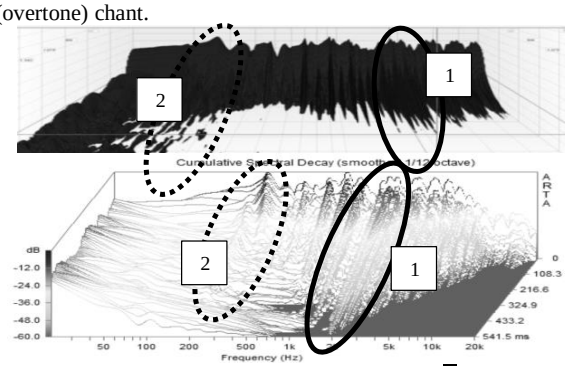


Figure 13: *Waterfall. «Repetition Tonality"* 2 *(dotted), "Flutter Band Tonality"* 1 *(solid line)*

Fig. 14 shows an overview of the two main "tonalities" of flutter. The equally spaced lines (linear frequency axis), are the

"overtones" of the "Repetition Tonality" *fo* (marked 2). The overall filtering giving the mid/high frequency "tail" is the "Flutter-Band-Tonality"(marked 1) as a result of the High Pass Filter due to non-infinite surfaces and increasing distance between mirror source and surface for each flutter reflection, and the Low Pass filtering due to air absorption. The Flutter Band Filtering is perceived easily for all positions, but the impact of the Repetition Tonality is highly dependent on positions of sender and receiver.

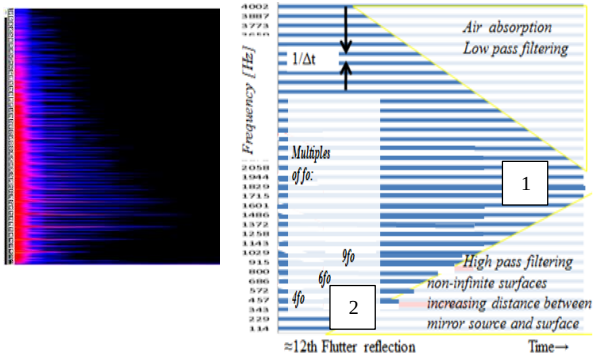


Figure 14: *The two main "tonalities" of flutter. Measurement and schematic overview*

The Appendix includes additional remarks on possible comb filter coloration due to flutter for longer signals, and a third tonality that we have called "Fresnel Zone Tonality". Both these effects are of minor importance on the overall timbre, compared to the two main "tonalities" of flutter shown in fig.14.

8. FLUTTER AS AN AUDIO EFFECT

Simplifications of the equations shown in this paper were put into a simple Max/Msp patch. To check this patch, a Dirac pulse was used as signal and the result in fig. 15 shows good agreement with the measurements shown in the earlier chapters.

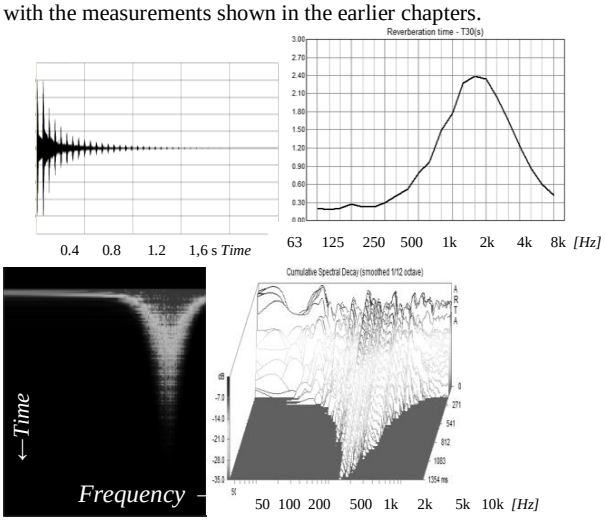


Figure 15: *Dirac pulse sent through Max/Msp patch. Impulse Response, Reverberation Time Wavelet Spectrogram and Waterfall*

Different musical signals were sent into the patch and the flutter-effect sounds somewhat "alien" or like being inside a (big, light weight) metal cylinder. For musical purposes, one could of course just convolve a signal with a recorded impulse response of a nice flutter, but the patch allows changing the parameters (moving the surfaces closer and farther away, changing the size of the surfaces and their absorption coefficients, as well as the positions of sender/receiver). For short percussive sounds and fast speech, "normal settings" in the Flutter-Patch give a rattling sound ending up around 2 kHz. For longer signals with a certain amount of bass, changing the distance between the surfaces allows a sort of "formant-resonance" change (also glissando) in the "bass", while keeping the main part of the decay almost constant as a "tail" at app. 2 kHz. Such a glissando might seem un-natural, but it is actually found in in real life in the bird- or snake-like sound heard when clapping in front of high stairs like the Chichen Itza pyramid in Mexico, where the steps of the stair are so high that the effective length between each reflection is gradually increasing, giving gradually longer time between each reflection. Such a glissando appears only for the "Repetition Tonality" of the flutter. The "Flutter-Band Tonality" of the patch will, as shown in this paper, remain (almost) constant when changing "distance between surfaces".

In the patch the "Repetition Tonality" can be shifted (un-naturally) all the way up to match the "Flutter Band Tonality" (around some 2 kHz) (and of course even further), giving a transition between the two main "tonalities" of the flutter. An extended version of the Flutter-patch includes sending the reflections successively to Left/Right (as for the physical situation when standing between two walls), and changing the distance between the ears for binaural listening as well as the distance between source and receiver. The patch can of course easily be expanded to any number of distinct loudspeakers. Some "musical" tests of the Flutter-patch with different music/speech as signal can be downloaded from: www.tor.halmrast.no

9. CONCLUSIONS

It is shown that the main "almost tonal" character of a flutter echo is not a (direct) result of the time between the reflections, but a result of two main filtering effects: The low frequencies are gradually reduced because the surfaces are finite, so that the fast growing distance from mirror source to reflecting surface gives a transformation from spherical waves to plane waves, and diffraction from the edges. The (very) high frequencies are reduced due to air absorption. It is shown that flutter actually has three "tonalities", but the combined filtering effect above is the most important.

Flutter can be used as a digital effect (in a Max/Msp patch). By changing the "geometry" in the simulation, one can gradually change the amount of filtering. The effect is different from ordinary delay/comb filters because the "tonality" is not (directly) dependent on the repetition rate, but will "always" end up in a mid/high frequency range, often around app. 2 kHz, assuming rigid, almost totally reflecting surfaces.

10. ACKNOWLEDGMENTS

Thanks to Harald Skjong (now at Norconsult, Sandvika, Norway) for permission to use parts of his M.Sc. Thesis [3].

11. REFERENCES

[1] K. Stockhausen: *Kontakte* 1958-60. http://en.wikipedia.org/wiki/Kontakte_(Stockhausen)

[2] T. Halmrast: "Why do flutter echoes ´always´ end up at 1-2 kHz?" *Proceeding of the Institute of Acoustics Vol. 37 Pt.3 2015. Available also from* www.tor.halmrast.no

[3] H. Skjong: Master Thesis, NTNU, Trondheim, Norway, 2015 (in Norwegian)

[4] Odeon Room Acoustics Software. http://www.odeon.dk/

[5] D. Y. Maa: "The flutter echoes". *J. Acoust. Soc. Amer.* 13 [1941], 170

[6] E. Krauth, R. Bücklein: "Modelluntersuchungen an Flatter-echos" *Frequenz. Zeitschrift für Schwingungs- und Schwachstromtechnik*, Band 18 Aug. 1964. Nr. 8. pp. 247-252.

[7] W. Kuhl: "Nachhallzeiten schwach gedampfter geschlossener Wellenzüge", *Acustica*, Vol. 55 1984, pp. 187-192

[8] J.H. Rindel: "Attenuation of sound reflections due to diffraction", *Nordic Acoustical Meeting*, Aalborg, Denmark Aug. 1986, pp. 257-260

[9] C.S. Clay et al.: "Specular reflections of transient pressures from finite width plane faces", *Journ. of the Acoustical Society of America*, 94(1993) Oct. No.4

[10] M. Kleiner: *Electroacoustics* CRC Press, Taylor & Francis. 2013, p 83.

[11] P. Svensson: *"Edge diffraction Matlab toolbox."* http://www.iet.ntnu.no/~svensson/software/,2013

[12] T. Halmrast: "Orchestral Timbre; Combfilter-Coloration from Reflections". *Journal of Sound and Vibration* 2000(1)352

[13] T. Halmrast: "Musical Timbre; Combfilter-Coloration from reflections", *Proceedings of the 2nd COST G-6 Workshop on Digital Audio Effects (DAFx-99)*, 1999,Trondheim, Norway

APPENDIX

COMB FILTER COLOURATION AND A THIRD TONALITY OF FLUTTER

Coloration

One or more repetitive reflections might give some kind of comb filtering; see [12] and [13]. The "tail" of the "Repetition Tonality" for flutter echoes will give a "Box-Klangfarbe" (a Comb-Between-Teeth-Bandwidth in the order of Critical Bandwidth) for most typical rooms in dwellings, for as long part of the decay as the Flutter Tonality pass band of the "tail" is broad enough to include sufficient amount of dips and peaks in the comb. For the last part of the flutter echo, the "tail" will include too so few dips and peaks that the subtractive synthesis has reached almost a pure tone, (a comb with just one or very few teeth).

Flutter Zone Tonality

A third tonality of flutter might be called the Fresnel-Zone-Tonality. This is not as easily perceived as the two main tonalities, and is highly dependent on geometry. The reflections from the edges of the surfaces form an additional rhythmic pattern which gives small extra lines in the specter. The difference in frequency between each of these lines is a function of the typical dimension of the surface (the closest Fresnel radius).

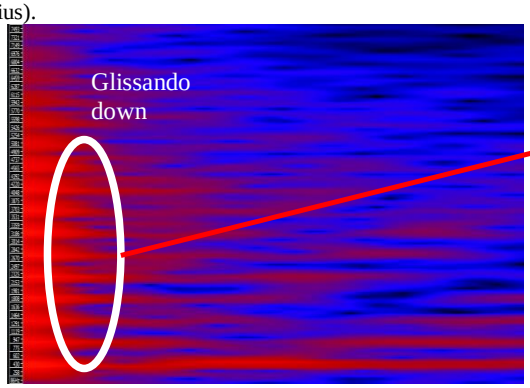


Figure App.1: *Zoom in on start (13reflections) of flutter. Glissando due to changes in Fresnel Radii in the first part of the decay*

As shown in fig. 11, the Fresnel radii increase for flutter reflections up to some 10-12. This gives that the Fresnel Zone Tonality shows small glissandi downwards for these first 10-12 flutter reflections, as the radius of the Fresnel zones increases (see fig.App.1). Fig. App.2 shows measurement showing this "Fresnel tonality" of flutter in the upper part of the figure, and in the lower part it is included in the schematic overview from fig. 14.

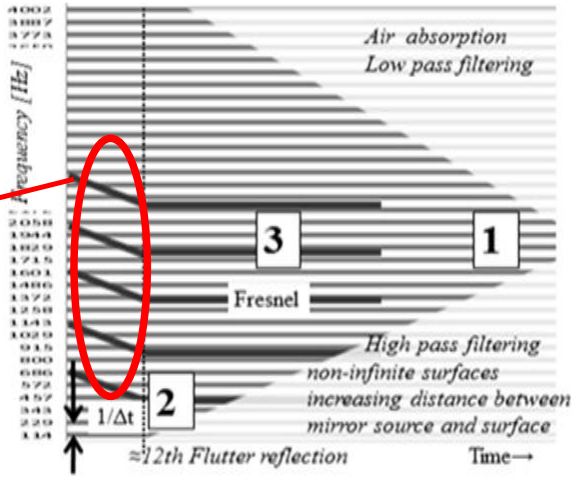


Figure App.2: *Fresnel "tonality" included in the overview from fig. 14*

EXTRACTION OF METRICAL STRUCTURE FROM MUSIC RECORDINGS

Elio Quinton, Mark Sandler
Centre for Digital Music,
Queen Mary University of London
London, UK
e.quinton@qmul.ac.uk
m.sandler@qmul.ac.uk

Christopher Harte
Melodient Limited
UK
chris@melodient.com

ABSTRACT

Rhythm is a fundamental aspect of music and metrical structure is an important rhythm-related element. Several mid-level features encoding metrical structure information have been proposed in the literature, although the explicit extraction of this information is rarely considered. In this paper, we present a method to extract the full metrical structure from music recordings without the need for any prior knowledge. The algorithm is evaluated against expert annotations of metrical structure for the GTZAN dataset, each track being annotated multiple times. Inter-annotator agreement and the resulting upper bound on algorithm performance are evaluated. The proposed system reaches 93% of this upper limit and largely outperforms the baseline method.

1. INTRODUCTION

Rhythm is a fundamental aspect of music and extraction of its properties from audio is a wide field of research. In this paper, we focus on the metrical structure of music. The model we use characterises the metrical structure by the hierarchical organisation of the underlying metrical levels and is described in more details in section 2. This model takes inspiration from music theory works such as [1], [2] or [3].

Information about metrical structure has been approached in different ways in Music Information Retrieval research. Various mid-level features such as *beat spectrum* [4], *fluctuation pattern* [5], *inter-onset histograms* [6], or *periodicity spectra* [7] have been used to perform specific tasks such as tempo estimation and beat tracking [8, 9] or classification and similarity [10, 11, 12, 13, 14, 15]. These features usually represent information about periodicities present in the audio signal, which are related, but not necessarily equivalent, to the pulse rates of the metrical levels. In other words, some information about the metrical structure is implicitly encoded in these features which is then used to perform other tasks. Information about metrical structure is not directly extracted from these features.

On the other hand, there is a small body of work aiming at specifically extracting some metrical information. For example, Gouyon proposed a method to produce a dichotomy between duple and triple meter [16]. The Echo Nest API[1] offers as "meter" assessment an integer number that specifies "how many beats are in each bar". Klapuri proposed a method to simultaneously extract three metrical levels that he describes as the "most important" ones [17]: the *tatum*, *tactus* and the *measure* levels. *Tatum* stems from 'temporal atom' and represents the shortest inter-onset interval present in the music. The *tactus* is typically defined as the rate at which listeners would tap along to the music. Tactus is also commonly associated with the *tempo* of a piece, although this view has been challenged [18]. The *measure* is defined as "[...] typically related to the harmonic change rate or to the length of a rhythmic pattern" [17]. In a similar fashion, Uhle proposed a method for estimation of tempo, "micro time" (relating the tatum period to tempo) and time signature [19]. Srinivasamurthy performed a study on the case of carnatic music [20], tracking the *sama* and *aksara* in order to characterise the tala cycle. The *aksara* is the smallest time unit of the cycle, so in that respect is analogous to the *tatum*. The *sama* is "the first aksara" of the cycle, that is to say the starting point of the cycle, which is analogous to the *measure* defined by Klapuri. Similar to the feature introduced by Peeters to perform rhythm classification in [10], Robine defines Meter Class Profiles [21] as vectors of thirteen dimensions representing the relative strength of pulses at rates related in a fixed set of integer ratios to the tempo (which is required as prior knowledge). As such, they can contain information about more than three metrical levels, but don't explicitly extract such information. Moreover, their discriminative power is only evaluated on the basis of time signature classes, thereby neglecting a part of the metrical structure. Robine notes that some information of interest is overlooked by such a reduction and this is a shortcoming we aim to tackle in this paper. At the exception of Lartillot's Matlab Toolbox [22], which we use as a baseline, none of these methods involve the direct extraction of the full metrical hierarchy.

The approach presented here aims at explicitly extracting the full metrical structure of a musical piece without requiring any prior knowledge. The structure adapts to the music and is therefore not limited in terms of number of metrical levels represented; their relationships only limited by the structural formalism described in section 2.

In order to evaluate the algorithm, we collected metrical structure annotations for the GTZAN dataset[2] from formally trained professional musicians. Each track of the dataset has been annotated by multiple annotators so that the inter-annotator disagreement and the resulting upper limit of achievable algorithm performance have been assessed [23].

In section 2 we introduce the formalism used to describe the metrical structure. The extraction algorithm is described in section 3, the evaluation using the new annotations is described in section 4 and results presented and discussed in section 5.

[1] http://developer.echonest.com/docs/v4

[2] The annotations will be made publicly available for download in case of acceptance of this paper

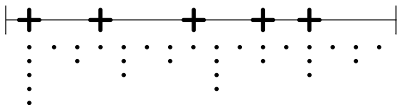


Figure 1: *A simple rumba clave rhythm pattern represented by the crosses. Each horizontal line of dots represents an underlying metrical level implied by the repetition of the pattern. Their hierarchical organisation is used to characterise the metrical structure*

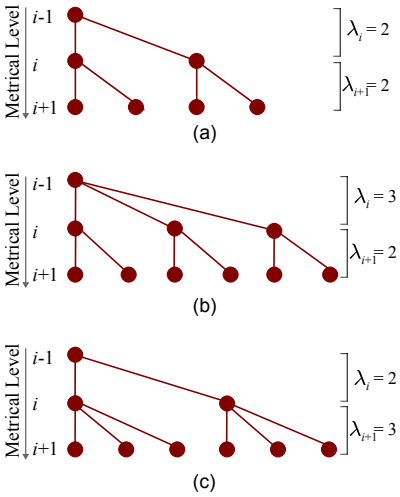


Figure 2: *Tree representation for metrical hierarchy. (a) A simple duple hierarchy dividing the lower level into two groups of two. (b) A simple triple hierarchy dividing the lower level into three groups of two. (c) A compound-duple hierarchy dividing the lower level into two groups of three.*

2. FORMALISING THE METRICAL HIERARCHY

The metrical structure of a music piece will be characterised here by the hierarchical organisation of its underlying metrical levels. Figure 1 illustrates the derivation of metrical levels from an example rhythm pattern. Naturally, the underlying metrical levels structure is dependent on the rhythm content of a musical piece. Figure 2 shows a hierarchical representation of metrical structure for several examples. Each horizontal level of nodes on the tree accounts for one metrical level (index $i \in [0, L]$), which is associated with a frequency, or rate f_i measured in BPM (Beats Per Minute). The number of metrical levels necessary to represent the rhythm hierarchy of a piece of music is therefore $L+1$. These rates can be grouped in ascending order in a vector $M = (f_0, f_1, \cdots, f_L)$. Hierarchical relationships are defined by the number of child nodes $\lambda_i \in \mathbb{N}$ each level generates. This implies that $\lambda_i = \frac{f_i}{f_{i-1}}$. A sequence of frequency ratios $\Lambda = \langle \lambda_1, \cdots, \lambda_i, \cdots, \lambda_L \rangle$ is defined. It contains only hierarchical relationships between the metrical levels and therefore can be used for tempo-independent analysis. Retrieving M from Λ only requires the provision of one absolute point of reference, that is one metrical rate. For instance $M = f_0 \star \Lambda$, where the symbol $\star$ is used to represent the fact that the frequency f_0 can be recursively multiplied by the elements λ_i of Λ so that $f_i = f_0 \cdot \prod_{k=1}^{i} \lambda_k$, with $i > 0$.

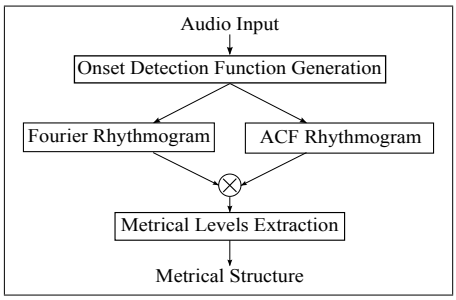


Figure 3: *The feature extraction algorithm is divided in three major steps: computing an onset detection function, performing a periodicity analysis by combining two rhythmograms and finally extracting the metrical structure from the result.*

Metrical hierarchy is related in musical stave notation terms to the time signature *and* the note values used in a composition. As an example, a musical piece using eighth notes in a $\frac{3}{4}$ time signature can be represented by Figure 2 (b), with metrical level $i-1$ being the bar level, level i the quarter note level (three quarter notes in one $\frac{3}{4}$ bar) and level $i+1$ being the eighth note level (quarter note divides into two eighth notes). Consider an example having this metrical structure and a quarter note rate of 150BPM, it would result in $M = (50, 150, 300)$ and $\Lambda = \langle 3, 2 \rangle$. If this piece had an additional layer of subdivision, such as sixteenth notes for example, it would result in $M = (50, 150, 300, 600)$, $\Lambda = \langle 3, 2, 2 \rangle$ and one more level of child nodes on a hierarchical tree representation. This example demonstrates that the information encoded by this representation of the metrical structure differs from the one encoded in a time signature notation. For instance, in the two example we just cited, both would easily be scored as $\frac{3}{4}$, but their metrical structure is different as a result of the use of an extra level of subdivision in the latter case.

3. FEATURE EXTRACTION ALGORITHM

Our extraction algorithm is performed on the audio recording of a piece of music and does not require any prior knowledge. The flowchart of the algorithm given in Figure 3 can be broken down into three processing steps. First, an onset detection function is computed from audio using the *superflux* method [24]. Then, we perform an analysis of the periodicities present in the musical signal, with the hypothesis that some of them will correspond to metrical level rates. Finally the metrical structure is estimated by peak-picking the periodicity spectrum. In this section, we describe the two latter stages.

3.1. Periodicity analysis

In order to perform the periodicity analysis, we rely on the approach introduced by Peeters [25]. Two rhythmograms are calculated in parallel using 12s Hann windows so that low periodicity rates are represented with good resolution and 0.36s hop size in order to maintain good time resolution; the first one, $\mathcal{R}_F(t, f)$ (with t representing time and f frequency), computed using a Fourier transform and the second one, $\mathcal{R}_A(t, f)$, using an autocorrelation function (ACF) with lags converted to a frequency scale. Given the dataset that will be used to carry the evaluation (cf. section 4.2), a

certain metrical consistency in the music tracks is assumed. Therefore, the Fourier transform and autocorrelation function based rhythmograms, $\mathcal{R}_F(t,f)$ and $\mathcal{R}_A(t,f)$ respectively, can be summarised in average spectra $\Omega_F(f)$ and $\Omega_A(f)$ by summing frames as given in Equation 1.

$$\begin{aligned}\Omega_F(f) &= \sum_t \mathcal{R}_F(t,f) \\ \Omega_A(f) &= \sum_t \mathcal{R}_A(t,f)\end{aligned} \tag{1}$$

These time-frequency transformations have the property to highlight the periodicities present in the signal, but also harmonics related to these periodicities. In particular, the spectrum produced using a Fourier transform of a periodic signal contains a series of higher harmonics while the ACF of the same signal would similarly contain a series of sub-harmonics. A strong hypothesis for the work presented here is that the periodicities contained in the onset detection function carry metrical structure information. However, harmonics of these periodicities are artefacts of the mathematical decomposition, which do not represent the periodicities initially present in the signal and therefore do not represent the metrical structure. A composite spectrum $\Omega_C(f)$ is produced by calculating the Hadamard product[3] of the spectra $\Omega_F(f)$ and $\Omega_A(f)$, previously resampled to a common frequency axis with 0.1 BPM resolution, and normalising the result:

$$\Omega_C(f) = \frac{\Big(\Omega_A(f) \circ \Omega_F(f)\Big)}{\max\limits_f \Big(\Omega_A(f) \circ \Omega_F(f)\Big)} \tag{2}$$

This approach aims at cancelling out the sub and higher harmonics so that only the periodicities present in the onset detection function remain in the composite spectrum $\Omega_C(f)$ because they are common to the two spectra $\Omega_F(f)$ and $\Omega_A(f)$. Figure 4 illustrates the effect of this approach on an example from the GTZAN dataset.

This track-level configuration is adopted because it suits the dataset used here. However, in a more general setting, the multiplication can be performed for every rhythmogram frame (or group of frames), and therefore capture the temporal evolution of the metrical structure.

3.2. Peak-picking algorithm

As stated earlier, our hypothesis is that metrical levels are represented by periodicities in the onset detection function, and therefore show up as peaks in the spectrum $\Omega_C(f)$. However, experience has shown that not necessarily all the peaks in $\Omega_C(f)$ are related to metrical levels. As a consequence, the metrical structure will be estimated in three steps: peak-picking $\Omega_C(f)$, generating one or more metrical structure candidates and then choosing the one that best fits the data.

First, a simple algorithm detecting local maxima if an element is larger than both of its neighbours is employed to find all the peaks in $\Omega_C(f)$. Only the peaks higher than a given threshold (0.005) are kept.

Secondly, from this list of peaks, the biggest is selected and its abscissa in $\Omega_C(f)$ is labeled f_{max} (located around 200BPM in the example of Figure 4). This represents the rate containing the most energy in the spectrum and is therefore assumed to represent a salient metrical level. The metrical structure estimation is not

[3] An element by element multiplication denoted as $\circ$

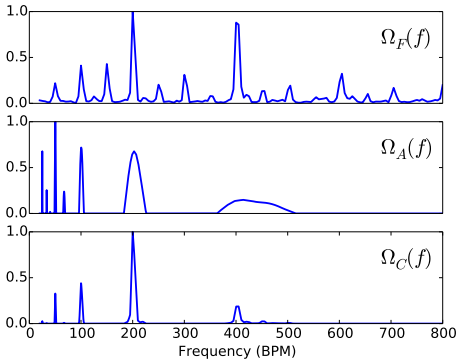


Figure 4: *Example periodicity spectra for the track blues.00053. Respectively from top to bottom, Fourier transform based, $\Omega_F(f)$, autocorrelation function based, $\Omega_A(f)$ and the result of their multiplication, $\Omega_C(f)$. Most of the harmonics in the Fourier and ACF spectra are rejected from $\Omega_C(f)$.*

sensitive to the choice f_{max} (i.e. it can equally correspond to any metrical level), however picking the most energetic rate minimises the likelihood of deriving f_{max} from a spurious peak and therefore maximises the robustness of the system in that respect. By implication the rates of all other metrical levels f_j should be related to f_{max} by integer ratios (cf. section 2). Then, the abscissa f_j of the j^{th} peak in $\Omega_C(f)$ is compared to f_{max} and is kept as a candidate level if, and only if, it satisfies one of the following conditions

$$\exists n \in \mathbb{N} : \begin{cases} \frac{f_j}{f_{\text{max}}} = n & \text{if} \quad f_j > f_{\text{max}} \\ \frac{f_{\text{max}}}{f_j} = n & \text{if} \quad f_j < f_{\text{max}} \end{cases} \tag{3}$$

otherwise it is rejected. Finding all the peaks that are integer ratios of f_{max} is not sufficient to guarantee that they form a hierarchy consistent with the model introduced in section 2, however. The rate of each metrical level and its immediate neighbour must be related by an integer ratio λ_i too.

As a consequence, the last peak-picking step is as follows: starting with f_{max}, iterative comparison of metrical level candidates f_j is performed upwards (comparison with candidates with higher rates) and downwards (comparison with candidates with lower rates). For that purpose, the procedure described by algorithm 1 is applied repeatedly to each candidate until the list of candidates is exhausted. Algorithm 1 applies for the upwards case. The downward case algorithm is easily obtained by symmetry. For each candidate f_j the algorithm considers its two nearest neighbours and appends the successful candidates to the metrical structure, rejects the others and creates additional metrical structure candidates if necessary.

Lines 1 to 3 filter out metrical level candidates not related in integer ratio to f_j. Once an integer ratio $\frac{f_q}{f_j}$ with $q > j$ is found, the second nearest neighbour f_{q+1} is taken in account. A special case occurs when $\frac{f_{q+1}}{f_j}$ is an integer ratio but $\frac{f_{q+1}}{f_q}$ is not. This means that the metrical level f_j could equally be subdivided in levels f_q or f_{q+1} whereas these two levels can't coexist in the same metrical hierarchy. In such a situation, two parallel hierarchy candidates are generated (lines 7 and 8) and constructed independently by calling two new instances of the peak-picking kernel

Algorithm 1 Peak-picking kernel: $\mathcal{K}(f_j, M)$

Require: f_j is the level under analysis and M, the metrical structure candidates
1: **while** $\frac{f_{j+1}}{f_j} \notin \mathbb{N}$ **do**
2: $f_{j+1} \leftarrow f_{j+2}$
3: **end while**
4: $f_q \leftarrow f_{j+1}$
5: **if** $\frac{f_{q+1}}{f_j} \in \mathbb{N}$ **then**
6: **if** $\frac{f_{q+1}}{f_q} \notin \mathbb{N}$ **then**
7: $M_1 \leftarrow M$
8: $M_2 \leftarrow M$
9: $f_j \leftarrow f_q$
10: $(f_j, M_1) \leftarrow \mathcal{K}(f_j, M_1)$ {call peak-picking kernel}
11: $M \leftarrow (M, M_1)$
12: $f_j \leftarrow f_{q+1}$
13: $(f_j, M_2) \leftarrow \mathcal{K}(f_j, M_2)$ {call peak-picking kernel}
14: $M \leftarrow (M, M_2)$
15: **else**
16: append f_{j+1} to M
17: $f_j \leftarrow f_{j+1}$
18: **end if**
19: **else**
20: append f_{j+1} to M
21: $f_j \leftarrow f_{j+1}$
22: **end if**
23: **return** f_j, M

(lines 9 to 14). Unless this condition is entered, f_{j+1} is appended to the metrical structure, the index of level under analysis is incremented (lines 17 and 21), and the peak-picking kernel called again.

At the end of this stage, hierarchy candidates have been generated, and are represented by their vector M. Finally, for each hierarchy candidate, each one of the metrical levels f_i is associated with a weight $w_i = \Omega_C(f_i)$ stored in $W = (w_0, w_1, \cdots, w_L)$. Each hierarchy candidate is graded by the sum of the weights of its metrical levels $\Theta = \sum_i w_i$. The hierarchy with the biggest cumulated weight Θ is considered as the most salient, and is therefore chosen as the hierarchy that best fits the data. As an example for the track *disco.00045*, for which the various periodicity spectra were given in Figure 4, the metrical hierarchy extracted is $M = (30.7, 61.5, 124.5, 245, 490.1)$ and $W = (0.05, 0.6, 1.0, 0.7, 0.9)$. Considering a quarter note at 124.5 BPM, the vector M represents a metrical structure exclusively based on duple subdivisions that would easily be scored in $\frac{4}{4}$, in which case the 490.1 BPM rate would represent sixteenth notes and the 30.7 BPM rate would represent the bar level.

3.3. Limitations

The metrical structure model used here is fit for representation of any sort of isosynchronous metrical structure. However, it does not enable representation of non-isosynchronous groupings. Consider a meter featuring a cycle of 5 units of a given metrical level grouped in threes and twos notated 3+2 (Dave Brubeck's Take Five is an example of such grouping). In our model the 3+2 grouping would not be accounted for. Nevertheless, the 5 ratio between the cycle and the metrical level used as a base for grouping fits in the model thus accounting for a meter "in five". Expanding the model to include non-isosynchronous metrical groupings representation is an avenue for future work, in which case the technical implementation might need to be adapted accordingly. Fourier transforms are probably not the best formalism to represent non-isosynchronous groupings because they decompose the signal on a basis of sine wave functions, which are intrinsically isosynchronous.

4. ALGORITHM EVALUATION

4.1. Evaluation metrics

Evaluation of the metrical structure extraction algorithm is performed on the GTZAN dataset as follows. For each track, a pairwise comparison of every level of the metrical hierarchy of the annotation (AN) and the extracted feature (EF) is performed. The metrical level rates from a vector M of size N are converted to a logarithmic scale. A binary matrix $\mathcal{M}$ of size $N_{AN} \times N_{EF}$ storing the matching information between extracted feature and annotation is built with each element $\mathcal{M}_{ij}$ defined as:

$$\mathcal{M}_{ij} = \begin{cases} 1 & \text{if } |f_i^{AN} - f_j^{EF}| < \xi \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Consequently, each match between an annotation and an extracted metrical level is associated with the value 1 while mismatches are associated with 0. A tolerance ξ is applied to account for the variability of human rating; its value set at 15% of the annotated value.

In this context, a false negative would be characterised by a row of zeros in the matrix $\mathcal{M}$ because they correspond to levels being present in the annotation but not in the extracted feature. Likewise, a false positive would be characterised by a column of zeros. The number of true positives is obtained by summing all the coefficients $\mathcal{M}_{ij}$ of the matrix. Finally, standard information retrieval system metrics are applied. For each track, Precision, Recall and F-measure are calculated, measuring the performance of the system on each track. Average values of these scores across all tracks of the dataset are then calculated.

An example of such metrics is given below. It corresponds to the evaluation of the extracted metrical structure against one annotation for the track *rock.00029*.

$$\mathcal{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

In this case, there are four true positives, i.e. four levels matching, indicated by the ones, one false negative indicated by the last row of zeros and no false positive as there is no column of zeros. It results in Precision=1.0, Recall=0.80 and F-measure=0.89.

4.2. Evaluation Dataset

We have produced expert annotations for the GTZAN dataset [26], which is composed of 1000 music excerpts of 30 seconds duration grouped in 10 genres (with 100 tracks in each group). This dataset covers a range of metrical structures, although simple duple type of meter (typically scored in $\frac{4}{4}$) accounts for a large part of the distribution. Considering the short length of the tracks, it is assumed that the metrical structure is relatively constant throughout

Table 1: *System configurations ('methods') under evaluation defined by three parameters: the periodicity spectrum used 'PS', the activation of the second peak-picking step 'PF', and the activation of the peak-picking kernel 'PPK'. Results are presented for each method as well as for the baseline method [22] as Precision, Recall, F-measure and Performance Relative to the Upper Limit (PRUL) scores.*

	PS	PF	PPK	Precision	Recall	F-measure	PRUL
Method 1	$\Omega_C(f)$	on	on	**0.83**	**0.84**	**0.82**	**93.2%**
Method 2	$\Omega_C(f)$	on	off	0.61	0.86	0.68	77.3%
Method 3	$\Omega_C(f)$	off	off	0.51	0.96	0.64	72.7%
Method 4	$\Omega_A(f)$	on	on	0.86	0.77	0.80	90.9%
Method 5	$\Omega_A(f)$	on	off	0.70	0.79	0.72	81.8%
Method 6	$\Omega_A(f)$	off	off	0.43	0.95	0.58	65.9%
Lartillot [22]	-	-	-	0.36	0.55	0.43	48.9%

the excerpts and consequently only an overall annotation at the track level was produced. This assumption proves right in the vast majority of the cases. The annotators were presented with one, randomly picked track from the dataset at a time and asked to annotate the rate (measured in BPM) of every metrical level they could hear in the music. They could achieve this either by filling in the BPM value directly or by tapping along to automatically measure this rate.

In order to provide an estimation of the reliability of the annotations, the dataset has been entirely annotated by multiple experts. Flexer showed that inter-rater disagreement results in an upper limit for the performance possibly achievable by an algorithm [23]. In our case, for every track, each pair of annotators is considered and the level of inter-annotator agreement is measured using the metrics introduced in section 4.1. Instead of comparing annotation data (AN) and an extracted feature (EF), annotations produced by one annotator are compared with annotations produced by another. The F-measure is used as a figure of merit to assess the agreement for each track, 1 meaning perfect agreement (annotators have annotated a structure that contains exactly the same metrical levels) and 0 meaning complete disagreement (nothing in common in their annotations). The average F-measure obtained across the dataset is then 0.88. This reflects a high level of inter-annotator agreement on average while setting the upper limit of average F-measure possibly achievable by an algorithm on this dataset [23]. In the following, for each track, the extracted feature is evaluated against all the annotations available ; from which are calculated the average values presented below.

4.3. Baseline method

The *mirmetre()* function from the mirtoolbox[4] [22] has been used as a baseline. The metrical structure estimation proposed in [22] comprises three steps that are very similar to the ones in the method presented in this paper. First of all, an onset detection function is processed using a spectral flux method. Secondly an analysis of the periodicities present in this onset detection curve is performed by calculating an ACF rhythmogram (labeled "autocorrelogram" in the original publication). Finally, the metrical structure is estimated from the ACF rhythmogram. In our experiment, we set the window length and hop size identical to the values used for the algorithm described in section 3. All other parameters were set to default values. The metrical structure is returned in the form of a list of metrical level pulse rates. For each metrical level rate, an average value for the entire duration of the track is used for the evaluation.

[4]version 1.6.1

5. RESULTS

5.1. Experiment

In order to assess the usefulness of the different elements of the algorithm, the evaluation is repeated several times leaving some elements out. The role of three elements is investigated in particular. Firstly, the periodicity spectrum (PS) used either $\Omega_C(f)$, which results from the multiplication of the ACF and Fourier transform-based rhythmograms (cf. section 3), or $\Omega_A(f)$ in which case no multiplication is performed and the metrical structure extraction is performed directly on $\Omega_A(f)$. This enables comparison with the baseline method. Secondly, the peak filtering step described by Equation 3 and labeled 'PF' can be turned on and off. Finally the metrical hierarchy-constrained peak-picking step involving the peak-picking kernel $\mathcal{K}$ of algorithm1 can also be turned on and off and is referred to as PPK. The system configurations under evaluation are given in Table 1 and labeled as 'methods'. Method 1 corresponds to the complete system, as presented in section 3.

5.2. Results and discussion

For all methods under evaluation, we present in Table 1 the results as average precision, recall and F-measure scores for the entire dataset. Only the metrical level rates in the range 30-800BPM are considered for evaluation. The 30BPM lower limit is chosen because periodicity spectra (in particular $\Omega_F(f)$) tend to be very noisy in the 0-25BPM range. The 800BPM limit loosely corresponds to the fastest rate playable by virtuoso musicians[5]. We also calculate the Performance relative to the upper limit (PRUL) implied by the inter-annotator disagreement established in subsection 4.2 to an F-measure of 0.88. Consequently, for each method we have PRUL $= \frac{100 \cdot x}{0.88}$ where x is the corresponding average F-measure.

Comparing the results of Method 1 and 2 clearly shows that constraining the peak-picking algorithm with a musically meaningful model for metrical structure (via the activation of step PPK) results in a substantial increase in performance (0.14 points of F-measure score). This is primarily achieved by increasing precision score at the expense of a very small decrease of recall, which means that the PPK step effectively helps picking peaks that correspond to metrical level rates with a very little rate of error. Comparison of methods 2 and 3 reveals that the peak filtering step PF only brings a small improvement, and therefore is not sufficient to

[5]Work on music perception such as [2] mention an upper threshold around 100ms (600BPM), but virtuoso playing involves metrical rates around 600BPM and slightly above. Consequently, we increased this limit to 800BPM to leave some headroom.

extract a meaningful metrical structure on its own. A similar trend emerges from comparison of methods 4, 5 and 6.

Methods 3 and 6 both have the PF and PPK steps deactivated; only the first raw peak-picking step is active (cf. section 3). The evaluation of method 3 enables an assessment of the metrical information captured by $\Omega_C(f)$, from which tempo estimation was performed in [25]. Methods 3 and 6 exhibit similar performance in terms of recall with very high scores (0.96 and 0.95 respectively), which is to be expected because all the peaks present in $\Omega_C(f)$ are still considered at this stage. It means that almost all the metrical level rates are captured as peaks in the periodicity spectra. This was a hypothesis for the design of the extraction process and is validated by the present result. In addition, method 3 scores higher than method 6 in terms of precision. Once again this result is consistent with the assumption that irrelevant peaks would be rejected by the multiplication of $\Omega_A(f)$ and $\Omega_F(f)$. However, the rather low precision (0.51) also demonstrates that $\Omega_C(f)$ does not only contain peaks relating to metrical level rates. From the higher performance reached by method 1, we can conclude that peak-picking strategy materialised by steps PF and PPK is essential to perform accurate metrical structure extraction.

Lartillot's baseline method should be compared with methods 4, 5 and 6, as they all use ACF to estimate periodicities of the onset detection function. In all cases, the baseline method is outperformed. Given that the onset detection function and periodicity estimation used in the baseline method are not largely different from the algorithm presented in this paper, the difference probably resides mostly in the metrical structure estimation steps. As a consequence, it corroborates the idea that the peak-picking of the periodicity spectra is a difficult and sensitive, yet crucial step. Lartillot's peak picking is achieved using some heuristics that are not strongly rooted in music theory whereas our constraining of the metrical structure estimation with a musicologically motivated model proves to be instrumental in achieving an optimal level of agreement with human experts. Method 1, which involves the use of all the processing stages, delivers the best overall performance and achieves the highest F-measure reaching 93.2% of the upper limit imposed by inter-rater disagreement.

6. CONCLUSIONS

We have presented a method for explicit extraction of the full metrical structure from music recordings without the need for any prior knowledge. The extraction process is constrained by a model rooted in music theory, which proves to be a critical step in achieving high performance. The algorithm is evaluated against newly produced annotations of the GTZAN dataset. Taking in account inter-annotator disagreement, we find that our system reaches 93% of maximum achievable accuracy, and largely outperforms the baseline method.

It has been shown that using metrical structure information can help improve beat tracking [27]. The method we have introduced in this paper conforms with expert human judgment and we envision that it could be useful in informing other MIR tasks such as beat-tracking, downbeat estimation and transcription. Moreover, there is evidence that the metrical structure plays an important role in perception of musical pace [2]: "Differences in surface rhythm and metrical structure do interfere with judgments of tempo [in this context meaning how fast the music feels], even if two passages have the same beat rate". The metrical structure as a feature can therefore be useful in the assessment of pace and related tasks. For instance, it could have applications such as automatic music sequencing, music database navigation, or mashup creation and complement systems such as [28].

7. ACKNOWLEDGMENTS

This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) and Omnifone Ltd.

8. REFERENCES

[1] Fred Lerdahl and Ray S. Jackendoff, *A Generative Theory of Tonal Music*, MIT Press, 1983.

[2] Justin London, *Hearing in time*, Oxford University Press, 2012.

[3] Chunyang Song, *Syncopation: Unifying Music Theory and Perception*, Ph.D. thesis, Queen Mary University of London, 2014.

[4] Jonathan Foote and Shingo Uchihashi, "The Beat Spectrum: A New Approach To Rhythm Analysis.," in *ICME*, 2001.

[5] Elias Pampalk, Andreas Rauber, and Dieter Merkl, "Content-based organization and visualization of music archives," in *Proceedings of the tenth ACM international conference on Multimedia*, 2002, pp. 570–579.

[6] Simon Dixon, Elias Pampalk, and Gerhard Widmer, "Classification of dance music by periodicity patterns.," in *ISMIR*, 2003.

[7] Andre Holzapfel and Yannis Stylianou, "Scale transform in rhythmic similarity of music," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 1, pp. 176–185, 2011.

[8] Masataka Goto, "An audio-based real-time beat tracking system for music with or without drum-sounds," *Journal of New Music Research*, vol. 30, no. 2, pp. 159–171, 2001.

[9] Simon Dixon, "Evaluation of the audio beat tracking system beatroot," *Journal of New Music Research*, vol. 36, no. 1, pp. 39–50, 2007.

[10] Geoffroy Peeters, "Rhythm Classification Using Spectral Rhythm Patterns.," in *ISMIR*, 2005, pp. 644–647.

[11] Jonathan Foote, Matthew L. Cooper, and Unjung Nam, "Audio Retrieval by Rhythmic Similarity.," in *ISMIR*, 2002.

[12] Matthias Gruhne, Christian Dittmar, and Daniel Gaertner, "Improving Rhythmic Similarity Computation by Beat Histogram Transformations.," in *ISMIR*, 2009, pp. 177–182.

[13] Maria Panteli, Niels Bogaards, and Aline Honingh, "Modeling Rhythm Simirlarity For Electronic Dance Music," in *International Society for Music Information Retrieval Conference*, 2014.

[14] Leigh Smith, *Rhythmic similarity using metrical profile matching*, Ann Arbor, MI: MPublishing, University of Michigan Library, 2010.

[15] Jouni Paulus and Anssi Klapuri, "Measuring the similarity of Rhythmic Patterns," in *ISMIR*, 2002.

[16] Fabien Gouyon and Perfecto Herrera, "Determination of the meter of musical audio signals: Seeking recurrences in beat segment descriptors," in *Audio Engineering Society Convention 114*. 2003, Audio Engineering Society.

[17] Anssi P. Klapuri, Antti J. Eronen, and Jaakko T. Astola, "Analysis of the meter of acoustic musical signals," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 1, pp. 342–355, 2006.

[18] Justin London, "Tactus $\neq$ Tempo: Some Dissociations Between Attentional Focus, Motor Behavior, and Tempo Judgment," *Empirical Musicology Review*, vol. 6, no. 1, pp. 43–55, Jan. 2011.

[19] Christian Uhle and Juergen Herre, "Estimation of tempo, micro time and time signature from percussive music," in *Proc. Int. Conference on Digital Audio Effects (DAFx)*, 2003.

[20] Ajay Srinivasamurthy and Xavier Serra, "A supervised approach to hierarchical metrical cycle tracking from audio music recordings," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. 2014, pp. 5217–5221, IEEE.

[21] Matthias Robine, Pierre Hanna, and Mathieu Lagrange, "Meter Class Profiles for Music Similarity and Retrieval.," in *ISMIR*, 2009, pp. 639–644.

[22] Olivier Lartillot, Donato Cereghetti, Kim Eliard, Wiebke J. Trost, Marc-Andre Rappaz, and Didier Grandjean, "Estimating tempo and metrical features by tracking the whole metrical hierarchy," in *Proceedings of the 3rd International Conference on Music & Emotion (ICME3), Jyvaskyla, Finland, 11th-15th June 2013. Geoff Luck & Olivier Brabant (Eds.)*. 2013, University of Jyvaskyla, Department of Music.

[23] Arthur Flexer, "On inter-rater agreement in audio music similarity," in *International Society for Music Information Retrieval Conference*, 2014.

[24] Sebastian Bock and Gerhard Widmer, "Maximum filter vibrato suppression for onset detection," in *Proc. of the 16th Int. Conf. on Digital Audio Effects (DAFx). Maynooth, Ireland (Sept 2013)*, 2013.

[25] Geoffroy Peeters, "Time variable tempo detection and beat marking," in *Proceedings of the ICMC*, 2005.

[26] George Tzanetakis and Perry Cook, "Musical genre classification of audio signals," *Speech and Audio Processing, IEEE transactions on*, vol. 10, no. 5, pp. 293–302, 2002.

[27] Norberto Degara, Antonio Pena, Matthew EP Davies, and Mark D. Plumbley, "Note onset detection using rhythmic structure," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. 2010, pp. 5526–5529, IEEE.

[28] Matthew EP Davies, Philippe Hamel, Kazutomo Yoshii, and Misako Goto, "AutoMashUpper: automatic creation of multi-song music mashups," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 22, no. 12, pp. 1726–1737, 2014.

A SET OF AUDIO FEATURES FOR THE MORPHOLOGICAL DESCRIPTION OF VOCAL IMITATIONS

Enrico Marchetto
UMR STMS IRCAM-CNRS-UPMC
Paris, France
enrico.marchetto@ircam.fr

Geoffroy Peeters
UMR STMS IRCAM-CNRS-UPMC
Paris, France
geoffroy.peeters@ircam.fr

ABSTRACT

In our current project, vocal signal has to be used to drive sound synthesis. In order to study the mapping between voice and synthesis parameters, the inverse problem is first studied. A set of reference synthesizer sounds have been created and each sound has been imitated by a large number of people. Each reference synthesizer sound belongs to one of the six following morphological categories: "up", "down", "up/down", "impulse", "repetition", "stable". The goal of this paper is to study the automatic estimation of these morphological categories from the vocal imitations. We propose three approaches for this. A base-line system is first introduced. It uses standard audio descriptors as inputs for a continuous Hidden Markov Model (HMM) and provides an accuracy of 55.1%. To improve this, we propose a set of slope descriptors which, converted into symbols, are used as input for a discrete HMM. This system reaches 70.8% accuracy. The recognition performance has been further increased by developing specific compact audio descriptors that directly highlight the morphological aspects of sounds instead of relying on HMM. This system allows reaching the highest accuracy: 83.6%.

1. INTRODUCTION

1.1. Using vocal imitations as sketches

In typical design approaches (whether in architecture, products, etc.), the very first step is often a "sketch", that is a simple graphical representation of the target. This initial sketch is a useful tool to enhance communications between designers and stakeholders. In the case of sound design, professionals often use vocal imitations to add more detail to the sound description [1], trying to transmit to their interlocutor the main cues of their sound idea [2]. "Vocal imitations" can therefore be considered as the sound design "sketches".

The goal of our current project, the SkAT-VG project, is to expand this vocal imitation idea toward a better sound design tool [3, 4]. The resulting device should be able to translate a vocal (and gestural) cue into a novel and pertinent synthetic sound. The interactive sound design begins with a phase in which the user produces an imitation and the system provides some draft sounds; in a second phase the latter are then interactively refined, again using voice and gestures. This paper addresses the task of automatic recognition of imitation, presenting different strategies to do that.

1.2. The recognition task and imitation dataset

In order to study the mapping between voice and synthesis parameters, the inverse problem is first studied. A set of reference synthesizer sounds (*stimuli*) have been created and each sound has been imitated by a large number of people. Each reference synthesizer sound belongs to one of the six following morphological categories: `up`, `down`, `up/down`, `stable`, `impulse`, `repetition`. Each of the categories is represented by two reference sounds, and each reference sound is imitated by 50 subjects.

The six categories are abstract and are defined as:

Up: Sounds which have an increasing profile in terms of spectral content and/or loudness, thus expressing a kind of rising;

Down: Opposite of the previous one, these stimuli present a downward profile;

Up/down: Sounds with non-monotonic profiles: can be described as combination of the previous two, the stimuli profile moves upward and then downward;

Impulse: This class contains sounds with very short duration and sharp attach and decay;

Repetition: Sounds which are composed by the repetition, with varied rhythmic patterns, of short and almost impulsive ones;

Stable: Longer sounds, with almost flat pitch and loudness profiles.

The goal of this paper is to study the automatic estimation of these six morphological categories from the vocal imitations of their reference sounds.

The study of the perception of vocal imitations (how do people choose their strategy to imitate a sound, how consistent are the imitations and how these imitations are recognized) have already been the subject of the paper [5] and will be the subject of further papers in the framework of the SkAT-VG project.

1.3. Related works

A well-known approach to allow time series recognition is the extraction of low level signal descriptors, which are then modeled using Hidden Markov Models (HMM) [6, 7]. This approach will be used to create our base-line system. In speech recognition [8, 9, 10], the best results are obtained combing language models (based on grammars) and acoustical models [11]. Unfortunately, abstract sounds are not bound to any grammar and a language model cannot be used in our case.

Another closely related topic is the recognition of "words for sounds", such as onomatopoeias. Proposed approaches to this problem, linked to speech recognition, still rely on phonemes [12] or lexical cues [13]. There are also examples of features clustering and modeling [14], which are related to our base-line system and to the first methodology that we propose. Description of sounds in

terms of morphological profiles has been initially proposed by P. Schaeffer works [15]. The automatic estimation of these profiles for abstract sounds has been previously studied by [16] and [17] which also propose dedicated descriptors.

2. AUTOMATIC RECOGNITION OF VOCAL IMITATIONS

In this section, we propose three methods to automatically recognize the six morphological categories indicated in sec. 1.2.

- The first method relies on the extraction of a set of instantaneous audio features $d_i(k)$, $i \in [1, ..., I]$ over time k. Each category is modeled by its own hidden Markov model.
- The second method uses the same instantaneous audio features $d_i(k)$, which are quantified into symbols to be used as input for discrete hidden Markov models.
- The third method does not rely at all on hidden Markov models, but models the time evolution directly in the audio features. We therefore denote them by "morphological audio descriptors" as in Peeters et al., 2010 [17].

Before applying one of these methods, we first detect the non-silent regions (named "active regions" in the following) using standard methods such as [18, 19, 20]. This leads to a set of N' Active Region(s) $A = \{[b_r, f_r] : r \in [1, ..., N']\}$ where b_r and f_r are their starting and ending time.

2.1. Base-line system using Local Trend descriptors

We extract 6 instantaneous audio features $d_i(k)$, $i \in [1, ..., 6]$ where k denotes the time frame number. In order to smooth the variation of $d_i(k)$ over time, a low-pass filter is applied (zero-phase filter). The first 4 are standard audio features: the spectral centroid, spectral spread, sprectral rolloff and the pitch. They are computed using standard techniques [21] and using the Swipep algorithm [22] for the pitch[1].

Given that one of the important specificities of the morphological categories relates to the temporal evolution of the spectrum content, we also propose two new audio features: "LPC-min" and "Spectral-peak-min". The novel features are defined as follows:

LPC-min: A one-pole preemphasis filter is applied. Low-order LPC is used to estimate the position of the single most prominent formant[2]. The prediction coefficients are converted into formant frequencies F_ρ, where ρ is the formant index [23, 24]. Only the frequencies $F_\rho > 20$Hz are kept. The LPC-min value is measured in Hz, and is defined as the minimum F_ρ.

Spectral-peak-min: From the energy spectrum (computed as the square DFT) we select the 5 most important frequency bins. The Spectral-peak-min is defined as the lowest frequency among these 5 frequencies; it is thus measured in Hz.

[1]We used the Swipep algorithm since it has state-of-the-art performances and is readily available on-line. Spectral centroid, spread and rolloff are computed using well-known formulas.

[2]Here the objective is not a full-fledged formants tracking, but a robust analysis of the energy location among frequencies, complementary to spectral centroid.

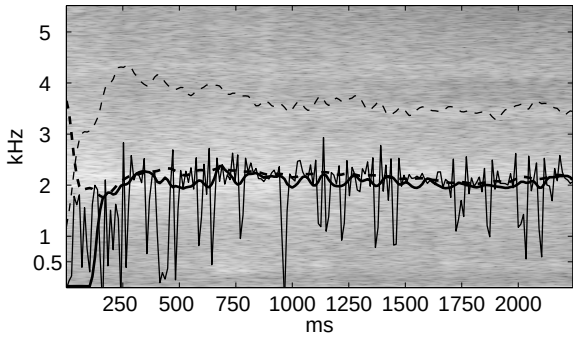


Figure 1: *Comparison between LPC-min (dashed bold line), Spectral-peak-min (bold line), spectral centroid (dashed line) and pitch by Swipep (thin line). This vocal imitation is noisy with a stable formant around 2kHz. Swipep does not detect any pitch, giving unreliable information, and the spectral centroid is moved toward higher frequencies. Both LPC-min and Spectral-peak-min are instead detecting and following the formant.*

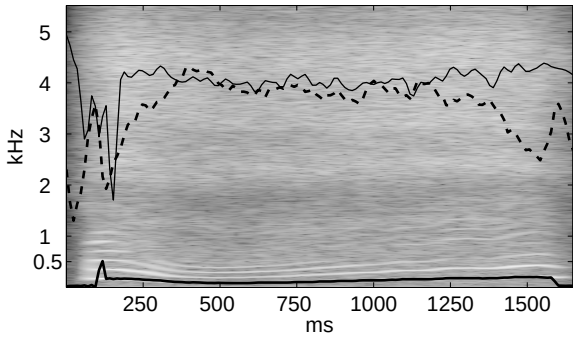


Figure 2: *Comparison between LPC-min (bold line), spectral centroid (dashed bold line) and Spectral-peak-min (thin line); pitch is not reported because perfectly matches LPC-min. This vocal imitation is harmonic but presents also noise in higher frequencies. LPC-min is clearly detecting the pitch; Spectral-peak-min is following the energy of noise, with better accuracy than centroid.*

It should be pointed out that these descriptors could have overlapping meanings, and are used together to reinforce the information. Spectral centroid and LPC-min could be similar on noisy sounds, but when a strong formant is present the centroid may loose meaning compared to LPC-min (Fig. 1). Spectral centroid and Spectral-peak-min could also be similar on noisy signal, but when a strong partial exists at the pitch the Spectral-peak-min is better at measuring it (Fig. 2).

The six categories to be recognized relate to evolution of values over time, hence we compute the derivative of each $d_i(k)$. The derivative $d'_i(k)$ is found by linear regression on the local values (5 points on the left and 5 points on the right of k). We finally normalize their range to $[-1, 1]$ using arctangent mapping: $d''_i(k) = 2/\pi \arctan(d'_i(k))$. This completes the computation of the Local Trend descriptors, exemplified in Fig. 3.

For each of the six categories c, we define a continuous hidden Markov model $\mathcal{M}_c$. Each $\mathcal{M}_c$ represents the transition between

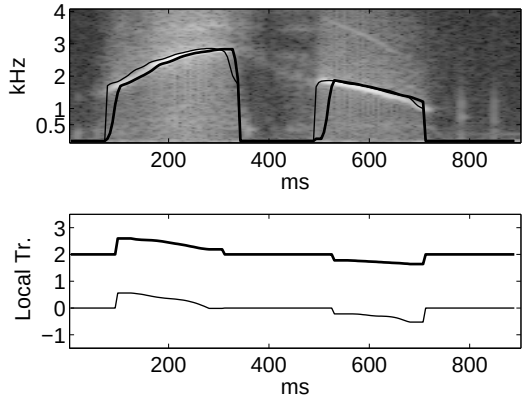


Figure 3: *Example of Local Trend descriptors on* `up/down` *imitation. Topmost panel is the signal spectrogram with spectral centroid (thin line) and Spectral-peak-min (bold line). The bottom panel shows the same two descriptors after Local Trend computation (scale is shifted for Spectral-peak-min for clarity).*

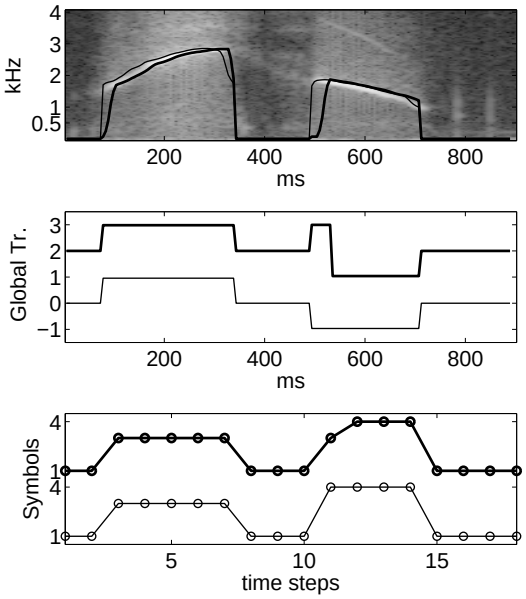


Figure 4: *Example of Global Trend descriptors on* `up/down` *imitation. Topmost panel is the signal spectrogram with spectral centroid (thin line) and Spectral-peak-min (bold line). Middle panel shows the same two descriptors after Global Trend computation (scale is shifted for Spectral-peak-min for clarity). In the bottom panel the two descriptors are quantized into symbols and downsampled (once more Spectral-peak-min has been shifted).*

a set of $S = 4$ states. The emission probability (probability of emitting $d_i''(k)$ given state s) is modeled as a mixture of $M = 8$ Gaussians and diagonal covariance matrix. The HMMs are created in a supervised way. To understand this, let's consider the case of the `up/down` category. This one can be represented as the transition from a state "silent" to state "up" to state "down" and back to state "silence". In the same spirit the `up` category can be represented as the succession of states "silence", "up", "silence". We therefore define 4 states: "silence", "up", "down", "stable". The training of the six HMMs is done in two stages:

- We first train the observation probabilities. This is done independently of $\mathcal{M}_c$. Indeed, given that a state (such as "up" in the above example) can be shared by different $\mathcal{M}_c$, we train the emission probabilities using descriptors from the `up`, `down` and `stable` classes, plus an added silent class.
- The transition probabilities are trained for each category.

Considering that the number of self-transition $s(t+1) = s(t)$ is much larger than the non-self ones $s(t+1) \neq s(t)$, we found the training of the last difficult. In order to circumvent this, we decided to decimate over time the descriptors time series by a factor of 3. This allowed to increase the performances. Also, rather counterintuitively, better results were obtained by forcing the HMM training to *not* update the emission probabilities mixtures (thus only updating the transition matrix).

2.2. Global Trend descriptors

The same 6 audio features of sec. 2.1 are used as underlying time series for the Global Trend descriptors: the spectral centroid, spectral spread, spectral rolloff, pitch, LPC-min and Spectral-peak-min. Instead of using them directly as input to a continuous HMM, we convert them to symbols.

For each descriptor $d_i(k)$ and each active region r we apply the following procedure:

- We compute the linear regression over the region r. We denote by α_i^r its angular coefficient and by ϵ_i^r its prediction error.
- If ϵ_i^r is larger than a specific threshold K_1[3], r is split into two regions at the position of the maximum value of $d_i(k)$ and a two-piece minimum least-square linear regression is computed. We denote by α_i^{r1} and α_i^{r2} the corresponding angular coefficients. This process will allow us to discriminate between monotonic classes (such as `up` or `down`) and `up/down`.
- The quantized time serie $e_i(k)$ corresponding to $d_i(k)$ is then built. It has the value $e_i(k) = 0$ for k corresponding to silent part, $e_i(k) = \alpha_i^r$ for k corresponding to region r (or α_i^{r1} for region $r1$ and α_i^{r2} for region $r2$).

We then convert the values of $e_i(k)$ to symbols using the following rules:

$$e_i'(k) = \begin{cases} 1 & \text{if } |e_i(k)| \leq 10^{-7}; \\ 2 & \text{if } 10^{-7} < |e_i(k)| \leq 0.1; \\ 3 & \text{if } e_i(k) > 0.1; \\ 4 & \text{if } e_i(k) < -0.1. \end{cases} \quad (1)$$

The four symbols {1,2,3,4} express the overall condition of the time serie, respectively: silence, stable (small angular coefficient),

[3] K_1 has been optimized by grid-search. We use a value of $3 \cdot 10^3$.

upward (large positive angular coefficient), downward (large negative angular coefficient).

As a final step, the descriptors series $e'_i(k)$ are decimated over time by taking one value every 10 frames. It should be noted that we have chosen to not apply any antialiasing process, since we have found by experiment that active regions shorter than 10 samples usually correspond to errors. In Fig. 4 we illustrate the Global Trend descriptors.

Training. Since the time series $e'_i(k)$ are symbols (unordered values) we model them using discrete hidden Markov models. Each descriptor i is modeled by its own HMM $\mathcal{M}_{c,i}$.

For a given class c and a given descriptor i we denote by $E_{c,i}$ its emission matrix (with size $S \times O$, where S is the number of hidden states and O the number of symbols), by $T_{c,i}$ the transition matrix (with sizes $S \times S$) and by $S_{c,i}(k)$ the decoded states at frame k.

In order to train the emission matrix for class c and descriptor i, we concatenate all descriptors of the sounds belonging to c into a matrix D_c. Each row of D_c corresponds to a given descriptor i.

We define a function $\mathcal{F}$ that normalizes an input matrix such that its rows elements sum up to 1.

The training algorithm is the following:

1. Set the initial value for the emission matrix to $E_{c,i} = \mathcal{F}(I + 0.05)$, where I is the diagonal identity matrix. This almost associates each state to a single symbol, but does not exclude the possibility for each state to emit each possible symbol.
2. Exploiting the previous association, estimate $T_{c,i}$ by accumulating all the emissions transitions found in row i of D_c into the matrix T. Obtain the transition matrix for i as $T_{c,i} = \mathcal{F}(T)$.
3. Use $T_{c,i}$ and $E_{c,i}$ to estimate the hidden states $S_{c,i}(k)$ by Viterbi decoding;
4. Re-estimate $E_{c,i}$ by counting the number of times each state generates each emission, and again normalizing by $\mathcal{F}$.

In principle, the re-estimation procedure (points 2.-4.) can be iterated, but early experiments showed little performance improvements.

Classification. In order to classify an unknown sound represented by its time series matrix D_*, we decode each row i of D_* using a specific class model $\mathcal{M}_{c,i}$ and the Viterbi decoding algorithm. Each model $\mathcal{M}_{c,i}$ provides a likelihood $l_{c,i}$. The final class label is found as $x = \text{argmax}_{c\in[1,6]} \sum_i l_{c,i}$.

2.3. Morphological descriptors

We introduce here a new set of *morphological* descriptors. These are crafted to compactly describe the structure of the signals present in the dataset.

Each audio file is represented by these descriptors using a vector with 8 components:

- Ψ_1, Ψ_2 and Ψ_3 measure repetitions or patterns in the serie;
- Ψ_4, Ψ_5 and Ψ_6 describe the active region(s);
- Ψ_7 and Ψ_8 are related to the global signal trend.

The descriptors embed directly the time evolution of the signal. Because of this they can be used directly for classification without requiring the use of HMM for temporal modeling. Moreover the Ψ_i values lay in homogeneous ranges. Because of this we will use for classification a simple k-Nearest Neighbor algorithm with Euclidean distance in the following.

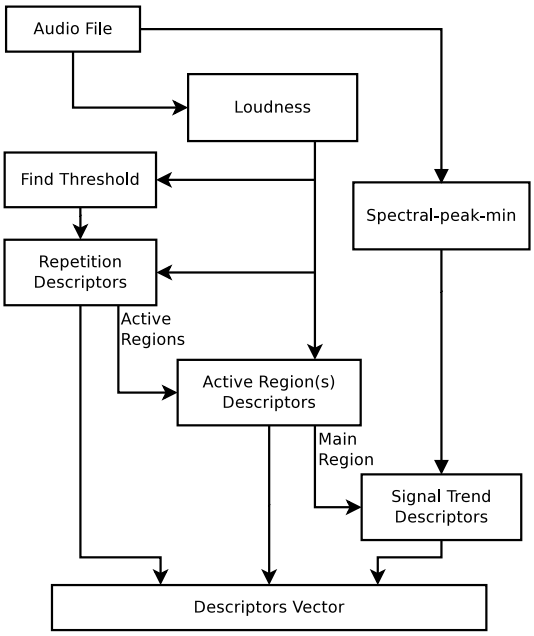


Figure 5: *Computation of morphological descriptors.*

Fig. 5 points out the main steps of the computation.

First the non-silent regions $A = \{[b_r, f_r] : r \in [1, ..., N']\}$ are detected.

For two consecutive active regions, r and $r+1$, we define the duty-cycle of r as $u_r = (f_r - b_r)/(b_{(r+1)} - b_r)$ (see Fig. 6). In this, we assume that every active region is followed by a silent one. When adjacent regions are in the same state (silent or non-silent) we simply merge them, thus enforcing the active/non-active alternance. The first two pattern descriptors Ψ_1 and Ψ_2 are defined as the mean and the standard deviation of non-silent regions duty-cycles u_r, $r \in [1, N']$. `impulse` and `repetition` classes are expected to have small values of Ψ_1. In the opposite, the other classes (stable, up, down, up/down) will have a single long active region with a large value of Ψ_1. Ψ_2 measures the regularity of the repeated patterns. In the case of a single active region, $\Psi_2 = 0$.

We define the "importance" i_r of an active region as the product between its length l_r and its mean loudness m_r (over the active region duration). Both l_r and m_r are normalized in the range $[0, 1]$ for each signal (the value of 1 is assigned to the longest and the loudest regions respectively).

The third descriptor Ψ_3 is the number of active regions which have *Importance* i_r above a threshold K_2. A value of $K_2 = 0.25$ is chosen in order to correspond to the product of half-range normalized values of l_r and m_r. Ψ_3 is computed as:

$$\Psi_3 = \frac{\arctan\left(\text{card}\left(\{i_r \text{ such that } i_r > K_2\}\right) - 1\right)}{(\pi/2)} \qquad (2)$$

The threshold on i_r allows the rejection of very short active

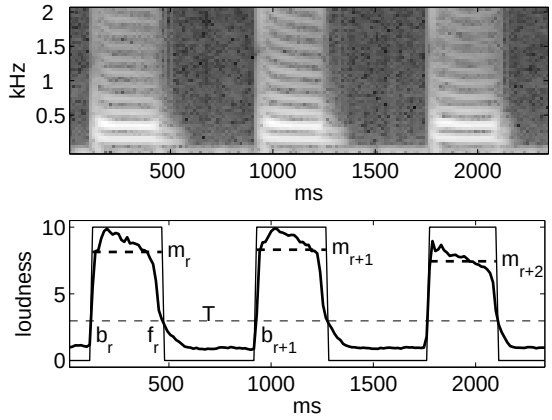


Figure 6: *Computation of descriptors Ψ_1, Ψ_2 and Ψ_3 on a* `repetition` *imitation. The loudness profile is showed (bold line), along with threshold T (dashed line) and active region detection (thin line); mean loudness value m_r (dashed bold line) is used to compute importance i_r.*

regions, or with very low loudness level. For single-region signals Φ_3 is equal to 0, while for signals with three or more regions (typical of `repetition` category) Ψ_3 is above 0.7.

Descriptors Ψ_4 and Ψ_5 focus on the main region R only, that is the non-silent region with highest importance i_R.

The Ψ_4 descriptor is the duty cycle of the main region *defined on the whole signal*: $\Psi_4 = (f_R - b_R)/N$, where b_R and f_R are the start and the end of R and N is the total signal length. Ψ_4 is expected to discriminate `impulse` and `stable` classes.

The descriptor Ψ_5 is computed on the loudness time serie in the main region $d_L(k)$, $k \in [b_R, f_R]$, which has length l_R. The original serie and its half-length circularly-shifted copy are used:

$$\Psi_5 = \sum_{k=1}^{l_R} \left[d_L(k) - d_L\left((k + \frac{l_R}{2}) \bmod l_R \right) \right]^2 \qquad (3)$$

Ψ_5 is thus the energy of the difference between $d_L(k)$ and its shifted copy. The descriptor is then normalized in the $[0, 1]$ range using $\arctan$, as for Ψ_3. Ψ_5 improves the discrimination between classes which have flat or non-flat evolution, such as `up/down` *vs* `stable`. The descriptor Ψ_6 is defined as the sum of the active regions lengths l_r, minus a constant γ. Ψ_6 is then normalized by arctangent, such as in (2), and γ is optimized to have the "shift" of the arctangent function improving the discrimination between `impulse` and `stable` (or `repetition`) classes.

The Ψ_7 and Ψ_8 morphological descriptors have been developed to measure the slope of the signal. The aim is to evaluate the slope between the beginning and the middle, and between the middle and the end, of a given time serie.

Spectral-peak-min is taken as an underlying descriptor: only its values in the main region R are considered, deleting those below 40Hz as they are unreliable. To overcome boundary effects, Spectral-peak-min is observed within three windows of 11 samples taken at the beginning, the middle and the end of the region

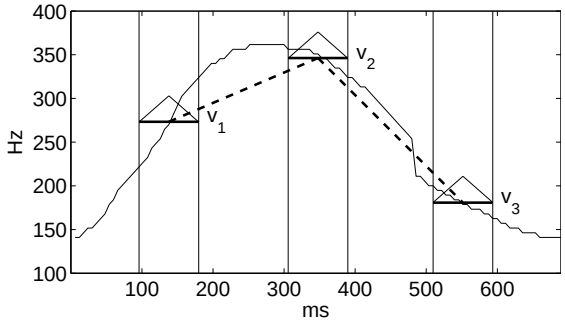


Figure 7: *Computation of descriptors Ψ_7 and Ψ_8 on an* `up/down` *imitation. Spectral-peak-min is showed (thin line), with 3 windows centered at $1/5$, $1/2$ and $4/5$ of its total length. Mean values v_j (bold line) are used to find Ψ_7 and Ψ_8, which are proportional to the slopes (dashed bold line).*

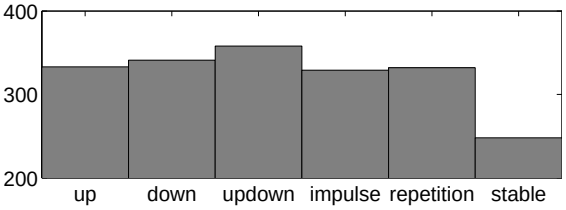


Figure 8: *Distribution into the six categories of the dataset.*

(see Fig. 7). In each window, the Spectral-peak-min is weighted by a triangular window function and then the average is computed. This leads to three mean values: $V = [v_1, v_2, v_3]$. The trend descriptors Ψ_7 and Ψ_8 are found as:

$$\Psi_7 = \frac{v_2 - v_1}{v_1} \qquad \Psi_8 = \frac{v_3 - v_2}{v_2} \qquad (4)$$

and normalized using again the $\arctan$ function. Local windows are used in order to smooth the signal, possibly generated by noisy time series, and the triangular functions give more importance to the central part of the window. Ψ_7 and Ψ_8 measure the evolution in time of the signal: they discriminate between `up`, `down` and `up/down`.

3. EVALUATION

3.1. Description of the train and test sets

The dataset used in this paper comes from a perceptual experiment. In this experiment, 50 French subjects were asked to imitate sounds. For each of the 6 classes described in Sec. 1.2, two reference sounds have been selected. After listening to one of the reference sounds (without knowing its class), subjects were asked to imitate it using only voice (VO) or using voice and gesture (VG). In each case, the subject could do several trials. In theory, each class is therefore represented by 1000 audio examples: 2 (reference sounds) times 50 (number of subjects) times 2 (VO and VG) times the number of trials of each subject (ranging from 1 to 5). In practice, considering the variability of the number of trials of each

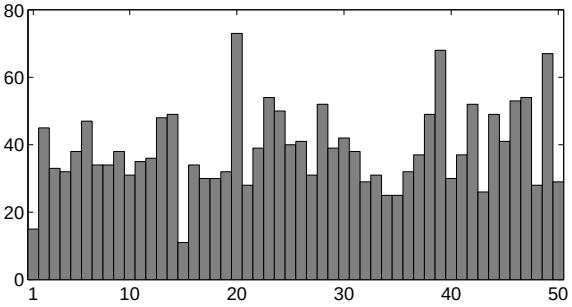


Figure 9: *Number of recordings per subject.*

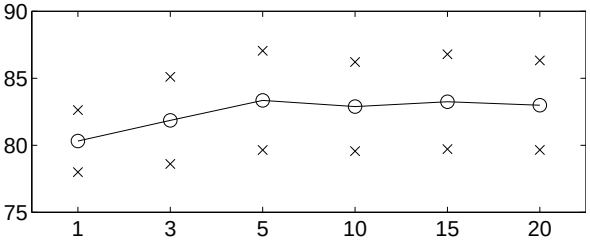


Figure 10: *Recognition accuracy in % (line and circles) as function of k value in k-NN. Results are averaged by 5 folds, and the* ± 1 *standard deviation intervals are marked by* $\times$.

subject, the number of audio examples is lower. We represent the number of audio examples per class in Fig. 8. The average number of imitations provided by each subject is about 40, with large variations as showed in Fig. 9. On average, every subject provided 1.6 trials per stimulus (instead of 5).

The total size of our test-set is 1941 audio files, rather equally distributed among the six categories (except `stable`, see Fig. 8). In the following experiments all the available data is exploited.

3.2. Comparison of the recognition methods

We have described three recognition methods in this paper:

Local Trend: the descriptors rely on local variations of the signal, and are modeled using continuous HMMs;

Global Trend: the main evolutions of the signal are measured by discretized descriptors, modeled using discrete HMMs;

Morphological: the descriptors directly represent both the signal shape and its time evolution; the recognition of classes by the Morphological descriptors relies on the k-Nearest Neighbor algorithm (see sec. 2.3).

The results are presented in Table 1. All the figures have been obtained using 5-folds crossvalidation, selecting train and test set in order to not have the same subject in both. Recall and precision values are given for each of the three methods and for each class. The mean recall and precision, and the overall accuracy, are given at the bottom of Table 1.

The Global Trend descriptors obtain an accuracy of 70.8%, a 28.5% improvement compared to the base-line system (Local Trend, 55.1%). The Morphological descriptors give the best performances, with 83.6% accuracy (51.7% improvement over baseline). Fig. 10 shows the values of accuracy obtained by the system when using different values for k. The best value of $k = 5$ has been selected for the presented results.

Table 1: *Recognition results by different methods, averaged over the 5 crossvalidation folds.*

Methods	Local Trend		Global Trend		Morphol.	
Measures	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.
`up`	80.8	83.9	83.2	81.8	**87.7**	79.6
`down`	**88.5**	43.9	76.2	76.3	71.5	73.7
`up/down`	38.2	53.0	39.1	57.2	**76.3**	76.2
`impulse`	25.3	54.1	60.3	79.1	**91.5**	91.9
`repetition`	29.5	35.5	78.0	62.4	**90.3**	93.2
`stable`	72.9	99.2	**97.2**	70.0	85.8	92.4
Average	55.9	61.6	72.3	71.1	**83.9**	84.5
Avg. Accuracy	55.1		70.8		**83.6**	

It is interesting to look at the results class-by-class. `up` and `down` classes are well recognized by all methods. There are instead classes for which Morphological descriptors are better, such as `repetition`. The opposite case also happens, as the recall for `stable` is better using Global Trend; for `down` the best recall is given by Local Trend, but with poor precision.

The overall conclusion which can be drawn is that Morphological descriptors have a better performance because they work rather well on all classes, giving comparable recall/precision. This is not the case for Local and Global Trend, which have instead one or more classes with particularly bad results.

3.3. Discussion of the results

The Local Trend method has good performances on the `up`, `down` and `stable` classes, but not on the remaining ones. It has been verified that `up/down` recordings are often recognized as `down` (which has in fact a low precision). A possible explanation for the confusion arises by observing the spectrograms of the recordings: it has been found that many subjects prepare the downward slope in `down` by first producing a rising profile (see Fig. 11). Moreover, while `up/down` is recognized as `down`, the confusion with `up` is less frequent: in `up/down` the downward phase is usually stronger, thus justifying the observations.

The global shapes of `up`, `down` and `stable` are well modeled. Classes `impulse` and `repetition` are instead problematic: the transition matrices of the HMMs do not succeed to capture the temporal cues of the signals, and this has a bad influence on classes which are defined relying on that.

The purpose of Global Trend methodology is to provide better modeling of the overall temporal evolution of the signal, hence avoiding these shortcomings. The quantization of the descriptors in four symbols has the effect of keeping only the most relevant information, and the decimation enforces the transition matrices to describe the temporal cues of the signals. Despite having still poor results on `up/down`, Global Trend outperforms Local exactly because it improves the recognition of `impulse` and `repetition`.

The Morphological descriptors have been designed to embed the main characteristics of the classes into a compact and effec-

Table 2: *Confusion matrix for the morphological descriptors.*

Classes	1	2	3	4	5	6
up - 1	**292**	20	13	2	2	4
down - 2	32	**244**	49	9	6	1
up/down - 3	15	47	**273**	11	6	6
impulse - 4	4	11	8	**301**	3	2
repetition - 5	9	7	6	5	**300**	5
stable - 6	16	3	11	0	5	**213**

tive representation. Among their good performances, it should be pointed out the improved discrimination between `down` and `up/down` compared to the previous methods. However, the confusion matrix in Table 2 shows that the issue is still present. Similarly there is confusion between `up` and `down`. `repetition` is well recognized, thanks to the presence of the specific Ψ_3 descriptor. Both `impulse` and `stable` are confused, even if not to a large extent, with `up`, `down` and `up/down`. This could be explained by the fact that the system identifies rising or falling cues even in the `impulse` and `stable` imitations, and provides a classification according to this.

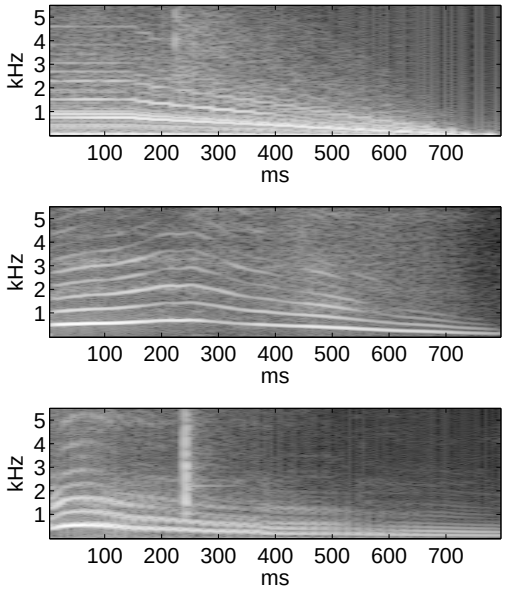


Figure 11: *Example of imitations labeled as class `down`. Topmost panel is one of the two reference stimuli for `down`. The following two panels are imitations of the stimulus: both begin with a rising pitch, followed by the expected lowering; these imitations are likely to be recognized as class `up/down`.*

3.4. Discussion on the dataset

The dataset classes have already been introduced in Sec. 1.2. In the following we give details about the dataset definition which may have an impact on the recognition. The classes `up`, `down`, `up/down` and `stable` are all defined by their evolution in time; this is not the case for `impulse` and `repetition`, which are instead defined by their duration and rhythm, respectively. In other words, the stimuli are labeled according to different domains. Moreover `up` and `down` are semantically portions of `up/down`: it is thus likely to have confusion between these three classes.

There are several factors which lead to a strong intra-class variability. Each class is defined by 2 different reference sounds, and each of the 50 subjects provides many recordings. The imitations are done by non-experts: the same stimuli is thus imitated using different strategies, and sometimes the imitations contents can be conflicting.

There are, finally, some issues related to the content of the dataset, that is to the vocal imitations. The definition of the classes, although very clear, can not be translated straightforwardly into acoustic cues: there are examples in which the subject imitates a complex stimulus by rising the pitch and lowering the first formant (thus decreasing the spectral centroid). The imitation has thus ambiguous meaning because its descriptors will have opposite evolutions. Many recordings suffer from boundary effects: the imitator could begin to produce sounds during the preparation phase of the articulation, introducing spurious effects at the begin of the recording. Similar artifacts can also be spotted at the end of some sounds.

The recognition system therefore has to cope with all the exposed issues of the dataset. The descriptors have to provide the main cues of the signal even in case of noisy or unreliable underlying time series (loudness, spectral centroid, etc.). This situation motivates the use of different descriptors with similar meanings, in order to reinforce the extracted information.

4. CONCLUSION

A dataset has been compiled, in the context of the SkAT-VG project, in which a large number of subjects have imitated sounds which belong to six morphological categories.

We have presented three methodologies to automatically recognize these morphological categories.

Our base-line system uses Local Trend descriptors, which are designed to measure the local behavior of the time serie. The descriptors are then modeled by continuous hidden Markov models with 4 states. The system reached an accuracy of 55.1%. An improved approach is based on Global Trend descriptors, which express the evolution of the signal along its whole duration. This second set of descriptors is modeled by discrete hidden Markov models, using 4 states and 4 emitted symbols. The obtained accuracy is 70.8%. Our third approach for the automatic recognition is based on the Morphological descriptors, which are designed to give a compact representation of the time series evolution. These descriptors do not need temporal modeling, such as HMM, and have low dimensionality. The classification is thus done using the k-Nearest Neighbor algorithm. This system has provided the best recognition accuracy of 83.6%.

The automatic classification on the given dataset proved to be a non-trivial task, despite the apparently clear definition of the classes. The manual check of many recordings within the dataset has confirmed a degree of confusion between certain classes, due to objective spectrograms similarities. A notable example is the pair `down` and `up/down`. Moreover the dataset is made by imitations of reference sounds, and imitators use different strategies to render the same stimulus: this rises the intra-class variability.

The non-optimal recognitions results are in part justified by these findings.

The presented set of Morphological descriptors goes toward the characterization of sounds by their acoustic *shape*. Future work will involve the organization of the descriptors in a systematic topology, similarly to [15]; this may point out shortcomings of the proposed set. Adding new descriptors could hence encompass a broader set of signal categories.

The proposed Morphological descriptors could be applied to other datasets, with classes defined in different ways. The integration of the Morphological descriptors with other, more traditional, ones is another topic of future studies, fostering recognition accuracy in many contexts.

5. ACKNOWLEDGMENTS

This work was supported by the 7th Framework Programme of the European Union (FP7-ICT-2013-C FET-Future Emerging Technologies) under grant agreement 618067. The audio dataset as well as the choice of the categories and the annotations used in this paper have been created by the Perception and Sound Design team of IRCAM. We are grateful to the reviewers and to S. Perry for their valuable contributions and corrections.

6. REFERENCES

[1] G. Lemaitre and D. Rocchesso, "On the effectiveness of vocal imitations and verbal descriptions of sounds," *Journal of the Acoustical Society of America*, vol. 135, no. 2, pp. 862–873, 2014.

[2] G. Lemaitre, A. Dessein, K. Aura, and P. Susini, "Do vocal imitations enable the identification of the imitated sounds?," in *Proceedings of the 8^{th} annual Auditory Perception, Cognition and Action Meeting (APCAM 2009)*, Boston, MA.

[3] G. Lemaitre, A. Dessein, P. Susini, and K. Aura, "Vocal imitations and the identification of sound events," *Ecological Psychology*, vol. 23, no. 4, pp. 267–307, 2011.

[4] M. Cartwright and B. Pardo, "VocalSketch: Vocally imitating audio concepts," in *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, 2015.

[5] Guillaume Lemaitre and Davide Rocchesso, "On the effectiveness of vocal imitations and verbal descriptions of sounds," *The Journal of the Acoustical Society of America*, vol. 135, no. 2, pp. 862–873, 2014.

[6] Lawrence Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[7] S. Furui, *Digital Speech Processing, Synthesis, and Recognition*, New York: Marcel Dekker, 2000.

[8] Lawrence R. Rabiner and Biing-Hwang Juang, *Fundamentals of speech recognition*, vol. 14, PTR Prentice Hall Englewood Cliffs, 1993.

[9] Biing-Hwang Juang and Lawrence R Rabiner, "Automatic speech recognition–a brief history of the technology development," *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara*, vol. 1, 2005.

[10] Mohamed Benzeghiba, Renato De Mori, Olivier Deroo, Stephane Dupont, Teodora Erbes, Denis Jouvet, Luciano Fissore, Pietro Laface, Alfred Mertins, Christophe Ris, et al., "Automatic speech recognition and speech variability: A review," *Speech Communication*, vol. 49, no. 10, pp. 763–786, 2007.

[11] George Saon and Jen-Tzung Chien, "Large-vocabulary continuous speech recognition systems: A look at some recent advances," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 18–33, 2012.

[12] Kazushi Ishihara, Tomohiro Nakatani, Tetsuya Ogata, and Hiroshi G Okuno, "Automatic sound-imitation word recognition from environmental sounds focusing on ambiguity problem in determining phonemes," in *PRICAI 2004: Trends in Artificial Intelligence*, pp. 909–918. Springer, 2004.

[13] Shiva Sundaram and Shrikanth Narayanan, "Vector-based representation and clustering of audio using onomatopoeia words," in *Proceedings of the American Association for Artificial Intelligence (AAAI) symposium series. Arlington, VA*, 2006.

[14] S. Sundaram and S. Narayanan, "Classification of sound clips by two schemes: Using onomatopoeia and semantic labels," in *Multimedia and Expo, 2008 IEEE International Conference on*, June 2008, pp. 1341–1344.

[15] P. Schaeffer, *Trait des objets musicaux*, Paris: Seuil, 1966.

[16] Julien Ricard and Perfecto Herrera, "Morphological sound description: Computational model and usability evaluation," in *Audio Engineering Society Convention 116*. Audio Engineering Society, 2004.

[17] Geoffroy Peeters and Emmanuel Deruty, "Sound indexing using morphological description," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 3, pp. 675–687, March 2010.

[18] Marijn Huijbregts and Franciska de Jong, "Robust speech/non-speech classification in heterogeneous multimedia content," *Speech Communication*, vol. 53, no. 2, pp. 143–153, 2011.

[19] M. Ito and Robert W. Donaldson, "Zero-crossing measurements for analysis and recognition of speech sounds," *Audio and Electroacoustics, IEEE Transactions on*, vol. 19, no. 3, pp. 235–242, 1971.

[20] Javier Ramırez, José C. Segura, Carmen Benıtez, Angel De La Torre, and Antonio Rubio, "Efficient voice activity detection algorithms using long-term speech information," *Speech Communication*, vol. 42, no. 3, pp. 271–287, 2004.

[21] Geoffroy Peeters, "A large set of audio features for sound description (similarity and classification) in the Cuidado project," Cuidado project report, IRCAM, 2004.

[22] A. Camacho and J. G. Harris, "A sawtooth waveform inspired pitch estimator for speech and music," *Journal of the Acoustical Society of America*, vol. 124, pp. 1638–1652, 2008.

[23] S. McCandless, "An algorithm for automatic formant extraction using linear prediction spectra," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 22, no. 2, pp. 135–141, Apr 1974.

[24] J. D. Markel and A. H. Gray, *Linear Prediction of Speech*, Springer-Verlag, 1976.

ON STUDYING AUDITORY DISTANCE PERCEPTION IN CONCERT HALLS WITH MULTICHANNEL AURALIZATIONS

Antti Kuusinen,
Aalto University School of Science
Dept. of Computer Science
Espoo, Finland
antti.kuusinen@aalto.fi

Tapio Lokki,
Aalto University School of Science
Dept. of Computer Science
Espoo, Finland
tapio.lokki@aalto.fi

ABSTRACT

Virtual acoustics and auralizations have been previously used to study the perceptual properties of concert hall acoustics in a descriptive profiling framework. The results have indicated that the apparent auditory distance to the orchestra might play a crucial role in enhancing the listening experience and the appraisal of hall acoustics. However, it is unknown how the acoustics of the hall influence auditory distance perception in such large spaces. Here, we present one step towards studying auditory distance perception in concert halls with virtual acoustics. The aims of this investigation were to evaluate the feasibility of the auralizations and the system to study perceived distances as well as to obtain first evidence on the effects of hall acoustics and the source materials to distance perception. Auralizations were made from measured spatial impulse responses in two concert halls at 14 and 22 meter distances from the center of a calibrated loudspeaker orchestra on stage. Anechoic source materials included symphonic music and pink noise as well as signals produced by concatenating random segments of anechoic instrument recordings. Forty naive test subjects were blindfolded before entering the listening room, where they verbally reported distances to sound sources in the auralizations. Despite the large variance in distance judgments between the individuals, the reported distances were on average in the same range as the actual distances. The results show significant main effects of halls, distances and signals, but also some unexpected effects associated with the presentation order of the stimuli.

1. INTRODUCTION

Virtual acoustics and auralizations offer many possibilities to study the perceptual aspects of room acoustics. For example, it is possible to perform instantaneous side-by-side comparisons of different acoustic conditions; comparisons that are impossible to perform in real environments. Currently, there are a number of spatial sound technologies including ambisonics [1], wave-field synthesis [2], directional audio coding [3] and spatial decomposition method (SDM) [4] which can be used to recreate a measured or simulated sound field. Reproduction is realised either binaurally by headphones or by a multichannel loudspeaker setup in a (semi-)anechoic listening room. This article discusses a listening experiment on auditory distance perception (referred to with 'distance perception' or similar terms in the rest of the text) in concert halls with auralizations produced by SDM and reproduced in a multichannel setup.

Multichannel auralizations have been previously used in a number of formal listening experiments (e.g. [5]). Because the possible perceptual biases due to signal processing are the same in all stimuli, it is and it has been reasonable to assume that the perceptual differences between the auralizations of different halls and seating positions are representative of the differences between real conditions. Although the previous investigations together with the myriad of discussions with the test subjects and various experts who have listened to these auralizations do not give any reason to doubt the validity of this assumption, it is still formally unclear whether the auralized sources are actually perceived as being at the distance where they would be perceived in the real concert hall. Formal evidence on realistic perception of the distance to sound sources would greatly enhance the credibility of the results of any study - past and future - where these auralizations are being used as stimuli.

The motivation for the present study is twofold. On one hand, the work is motivated by the need to further test our system and auralizations with different anechoic source materials and especially with listeners who have little or none prior experience with spatial sound systems. Specific focus is on the very first perceptions of the auralizations, before any perceptual adaptation or calibration and/or learning has taken place. The first perception data - although susceptible to be more variable and less accurate - may be the most unbiased indication that the auditory perceptions of the sound sources in the auralizations are comparable to the perceptions of their real counterparts.

On the other hand, the previous studies [5, 6] on concert hall acoustics have indicated that "proximity" (or "intimacy" [7]), that is, the feeling of being close to the performers is one of the most important aspects of the listening experience in concert halls. This aspect is possibly linked to perceived distance, but due to the lack of substantial evidence on distance perception concerning large spaces, little is certain. Thus, we also seek to obtain preliminary evidence on the differences in perceived distances between concert halls before continuing with more detailed investigations in this respect.

2. BACKGROUND ON AUDITORY DISTANCE PERCEPTION

Considering distance perception to sound sources outside a few meters range from the listener, the main acoustic distance cues are intensity (or loudness), direct-to-reverberant energy ratio (DRR) and frequency spectrum [8, 9]. Intensity has been found to act as a relative cue, whereas DRR seems to act as an absolute distance cue [10, 11] - at least when a sound is perceived the first time. People might also use, or weight, cues differently with different signals. For instance it has been found that intensity cue is weighted more

with speech whereas the distance to a noise source was determined more by DRR [12]. In the context of concert hall acoustics, the sound strength G is commonly used instead of sound intensity to measure the perceived loudness of the sound field. G is normalised by the source sound intensity at 10 meters in free field, and thus, reflects the contribution of the room. G is also used in this article instead of sound intensity.

Also the inter-aural level and -time differences (ILDs, ITDs) have been found to act as distance cues for sources near the listener. In larger spaces, such as concert halls, the effectiveness of these inter-aural cues is unclear, but they are related to inter-aural cross-correlation (IACC), which measures the similarity of incoming sounds between the two ears. IACC have been linked to various perceptual qualities of concert hall acoustics, such as, width and envelopment.

Familiarity to source characteristics has also been found to influence distance judgments [8]. In most cases listeners have some a priori (long term) knowledge about how the sound is perceived at different distances and/or at different output levels. Musical instruments are typical sources that produce sounds which vary systematically in spectral content with playing dynamics. Moreover, the changes in spectral information, for instance, the attenuation of higher frequencies due air absorption, has been found to serve as a relative distance cue, independent of the variation in overall sound level [13].

Finally, the relationship between the physical distances and the perceived distances is known to follow a power function in the form $p = kr^a$ where k is a linear scaling factor and a is the exponent indicating the amount of "compression" ($a < 1$) or "expansion" ($a > 1$) of the perceived distances (p) compared with real physical distances (r) [8]. Average values for the k have been reported being around 1.32 and for the exponent a around 0.54, but because most previous studies included only distances up to around 10 meters, it is unclear whether these values are also representative of larger spaces.

3. METHODS AND MATERIALS

3.1. Spatial room impulse response measurements

Acoustic measurements were made with a calibrated array of loudspeakers, i.e., a loudspeaker orchestra [14] as a sound source and an array of six omnidirectional microphones as a receiver. Measurements and the processing with SDM were performed separately per each loudspeaker source on stage. The room impulse responses were performed with the swept-sine technique [15]. Details of the loudspeaker orchestra, the measurement technique and the technical equipment have been described in previous publications [16, 17].

Spatial room impulse responses (SRIRs) were obtained with SDM [4] which extracts the spatio-temporal evolution of the sound field from the impulse responses captured by the microphone array. The estimation of the directions of the arriving sounds is based on a combination of time-of-arrival and time-difference-of-arrival estimates calculated from the six pressure values of the omnidirectional microphone sensors. In the current implementation, the analysis is carried out in a sliding temporal window of 2 ms, with a hop-size of one sample (99 % overlap). This window length has been chosen to be in-line with the current knowledge about temporal resolution of human hearing [18]. By considering the echo density [19] of these concert halls and the length of the time window, it is possible to approximate a time instant when it is probable that more than one reflection occurs in the analysis window. The time instants for present halls with volumes of 15000 m^3 (BK) and 16000 m^3 (SB) are 120 ms and 124 ms, respectively. If this approximation is valid, SDM will yield accurate estimates for the direct sound and the early reflections. When the echo density increases, the sound field becomes more diffuse and stochastic, and more and more reflections from different directions will be included in the analysis window. When this happens, SDM will still yield only one direction estimate per analysis window, but these estimates in overall behave more or less randomly depending on the properties of the sound field. Therefore, in a diffuse sound field, SDM yields stochastic (diffuse) estimates.

By employing a hop size of one sample, this spatio-temporal analysis gives each sample in the impulse response a direction, as well as a pressure value which is obtained from one of the omnidirectional microphone responses. It is noteworthy that, for the microphone array used here, the reported RMSE error of the estimated locations of loudspeaker sources in real concert hall was 2 degrees in direction and 0.4 meters in distance [20].

SRIRs containing the information on directions and pressures for each sample are translated to the reproduction loudspeakers in the listening room. In this study, the direction of each sample in the spatial impulse response is used to position that sample in the direction of the nearest (the smallest angle difference) loudspeaker in the listening room setup, thus, distributing the original spatial impulse response to the spatial configuration of the reproduction loudspeakers. One alternative to this direct panning technique is amplitude panning between the loudspeakers [21], but it has been found to decrease the spectral brightness in the auralizations of concert hall acoustics [17].

After translating the impulse responses to the reproduction loudspeakers, they are convolved with anechoic instrument recordings [22]. Naturally, the convolutions with anechoic recordings were made in respect to the instrument positions in a real orchestra although practically any arrangement is possible. Finally, the convolved signals of each instrument are summed per reproduction channel to produce final samples with 24 channels for playback. Max/MSP 6 software was used to playback the audio and to set the output at a comfortable listening level.

A noteworthy difference between these auralizations and in-situ listening of a real orchestra is that all the perceptual cues in auralizations are less dynamic than in reality as they are produced by stationary sound sources with directivities that differ from those of real instruments [14]. It is currently unknown how such dynamic aspects affect the perception of sound sources. It is also unclear how the distance cues from different sources are used together and weighted when the task is to evaluate the distance to multiple distributed sound sources, instead of just one single source.

3.2. Concert halls

Strong reflections from the side has been found to enhance musical dynamics [23], what seems to be one governing aspect differentiating traditional shoebox shaped halls from other designs, such as the fan shape. To obtain some evidence whether the shape has an effect also on perceived distances, this study included a shoebox shaped Konzerthaus in Berlin (seats = 1575, volume = 15000 m^3) and an irregular fan shaped Beethovenhalle in Stuttgart (seats = 2000, volume = 16000 m^3), both measured without an audience. Concert hall layout and cross section are illustrated in Fig. 1. On-

site measured sound pressure A-levels with pink noise played back simultaneously from all the sources on stage were 84.4 dB (14 m to the center of the orchestra) and 82.6 dB (22 m) in BK, and 83 dB and 81.4 dB in SB. Taking these values at 14 meters, we calculated the theoretical value for 22 meters in free field conditions, and subtracted those from the measured SPLs. For both halls, the boost in SPLs at 22 meters was about 3 dB compared with the theoretical free field conditions.

The values for room acoustical parameters are tabulated in Table 1 and the values of G and DRR, which are the main distance cues in this context are also illustrated in Fig. 2.

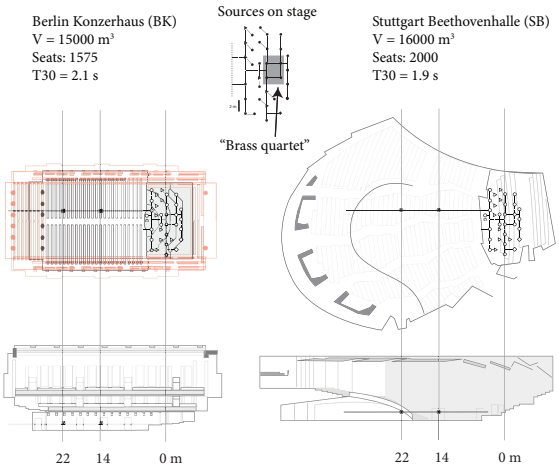


Figure 1: *Layouts and cross sections of the studied halls, and the positions of the sources in the loudspeaker orchestra on stage.*

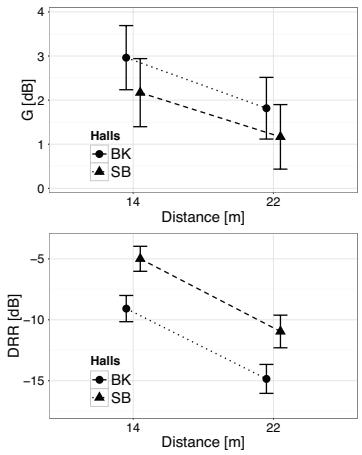


Figure 2: *G and DRR values in each hall over the distances used in the study. Values are averaged over the 500 Hz and 1 kHz octave bands. Bars repsresent the 95 % confidence intervals calculated over the 24 sources and the two frequency bands.*

3.3. Listening room

Listening room meets the ITU BS.1116 recommendation [24], with the exception that only the noise rating (NR) 30 is met, whereas NR 15 is recommended. Also the listening distance from the reproduction loudspeakers is less than the recommended two meters, about 1.5 meters on average. 24 reproduction loudspeakers are positioned around the listening position in 3-D layout. There are a few more loudspeakers in the frontal direction to account for the fact that human sound localisation accuracy is the greatest in the front. Details of the loudspeaker setup have been provided by Haapaniemi and Lokki [16]. The listening room and the loudspeaker layout are also illustrated in Fig. 3, where ITD contours are depicted in the figure for illustration purposes.

Because the listening room and reproduction setup might affect the relative loudness differences in the auralizations compared with real halls, we made additional auralizations with the same pink noise used in on-site measurements and measured the difference in SPLs between these auralizations and the on-site measurements. For on-site measurements, the differences between 14 and 22 meters were 1.8 dB and 1.6 dB for BK and SB, respectively. The same differences in the listening room measurements were 1.4 dB and 1.2 dB indicating the compression of 0.4 dB due to the listening room. Thus, the relative differences in real conditions might actually be a little greater than can be perceived in our listening room. In addition, the listening room itself is not anechoic but damped with absorbers and has a reverberation time of 0.2 seconds at mid-frequencies (averaged over 0.5-1 kHz frequency bands). Hence, this reverberation is confounded in sounds evaluated by the listeners.

For the listening, the test samples were set at a comfortable listening level with the average of approximately 78 to 80 dB across all samples. The actual sound levels were highly variable due to the nature and duration of the anechoic signals. The output level was determined by informal listening to not to overwhelm any listeners during the experiment. It is worth to mention that the focus here is in the relative differences between the conditions, without attempting to calibrate the listening levels to correspond to any 'true' references. We also considered to systematically randomise the output levels, but it was decided to be left out from this study.

3.4. Anechoic signals

ID	Descr.	Dur.	Sources	Configuration
M	Bruckner symphony	60 s	1 - 24	classic orc. config.
F	musical cacophony	48 s	1 - 24	classic orc. config.
B	stream of brass quartet	26 s	15, 16 17, 18	2 trumpets & 2 trombones
N	pink noise	5 s	1- 24	classic orc. config.

Table 2: Summary of the anechoic signals used in the listening experiment.

Table 2 summarises the types of anechoic signals used in the listening experiment. The source positions on stage are illustrated in Fig. 1. The signal abbreviated as 'M' is one minute excerpt

Hall	EDT (s)	DRR (dB)	G (dB)	C_{80} (dB)	J_{LF}	L_J (dB)	$IACC$
BK	2.1, 2.1	-4.0, -7.6	2.9, 1.8	-1.7, -2.3	0.25, 0.26	-2.6, -3.4	0.39, 0.31
SB	2.3, 2.1	-1.5, -5.4	2.1, 1.1	0.6, -1.2	0.11, 0.11	-5.5, -5.8	0.56, 0.48

Table 1: Values for acoustical parameters for the two distances D1 (14 m) and D2 (22 m) in each hall. Values are averaged over the 500 Hz and 1 kHz octave bands except for J_{LF}, which is averaged over the 125 Hz, 250 Hz, 500 Hz, and 1 kHz octave bands and L_J which is energy averaged over the 125 Hz, 250 Hz, 500 Hz, and 1 kHz octave bands.

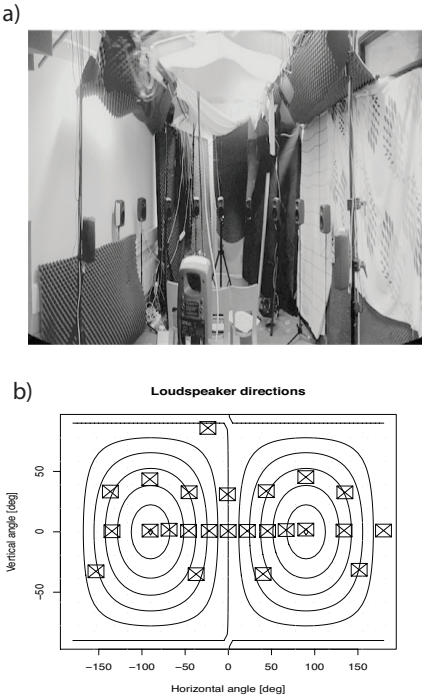


Figure 3: *a) Photo of the listening room. The listening position, i.e., the sweet spot is in the middle of the room. b) Illustration of the loudspeaker positions in the listening room. The ITD contours are drawn to better illustrate the directions of the loudspeakers in relation to the ears of the listeners.*

of Bruckner's 8th symphony, second movement. Shorter excerpts of this composition have been used in our previous investigations [26, 5], but here we wanted to provide test subjects with enough time for making their distance judgment.

In contrast to this excerpt of Bruckner's symphony, two signals ('F' and 'B') were constructed according to a procedure proposed by Kuusinen [27]. In brief, the anechoic recordings of excerpts of symphonies by Mozart, Beethoven, Mahler and Bruckner, where each individual instrument has been recorded separately, were first cut into variable length (short) segments. Then randomly selected segments (per each instrument) were concatenated to produce each instrument track. A certain amount (30 % of the total duration) of silence was required to be left in the individual tracks in order to reduce the cacophony observed in informal listening to various outputs of this procedure. Here, two different configurations of instruments was used. For the signal denoted as 'F', the configuration of the virtual orchestra approximated the configuration of a full classical orchestra. 'B' signal represents a brass quartet consisting of two trumpets and two trombones located in the middle of the stage, see Fig. 1. The reason to select brass instruments was that the directivity patterns of these instruments [28] are similar to those of the loudspeakers used as sources in the acoustic measurements.

Because a single randomisation procedure might yield an output which is far from an 'average' output, this procedure was run 100 times per each configuration. The acoustic features in these 100 samples were then analysed with MIRToolbox [29] in Matlab, using the *mirfeatures*-function, which extracts the averages of a number of acoustic features from the signals (see [29] for details). Next, this feature data was subjected to principal component analysis with varimax rotation on the first 5 components. The sample scores on these components were used to find the individual sample nearest to the center of the sample space in terms of Euclidean distances. The values of the acoustic features themselves are not investigated here, as they were only used as a means for sample selection.

The fourth sample type was a 5-second long pink noise, which was generated separately for each source to avoid unwanted additive effects when convolved with the spatial impulse responses.

3.5. Subjects

Forty subjects (12 women and 28 men in ages between 21 and 58 years) were recruited from people working in the department of Computer Science and from the building lobby. The recruitment process was facilitated by restricting the total duration of the experiment under 20 minutes, meaning that the listening itself was to be performed in 15 minutes, and that some compromises in the experimental design were to be made. Seven subjects reported having some prior experience in listening experiments, and four subjects had been in the listening room beforehand. The participants did not report any known hearing impairments when asked. This was the only screening criterium although it was not formally tested due to the time restrictions.

3.6. Listening experiment

Inspired by the experimental design used by Mershon and King [10], the experiment was based on an idea that each subject can give only one unbiased absolute distance estimate, that is, for the very first sound they hear in the experiment. All the other sounds are judged in relation to the first judgment and thus biased by the 'priming' or 'anchoring' effect of the first sound. Accordingly any type of training (e.g., sequences or otherwise) was also excluded from the experiment. Also, due to restricting the duration of the listening to 15 minutes, the repetitions of any of the stimuli were not included in the experiment.

Before entering the listening room, written and verbal instructions of the test procedure was presented to the test subjects. The main instructions were given as:

"Shortly, you will be blindfolded and guided to the listening room to listen to a set of sound samples. After each sample, your task is to verbally report the distance to the source/s of the sound in meters. If you hear multiple sources, report the distance to the center of the sources. You will hear each sound sample only once. Try to be as accurate as you can, but give the answer at least in one meter accuracy."

Then the subjects were blindfolded and guided to a chair at the listening position in the listening room. The experimenter stayed at the corner of the room to playback the sound samples from a laptop and to write down listener's verbal responses. The sound samples were presented one by one and after each sample the subject reported the distance to the sound sources in meters. Experimenter verbally repeated the reported distance to avoid any mistakes in the responses. An alternative to having the experimenter present in the room was to provide the listeners with a writing pad and a pen, as was done by Mershon and King [10], but this was considered unfeasible in practice without expanding the duration of the experiment.

The first part (i.e., first 4 sounds) of the experiment was a balanced between subjects block design with the one-minute long excerpt of Bruckner's symphony ('M'). The subjects were first divided into two groups of 20 people, according to the hall designated to the first two samples, that is, either BK or SB. In each group, half of the subjects (n = 10) were first presented with the closer sound sample and the other half with the further one. After the first four samples, the listening continued with the other three source materials ('F', 'B' and 'N') in random order. The presentation orders of the acoustic conditions within these signals were randomised.

4. RESULTS

The data consists of total of 640 observations, each being a combination of subject (N = 40), signal ('M','F', 'B' and 'N'), hall (SB and BK), and seating position (i.e., distances D1 and D2). Thus, there are 320 data points for each hall, 320 for each positions (abbr. pos), 160 for each signal and 40 for each combination of hall, distance and signal, referred to as 'sample' in the following. The 5 %, 25 %, 50 %, 75 %, 95 % percentiles for the whole data are 3.0, 7.4, 15.0, 28.0, and 50.0 meters, respectively. Fig. 4 a) and b) show histograms and normal quartile-quartile (QQ) residual plots for the data. Histogram illustrates that listeners often used the resolution of 5 meters instead of one meter which was given in the instructions. This behaviour is also seen as "steps" in the QQ-plot, which in addition shows that the data is approximately normal when the results are transformed to logarithmic coordinates. The geometric means for two distances across subjects and samples are 12.0 m for D1 and 14.8 m for D2 in BK and 13.2 m and 15.9 m in SB. Fig. 4 c) and d) shows the data per each subject and per each sample, repectively. Subjects have used very different scales in their responses, and few subjects reported some "wild" distance estimates (max = 240 m) in the case of pink noise.

Analysis of variance (anova) was carried out in log-transformed responses. Note that any formal test of normality will fail on this data set, because the data is effectively discrete (with resolution of 1 meter in untransformed coordinates). However, the analysis of variance and the corresponding F-tests, are known to be robust against the violations of the normality [30]. The main results were checked with non-parametric tests (Kruskal- Wallis) with the same results, but those are omitted here for brevity.

One main target was to investigate the distance judgements to the first presented stimuli. The first four stimuli were presented in a balanced between subjects block design, where the blocking factor was the presentation order of two halls. Anova results of these four samples are tabulated in Table 3. There is a significant main effect of position ($F(1,38) = 11.8$, $p < 0.01$) as well as an interaction effect between hall and the between subjects blocking factor (hall BK or hall SB first; $F(1,38) = 11.5$, $p < 0.01$)). The block effect is illustrated in Fig. 5 a) where each hall is judged closer/further depending on which one has been presented first. This indicates that the order of presentation had a major influence on the results.

Source	df	SS	MS	F	p(>F)
Between:					
block	1	0.34	0.34	0.11	0.70
residuals	38	123.3	3.24		
Within:					
hall	1	0.19	0.19	1.2	0.27
pos	1	0.71	0.71	11.8	**< 0.01**
hall*pos	1	0.02	0.02	0.24	0.62
block*hall	1	1.7	1.7	11.5	**< 0.01**
block*pos	1	0.01	0.01	0.25	0.62
block*hall*pos	1	0.02	0.02	0.32	0.57
residuals	38	2.34	0.06		

Table 3: Anova results regarding the first 4 samples. Significant effects ($p < 0.05$) are bold-faced.

Next, we included the rest of the data set in the analysis. The results of this repeated measures anova are tabulated in Table 4. Significant differences were found between the two positions, the two halls as well as between the four signals. There is also a significant interaction between the position and signals, meaning that the differences between the positions were influenced by the properties of the source material. The results are illustrated in Fig. 5 b).

Due to the significant interaction between the between subjects blocking factor and hall in the first anova, we investigated the order effect more closely. Within each signal, the samples were identified by their presentation order, and the factor 'order' was included in anova-model. The results revealed significant order effects for all source material except for the 'F', which is the random cacophony produced by "full orchestra". The order effects are illustrated in Fig. 5 c) and tabulated in Table 4.

5. DISCUSSION

It is noteworthy that the experiment was designed so that a single listening session could be performed in 15 minutes. On the one hand, this restriction facilitated the recruitment of 40 test subjects, but on the other hand, only a very limited number of stimuli could be included. Also, the stimuli were chosen to be long enough in duration in order to give listeners enough time to adapt to the sound field and to decide on their answer. As many as four different source materials were included in the experiment, what excluded

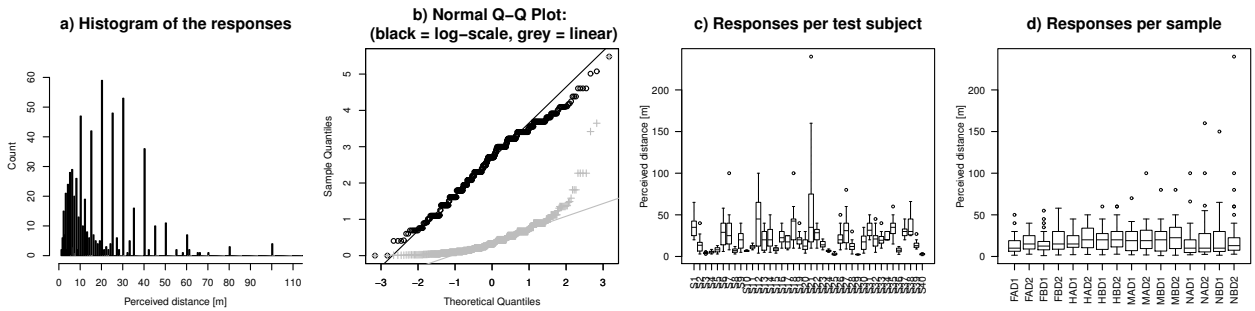


Figure 4: a) Histogram of the data. b) Normal quartile-quartile (QQ) plot of the residuals (grey: linear scale, black: logarithmic scale). c) Data per each test subject. d) Data per each combination of source material (F,H,M,N), hall (A or B) and distance (D1 (14 m) and D2 (22 m).

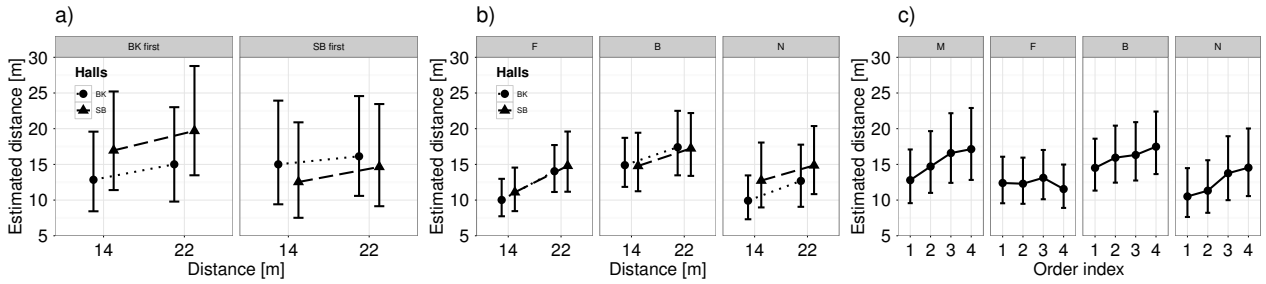


Figure 5: Results: a) The excerpt of Bruckner's symphony ('M') with a balanced between subjects design where half of the listeners listened to BK first (left) and half listened to SB first (right). Also within these groups, half of the subjects were first presented with the closer distance and the other half with the further distance. b) Other source materials. c) Order effect per each source material.

Source	df	SS	MS	F	p(>F)
Between:					
residuals	39	361.8	9.23	-	-
Within:					
pos	1	6.42	6.42	51.2	**< 0.01**
hall	1	1.16	1.16	7.40	**0.01**
signal	3	8.88	2.96	3.6	**0.02**
pos*signal	3	0.77	0.26	3.55	**0.02**
hall*signal	3	0.93	0.31	2.67	0.051
hall*pos	1	0.03	0.03	0.42	0.52
hall*pos*signal	3	0.10	0.03	0.44	0.73
residuals	585	139.5	0.24		
Order effects:					
sig. M	3	2.12	0.71	8.16	**< 0.01**
sig. F	3	0.37	0.12	1.3	0.28
sig. B	3	0.69	0.23	5.7	**< 0.01**
sig. N	3	2.5	0.83	7.2	**< 0.01**

Table 4: Results of overall analysis of variance. Significant effects ($p < 0.05$) are bold-faced.

collecting repeated judgments and thus, the assessment of the reliability of the distance judgments. Since we targeted the unbiased judgments when the sounds are listened to the very first time, it is possible that the results reflect the perceptions made during the period of perceptual calibration, adjustment and learning. This interpretation is supported by the observed order effect. However, when most of the perceptual studies include training sequences to the experimental design, it is sometimes worthwhile to experiment with alternative ways, because outside the laboratory such explicit training sequences hardly exist.

It is also noteworthy that the experimenter was present in the corner of the listening room - about 2.5 meters behind the listeners - verbally repeating the reported distances back to the listeners. Thus, the listeners in fact had an implicit reference from the experimenter's voice although this distance was never asked or told to the subjects. Nevertheless, for some people this reference may have anchored them to the dimensions of the listening room. There were few people who reported all distances to the range of the reproduction loudspeakers, what also speaks against the desired illusion of being in a larger space.

With these considerations in mind, the results showed that the majority of the listeners reported distances which were well beyond the dimensions of the listening room. Previous studies on distance perception have reported large inter-individual variances [8] and this study was not an exception. Especially the convolved pink noise was proven to be a difficult sample to judge as illustrated by the very large variance between the listeners. Neverthe-

less, in overall the judgement are well in the range of the actual physical distances. The perceptual distances were also underestimated which is in-line with the previous knowledge.

Considering the first four samples, there is a peculiar relationship between the measured DRR and strength parameter G, as illustrated in Fig. 2 and the distance estimates, as illustrated in Fig. 5a. When BK was presented first, the judgments follow G, and the higher DRR in SB seems to be counteracted by the lesser overall strength of the sound field. When SB was presented first, the judgments seem to follow DRR, so that the greater relative reverberant sound energy in BK (lower DRR) seems to make the sources sound further away, even though G is greater in BK. In other words, when the first two samples had overall loudness greater than the two subsequent samples, according to the values of G, the distance judgments were based on the loudness differences. When the later presented samples actually had greater G, they were still judged as being further away as if the relative judgments were based on differences in DRR and not on overall loudness.

However, because the results showed that the order of presentation influenced the judgments, as illustrated in Fig. 5c and that the subjects did not have any possibility to make repeated comparisons between the auralizations, we suspect that this peculiarity is related to the perceptual calibration to the task as well as to the stimuli. The order effect was observed for all signal types except for the signal which was described as "musical cacophony" ('F'-signal). Remembering that the experimental design consisted of the combination of balanced between subjects design (first four samples), and a randomised design within the other signals, the order effect seems to be robust but dependent on the properties of the source material. So, this order effect seems to explain the peculiar relationship between the distance judgments and DRR and G in the first four samples. Nevertheless, the reasons why the sources in later presented auralizations were judged as being further is an open question and deserves some speculation.

For the three signals which exhibit the order effect, it is possible that repeated exposure to the same musical source material allows for a gradual attentional shift from the musical features to the room acoustic cues initially overshadowed by the attentional (musical) targets in the signals, that is, by the musicality of the signals. By informal listening it was observed that 'F'-signal was actually sparser than the other signals providing separate unconnected sound elements which possibly allowed the listeners to directly focus on the room acoustics and the relevant auditory distance cues without being distracted by the musical features. Considering that this is the first study where this systematic stimulus production procedure [27] has been employed in a listening experiment, the 'F'-signal gave results which warrant for further investigations with this approach.

The main objective of this study was to evaluate the feasibility of these auralizations to study distance perception in concert halls and the results give a good premise to continue with more detailed experiments. Although the results did not show significant interaction effect between the hall and distance, it is possible that including more seating positions, that is, more variation in distance and collecting the distance judgments with a more rigorous and systematic testing procedure could reveal more on the effect of hall acoustics on auditory distance perception.

6. CONCLUSIONS

Spatial decomposition method (SDM) based multichannel auralizations were used to study auditory distance perception in concert halls with forty naive test subjects. There were large inter-individual differences in the reported distances, and many listeners used the resolution of five meters what reflects the inaccuracy of our auditory distance perception for longer distances. Still, the results indicated that these auralizations produced significantly different distance estimates between the two halls and two positions under evaluation. However, the results failed to show an interaction effect between distance and hall acoustics. Possible reason is that only two distances and two halls were included while a more systematic variation in distances and halls might reveal such an interaction. Overall, the present experiment yielded distance estimates which are reasonably close to the real distances indicating that these auralizations may well be used to study perceived distances in concert halls, but that the following studies should use a more rigorous experimental approach.

Finally, the results showed an interesting effect of the order of presentation which was observed for all but one source material. The subjects gave increasing distance estimates with repeated exposure to the same source material independent of the auralized acoustic conditions. Such an effect is possibly related to the musicality or other properties of the source material, but the definitive reasons for this effect remain unclear and open for future work.

7. ACKNOWLEDGMENTS

The research leading to these results have received funding from the Academy of Finland, project no. 257099. Also many thanks to the great number of anonymous reviewers!

8. REFERENCES

[1] M. A. Gerzon, "The design of precisely co- incident microphone arrays for stereo and surround sound," in *Proc. of The 50th AES Convention (Abstracts)*, vol. 23, pp. 402–404, 1975.

[2] A. J. Berkhout, D. de Vries, and P. Vogel, "Acoustic control by wave field synthesis," *The Journal of the Acoustical Society of America*, vol. 93, no. 5, pp. 2764–2778, 1993.

[3] V. Pulkki, "Spatial sound reproduction with directional audio coding," *Journal of the Audio Engineering Society*, vol. 55, no. 6, pp. 503–516, 2007.

[4] S. Tervo, J. Pätynen, A. Kuusinen, and T. Lokki, "Spatial decomposition method for room impulse responses," *Journal of the Audio Engineering Society*, vol. 61, no. 1/2, pp. 17–28, 2013.

[5] T. Lokki, J. Pätynen, A. Kuusinen, and S. Tervo, "Disentangling preference ratings of concert hall acoustics using subjective sensory profiles," *The Journal of the Acoustical Society of America*, vol. 132, no. 5, pp. 3148–3161, 2012.

[6] A. Kuusinen, J. Pätynen, S. Tervo, and T. Lokki, "Relationships between preference ratings, sensory profiles, and acoustical measurements in concert halls," *The Journal of the Acoustical Society of America*, vol. 135, no. 1, pp. 239–250, 2014.

[7] L. Beranek, *Concert halls and opera houses: music, acoustics, and architecture*, Springer Science & Business Media, 2004.

[8] P. Zahorik, D. S. Brungart, and A. W. Bronkhorst, "Auditory distance perception in humans: A summary of past and present research," *Acta Acustica united with Acustica*, vol. 91, no. 3, pp. 409–420, 2005.

[9] N. Kopčo and B. G. Shinn-Cunningham, "Effect of stimulus spectrum on distance perception for nearby sourcesa)," *The Journal of the Acoustical Society of America*, vol. 130, no. 3, pp. 1530–1541, 2011.

[10] D. H. Mershon and L. E. King, "Intensity and reverberation as factors in the auditory perception of egocentric distance," *Perception & Psychophysics*, vol. 18, no. 6, pp. 409–415, 1975.

[11] S. H. Nielsen, "Auditory distance perception in different rooms," *Journal of the Audio Engineering Society*, vol. 41, no. 10, pp. 755–770, 1993.

[12] P. Zahorik, "Assessing auditory distance perception using virtual acoustics," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1832–1846, 2002.

[13] A. D. Little, D. H. Mershon, and P. H. Cox, "Spectral content as a cue to perceived auditory distance," *Perception*, vol. 21, no. 3, pp. 405–416, 1992.

[14] J. Pätynen, *A virtual symphony orchestra for studies on concert hall acoustics*, Ph.D. thesis, Aalto University School of Science, November 2011.

[15] A. Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique," in *Proc. of the 108th AES Convention*, 2000.

[16] A. Haapaniemi and T. Lokki, "Identifying concert halls from source presence vs room presence," *The Journal of the Acoustical Society of America*, vol. 135, no. 6, pp. EL311–EL317, 2014.

[17] J. Pätynen, S. Tervo, and T. Lokki, "Amplitude panning decreases spectral brightness with concert hall auralizations," in *the 55th International AES Conference: Spatial Audio*, 2014.

[18] C. J. Plack, *The sense of hearing*, Psychology Press, 2013.

[19] H. Kuttruff, *Room acoustics*, Spon Press, London and New York, 2009.

[20] S. Tervo, T. Lokki, and L. Savioja, "Maximum likelihood estimation of loudspeaker locations from room impulse responses," *Journal of the Audio Engineering Society*, vol. 59, no. 11, pp. 845–857, 2011.

[21] Ville Pulkki, "Virtual sound source positioning using vector base amplitude panning," *Journal of the Audio Engineering Society*, vol. 45, no. 6, pp. 456–466, 1997.

[22] J. Pätynen, V. Pulkki, and T. Lokki, "Anechoic recording system for symphony orchestra," *Acta Acustica united with Acustica*, vol. 94, no. 6, pp. 856–865, Dec. 2008.

[23] J. Pätynen, S. Tervo, P. W. Robinson, and T. Lokki, "Concert halls with strong lateral reflections enhance musical dynamics," *Proceedings of the National Academy of Sciences*, vol. 111, no. 12, pp. 4409–4414, 2014.

[24] ITU-R BS.1116-1, *Methods for the subjective assessment of small impairments in audio systems including multichannel sound systems*, 1997.

[25] R. Y Litovsky, H. S. Colburn, W. A. Yost, and S. J . Guzman, "The precedence effect," *The Journal of the Acoustical Society of America*, vol. 106, no. 4, pp. 1633–1654, 1999.

[26] T. Lokki, J. Pätynen, A. Kuusinen, H. Vertanen, and S. Tervo, "Concert hall acoustics assessment with individually elicited attributes," *J. Acoust. Soc. Am.*, vol. 130, no. 2, pp. 835–849, 2011.

[27] A. Kuusinen, "An anechoic audio corpus for room acoustics and related studies," in *Proc. of International Symposium of Auralization and Ambisonics, Berlin, Germany*, 2014.

[28] J. Pätynen and T. Lokki, "Directivities of symphony orchestra instruments," *Acta Acustica United with Acustica*, vol. 96, no. 1, pp. 138–167, 2010.

[29] O. Lartillot, P. Toiviainen, and T. Eerola, "A matlab toolbox for music information retrieval," in *Data analysis, machine learning and applications*, pp. 261–268. Springer, 2008.

[30] E. S. Pearson, "The analysis of variance in cases of non-normal variation," *Biometrika*, pp. 114–133, 1931.

SPATIAL AUDIO QUALITY AND USER PREFERENCE OF LISTENING SYSTEMS IN VIDEO GAMES

Joe Rees-Jones,

Audio Lab, Department of Electronics,
University of York
York, UK
jrj504@york.ac.uk

Jude Brereton,

Audio Lab, Department of Electronics,
University of York
York, UK
jude.brereton@york.ac.uk

Damian Murphy,

Audio Lab, Department of Electronics,
University of York
York, UK
damian.murphy@york.ac.uk

ABSTRACT

Spatial audio playback solutions provide video game players with ways to experience more immersive and engaging video game content. This paper aims to find whether listening systems that are able to more accurately convey spatial information are preferred by video game players, and to what extent this is true for different loudspeaker configurations whilst engaged in video game play. Results do suggest that a listening system with high perceived spatial quality is more preferred.

1. INTRODUCTION

Spatial audio hardware and software solutions are quickly becoming the state of the art in video game audio playback. Whether this is through multichannel speaker configurations, such as stereo, 7.1 surround, Dolby ATMOS [1], or binaural headphone solutions such as DTS HeadphoneX [2] and Razer Surround [3], video game players can now experience immersive audio in a variety of ways.

The dynamic nature of video gameplay lends itself well to the fundamental qualities of spatial audio, arguably more so than cinema or music. A vast majority of what is perceived by the player, especially in first and third-person content occurs off-screen, either behind or to the side of the camera view [4]. This is where effective use of spatial audio can be especially useful, where spatialised sound cues can be used to warn players of impending threats, potentially giving a competitive advantage in multiplayer situations [5, 6]. Sound cues can also be used to reduce the information presented on screen, for example on the heads up display (HUD), to the extent where it is possible to develop audio only games [7].

This paper describes an experiment to ascertain whether the use of spatial audio in game audio playback rendering solutions, and in the games development process, has a positive impact on video game player quality of experience. Appropriate game content will be sourced and, in playing through a scene, participants will subjectively rate the spatial sound quality and preference of some typical game audio playback systems. It is hypothesised that an audio system with subjectively high spatial sound quality will be preferred to those with lower capabilities. Section 2 outlines the process of choosing appropriate video game content for the purposes of experimentation, and the experimental method. The spatial attributes used to assess said content are also given. Results and analysis are given in Section 3.

2. EXPERIMENTAL DESIGN

2.1. Content Selection

Video game content appropriate for the purposes of this study was chosen according to a specific set of criteria divided into two main categories:

Audio:

- Appropriate and effective use of spatial audio.
- 7.1 surround sound compatibility.
- Third-party critical acclaim for use of audio.

Gameplay:

- Repeatability.
- A limited number of 'fail-states', meaning it should be difficult for a player to fail as a result of death or not completing an objective.
- The ability to easily restart with little or no back-tracking.
- Simple controls.
- An easy to follow, preferably linear path.

Extensive play testing of a wide range of content revealed Naughty Dog's *The Last of Us: Remastered*, (on the Playstation 4) to be the most appropriate video game for the purposes of this study as it fulfills all the set criteria. The game is currently the most awarded in recorded history [8] and has won a multitude of audio related awards including a BAFTA for Audio Achievement [9] and G.A.N.G (Game Audio Network Guild) awards for Audio of the Year, Sound Design of the Year and Best Audio Mix in 2014 [10], showing considerable acclaim in the industry.

Not only does the spatial mix aesthetically enrich the virtual environments and settings presented to the player through effective use of spatial audio, but it is also essential to the gameplay. The audio in itself can be considered as one of the core gameplay mechanics, where players are able to locate potential threats purely through listening. Playing the game through a multi-channel audio system seems to significantly increase chances of survival where players are able to gain a tactical advantage over enemies by listening for their whereabouts.

The introductory sequence of *The Last of Us: Remastered* is especially appropriate for the purposes of experimentation. The player is required to follow a linear path, reducing the likelihood for inexperienced participants to get lost. The majority of the audio events are scripted and won't trigger until the player encounters a particular section, ensuring similar auditory experiences on multiple play-throughs. The number of fail-states in the sequence is low, where even if the player does fail they are able to quickly continue the play-through with minimal consequence.

It has also been shown that giving participants' too much control in an experiment such as this will have an adverse effect on their ability to rate sound quality [11], making this particular section in *The Last of Us: Remastered* ideal - all that is required of the player is to move their character and follow clear, on-screen button prompts for more complex interactions.

2.2. Playback Scenarios

Three playback listening conditions were used in the experiment: mono, stereo and 7.1 surround-sound. These three conditions were chosen as they best represent current audio solutions available to the average gamer. Mono was used for the least spatially capable listening condition as it is near impossible to convey accurate spatial information over one loudspeaker. Stereo playback permits sound to be positioned horizontally left and right giving some amount of spatialisation. The 7.1 surround-sound loudspeaker arrangement is the best spatial solution of the three as it allows for full horizontal spatialisation, enveloping the player in sound and also allows for fairly accurate sound source localisation. In consumer gaming, it is also currently the limit as to what is offered in terms of surround-sound channel count.

In order to ensure participants are not biased in their responses by being able to detect the number of speakers actually active in the playback system, it was decided that both the mono and stereo listening conditions would make use of all of the 8 speakers to be used in the 7.1 surround-sound configuration. It is possible to set the AV receiver used in the tests, which handles all the audio and visual information from the Playstaion 4 and outputs it to the appropriate listening and viewing apparatus, to *Full Mono* where a single mono mix-down of the game audio stream is sent to all connected speakers. Similarly the listening mode *All Ch. Stereo* takes the stereo game audio stream (which may well be down-mixed automatically from a higher channel count by either the game engine or the receiver) and sends audio for the left channel to all three speakers to the left of the listener and the same for the right. The centre channel outputs a sum of the left and right channels. Typically it would not be expected that a listener would experience mono or stereo material in this fashion. A subwoofer was included in all three listening conditions for low frequency effects. Content was presented to participants in this way to make the transition between listening conditions less obvious. It can be expected that the effectiveness of the game's spatial audio would still be limited since the physical limitations of these listening modes would not allow for the exploitation of critical spatial information, especially in regards to sound cues from behind the listener.

To assess each playback scenario participants were split into three groups (A, B and C), each of which was exposed to two of the possible three listening conditions:

- Group A - Mono – Stereo.
- Group B - Mono – Surround.
- Group C - Stereo – Surround.

Due to the length of the chosen scene from *The Last of Us: Remastered* (approximately 12 minutes) it was decided participants should only be exposed to two of the three listening conditions, significantly reducing the amount of time required of each participant and also the risk of any learning bias that may be present after three play-throughs. A third play-through may also confuse subjects' judgement and therefore affect preference ratings. Participant identities were anonymised by assigning each a unique number and the order of exposure to the two listening conditions was randomised.

2.3. Spatial Attributes

The list of attributes chosen to rate the spatial sound quality of the game audio content is as follows:

- Depth.
- Distance.
- Sound Source Localisation.
- Sound Source Definition.
- Stability.
- Envelopment of Reverberation.
- Source Width.

These attributes and their accompanying descriptors (for the benefit of participants) were sourced from several different spatial attribute lists suggested by the SAQI (Spatial Audio Quality Inventory) [12], ITU recommendation BS.1284-1 [13], and other publications [14-16].
These spatial attributes were rated by participants on a 5-point quality scale ranging from Bad (1), Poor (2), Fair (3), Good (4) to Excellent (5) as suggested by ITU recommendation BS.1284-1 [13].

After completion of both play-throughs preference was measured on a 7-point paired comparison scale where participants were able to state which of the two listening conditions they were personally exposed to they preferred and to what extent. The scale ranged from Strong Preference for A (3), Preference for A (2), Slight Preference for A (1), No Preference (0), Slight Preference for B (1), Preference for B (2) and Strong Preference for B (3) [17].

Using paired comparison of this type it can be assumed that the preference rating for stimuli A will be the same value, but of opposite magnitude, to the rating given to stimuli B and vice-versa [18]. For example if surround-sound (A) were being compared to mono (B) and received a rating of 3 for strong preference, mono would then receive a -3 as it can be presumed it is strongly not preferred.

2.4. Apparatus

The Last of Us: Remastered was played on a Sony Playstation 4 connected via HDMI to an Onkyo TX-NR838 AV Receiver. Six Genelec 8040As, one Genelec 8040B (centre channel) and a Genelec 7060B Active Subwoofer were arranged according to the ITU specification [19] for 7.1 surround-sound listening and connected to the appropriate audio outputs of the receiver. The sound level of each speaker was measured with an SPL meter and test tone to ensure all speakers were at a consistent level. Overall volume was controlled by the receiver and set to a comfortable level for the duration of the experiment.

An Optoma HD200X projector was used for visual feedback and connected via HDMI to the main video out of the receiver. An office chair was positioned in the sweet spot for participants to be seated whilst partaking in the experiment. The listening room was surrounded by a thick absorbing drape with foam acoustic panelling above the listener. The extra speakers above, below and to the side of listener, shown in Figure 2, that do not conform to the 7.1 surround sound speaker configuration were not active.

3. RESULTS

In total 21 participants took part in the experiment (17 male and 4 female). 20 of these participants were aged 20-35 with one over 50. In regards to gameplay experience 5 of the participants play games on a daily basis, 10 weekly, 3 monthly, 2 yearly and 1 never. Participants were split evenly into 3 groups (A, B and C) corresponding to the 3 sets of listening conditions giving 7 per group and 14 for each listening condition.

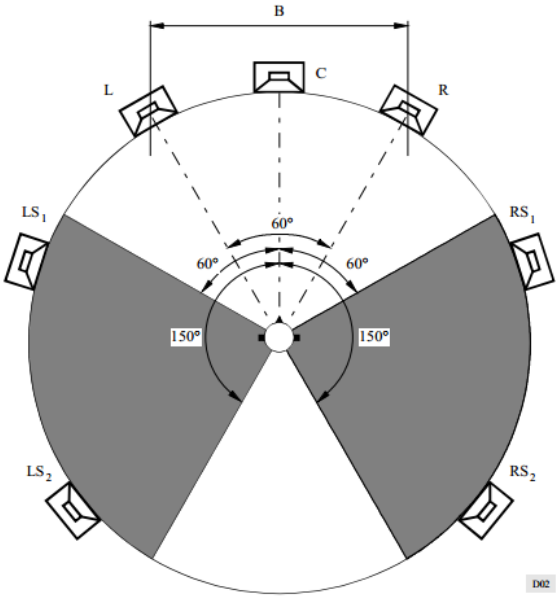


Figure 1: *The full experimental surround-sound loudspeaker configuration used, conforming to the 7.1 standard - from [19].*

Participants were first given an experiment pack containing an information sheet with consent form, event timeline, control scheme, spatial attribute list with descriptors and questionnaire. Spatial attributes were further explained to participants by the experimenter upon request. Subtitles were turned on and the vertical axis for the game's camera control inverted if requested by the participant.

Depending on group allocation, and unknown to the participant, one of two listening conditions was randomly chosen and set on the receiver by the experimenter. After completing the first play-through participants were asked to exit the listening space and fill in the first spatial quality questionnaire whilst the experimenter reset the game and set the receiver to the second listening condition. Upon

completion of the second play-through participants filled in the remainder of the questionnaire. The experimenter stayed in the room throughout the test to offer help in navigating the game if needed.

Comments regarding the experiment were also noted by participants.

Figure 2: *The listening room used throughout the experiment. Loudspeakers not part of the defined 7.1 playback system were non-active during the test.*

3.1. Analysis

Before analysis it was decided the stability attribute would be removed as one participant did not give a score due to a lack of understanding. It was felt it would be less detrimental to the experiment if the attribute was removed from the analysis process rather than omitting a participant's entire set of results. Individual spatial attribute scores for each participant were first normalised by generating z-scores [20] and then averaged by participant to give overall spatial quality ratings for each listening condition. All statistical analysis was conducted in MATLAB.

Independent t-tests revealed there to be a statistically significant difference between the spatial quality ratings for all three listening conditions given by groups A, B and C at $p<.05$. This suggests that perceived spatial quality is influenced by the listening condition a participant is exposed to. Results from the t-tests are given in Table 1.

Table 1: *Table of t-test results across all listening conditions and groups. A value of '1' represents a rejection of the null hypothesis at $p<.05$.*

	Mono (a)	Stereo (a)	Surround (b)
Mono (b)	X	1	1
Stereo (c)	1	X	1
Surround (c)	1	1	X

One-way ANOVA tests revealed there was no statistically significant difference between spatial quality scores given by different groups assessing the mono and surround listening conditions at $p<.05$. For mono groups A and B [$F(1,6) = 4.34$, $p = 0.06$], and surround groups B and C [$F(1,6) = 3.69$, $p = 0.08$]. These results suggest that individuals from different sample groups will have a similar subjective opinion regarding spatial sound quality of the same listening condition when engaged in playing a video game. However, this cannot be said for the stereo condition where [$F(1,6) = 16.22$, $p = 0.002$]. Through further analysis it was shown that the spatial quality scores given for stereo when compared to mono were significantly

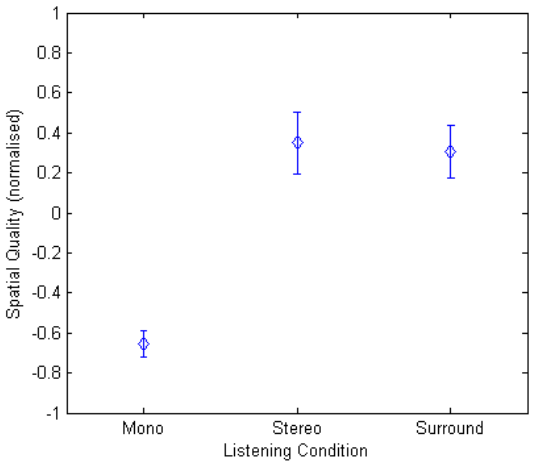


Figure 3: *The mean of the normalised spatial quality scores for the mono, stereo and 7.1 surround-sound listening conditions with standard error.*

higher than those when compared to 7.1 surround-sound.

A comparison between the group means of the three listening conditions indicates a statistically significant difference between the subjective spatial quality of mono, stereo and 7.1 surround sound. The mean spatial quality scores for each listening condition are presented in Figure 3. Analysis by one-way ANOVA determined [$F(2,13) = 21.17$, $p<0.001$] at the $p<.05$ level, suggesting that listening condition has a significant effect on subjective impression of spatial quality. An honest significance difference (HSD) post-hoc test revealed that both stereo and 7.1 surround-sound total spatial quality scores were considerably higher than those for mono. Spatial quality scores for stereo and surround are however not significantly different to one another at $p<.05$.

It was anticipated that mono would receive the lowest spatial quality score of the three listening conditions since the physical capabilities of a single, down-mixed audio signal make conveying accurate spatial information difficult. However, it was also expected that 7.1 surround-sound would be perceived by individuals to be the more

spatially capable system which, according to the results, is not the case.

Analysis of preference ratings given to the same listening condition by different groups revealed there to be no statistical difference between mono groups A and B, [$F(1,6) = 0.96$, $p = 0.34$] and surround groups B and C [$F(1,6) = 3.43$, $p = 0.09$]. There was however a much more noticeable difference between the preference ratings given by groups A and C for the stereo condition where [$F(1,6) = 8.65$, $p = 0.01$] at $p<.05$. HSD output revealed stereo received higher preference ratings from group A (Mono – Stereo) and lower ratings from group C (Stereo – Surround).

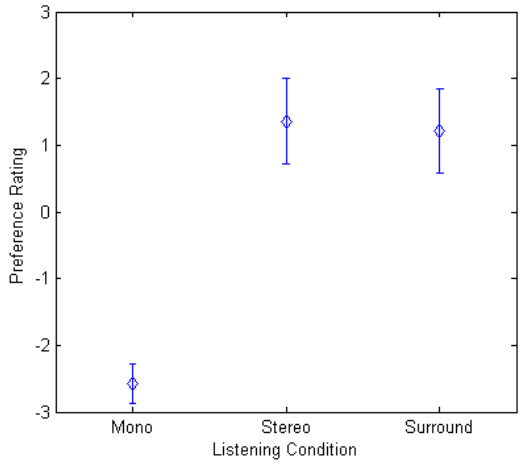


Figure 4: *The mean preference ratings for the mono, stereo and 7.1 surround-sound listening conditions with the standard error.*

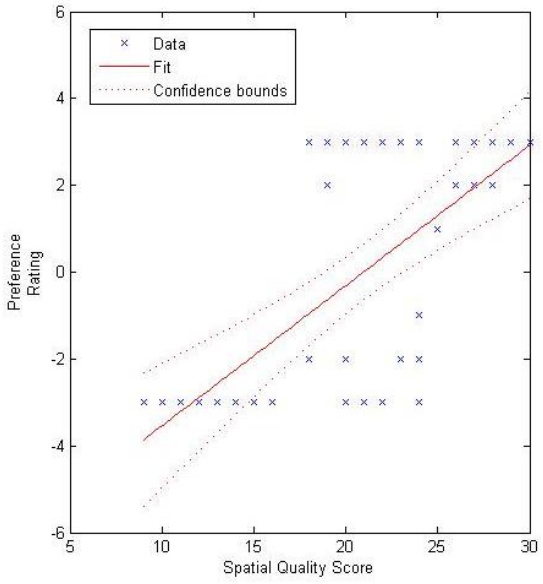


Figure 5: *Regression model of spatial quality scores vs preference ratings.*

Much like the spatial quality scores given for stereo, this implies that to some extent preference is being affected by the combination of conditions a participant is exposed to.

Analysis of preference ratings grouped by listening condition determined that [$F(2,13) = 16.63$, $p<0.001$] at $p<.05$, suggesting it has a significant effect on preference. The mean preference ratings for each listening condition are given in Figure 4. A Tukey's HSD post-hoc test of the preference ratings yielded a similar result to that of spatial quality, where mono was preferred the least but stereo and surround received statistically similar ratings at $p<.05$.

Again this result was not expected as it was hypothesised that 7.1 surround-sound would be more preferred than both the mono and stereo listening conditions. However, these results do show that as subjective impression of spatial quality increases, so does preference. This is illustrated by the regression model of non-normalised data in Figure 5 where the line of best fit shows how preference (Y axis) for a listening condition will increase as its spatial quality (X axis) improves.

3.2. Discussion

From the analysis of results it can be seen that a listening condition that is subjectively perceived to be of high spatial quality will be more preferred when engaged in video game play. This is especially clear in regards to the mono listening condition which received the lowest spatial quality scores and was also the least preferred.

However, the analysis also suggests that stereo is as spatially capable as 7.1 surround sound and equally as preferred. This outcome was not expected and is most likely due to the fact that all-channel stereo was used in place of traditional 2-channel frontal stereo in an attempt to make the transition between listening conditions less obvious. Stereo allows for some amount of spatialisation between the left and right channels, which will have been exaggerated by the use of all-channel stereo. This extreme panning to the left and right may have been perceived by listeners to be more spatial than regular stereo, resulting in a more positive opinion of the condition. It is also important to note that full mono and all-channel stereo are naturally going to feel more enveloping than their more traditional one and two-channel counterparts, since audio will be perceived to be all around the listener, even if the spatial information is not accurate.

The game's onscreen visuals coupled with the potential envelopment of full mono and all-channel stereo, may have also played a part in convincing participants into thinking they were experiencing a more spatialised soundscape. Studies have shown that visual stimuli can have significant effects on an individual's ability to perceive spatial aspects of a soundscape, especially relating to sound source localization [21, 22]. This is apparent in some of the participants' comments. After it was revealed to a participant which listening conditions they were personally exposed to it was implied that they were able to

localise the sounds of two passing police cars even in the full mono environment. Another stated they could hear a helicopter passing over their head in both the stereo and mono conditions even though loudspeakers to simulate elevation were not included in this experiment.

It was decided frontal two-channel stereo should be added as a fourth listening condition and directly compared to 7.1 surround sound in a second set of tests to find if the use of all-channel stereo influenced the original results. It was hypothesised that stereo would receive a lower spatial rating than 7.1 surround-sound and be less preferred.

3.3. Experimental Procedure

A fourth group of participants (D) was assembled to assess the spatial sound quality and give preference ratings for normal two-channel frontal stereo compared to 7.1 surround-sound. Participants were subject to the same procedure as the previous experiment. 6 new participants took part (4 male, 2 female) aged between 18 and 30.

3.4. Analysis

Spatial quality scores were normalised and averaged by participant, then analysed by one-way ANOVA. Analysis of the spatial quality scores for surround-sound and full frontal stereo determined [$F(1,5) = 20.59$, $p = 0.001$] at $p<0.05$. As expected the spatial quality of 7.1 surround-sound was significantly higher than that of traditional, two-channel stereo. The mean spatial quality scores for both conditions are illustrated in Figure 6.

Furthermore, analysis of the preference ratings given to frontal stereo and surround revealed [$F(1,5) = 320$, $p<0.001$] at $p<0.05$. By inspecting of the mean preference ratings presented in Figure 7 it can be seen that there is a clear difference between the two conditions.

3.5. Discussion

As was hypothesised, 7.1 surround-sound received higher spatial quality scores than the regular stereo listening environment and as a result was more preferred by participants. This further reinforces the idea that, if given the choice, players will prefer to experience their games through more spatially able listening environments. It can also be said that the unexpected results previously observed were most likely down to the use of full channel stereo, rather than regular two-channel frontal stereo, as a playback condition, giving a more realistic insight into playback solutions available to the average gamer.

However, a crucial question still remains unanswered: how perceivable are the differences between 7.1 surround-sound and all-channel stereo and where does the threshold of discrepancy lie?

4. CONCLUSIONS

Results from these experiments do suggest that video game players have a more preferable experience when listening to their games through spatial audio speaker systems. The first set of tests indicate that stereo and 7.1 surround-sound gameplay demonstrated clear perceptual spatial quality improvement in comparison to mono, thus being more preferred. The subjective similarity between stereo and 7.1 was not expected and shows no significant difference in player preference, however further tests using regular, two-channel frontal stereo suggest that this was due to the experimental conditions used in the original experiment.

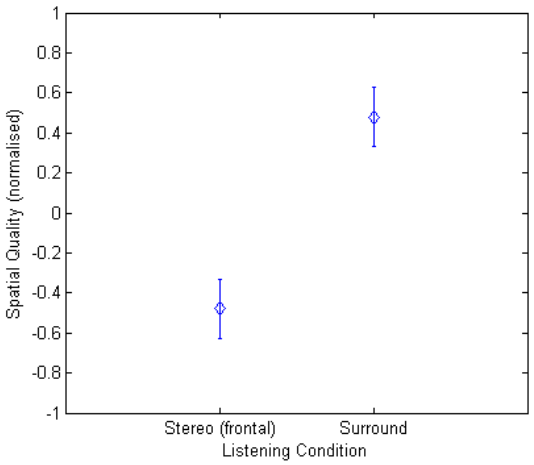


Figure 6: *The means of normalised spatial quality scores for the frontal stereo and 7.1 surround sound listening conditions with the standard error.*

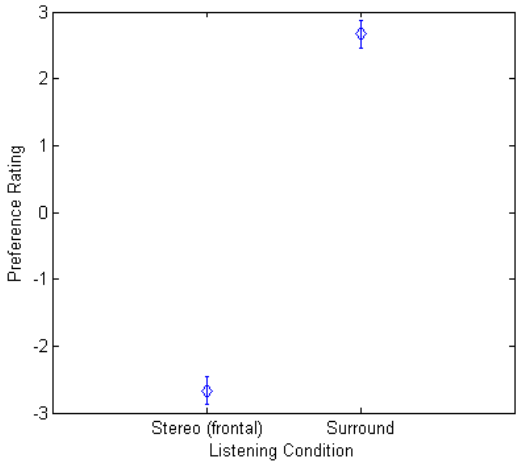


Figure 7: *The mean preference ratings for the frontal stereo and 7.1 surround sound listening conditions with the standard error.*

5. FURTHER WORK

Even where 7.1 surround-sound demonstrates improvement, a loudspeaker setup of this caliber isn't always the most practical configuration, especially for gamers where price, free space and living conditions have to be taken into account. Virtualising surround-sound headphones are the current alternative available to video game players on both next generation consoles and PC. Experiments designed to not only assess the spatial quality of such solutions, but also test whether players actually get better at their games as a result of using them would be highly beneficial.

The Last of Us: Remastered falls very firmly into the action-adventure genre so it can easily be seen how games like this and of similar a genre (such as FPS (first person shooter) and horror) could benefit from effective spatial audio. However, it is not clear how spatial audio may benefit players of less obvious genres such as puzzle games or side-scrollers. It would be interesting to see how spatial audio could be used in a more creative sense to provide auditory feedback in these types of games, rather than to create 'lifelike' audio simulations as is often the case.

One of the drawbacks of the experiments presented in this paper is the reliance on participants to be able to accurately rate the spatial quality of the listening conditions presented to them. There are a number of factors that can affect the reliability of such subjective measures such as:

- Participants' ability to recall their opinions on specific elements of the in-game soundscape after extensive playing time.
- The clarity of the spatial attributes to be rated.
- The assumption that playing a video game is not going to distract from the core purposes of the test.

An experiment is currently in development to find whether the introduction of spatial audio in a gaming environment has an effect on a player's physically measureable emotional output. Through skin conductance and heart rate measures it is possible to quantify an individual's arousal and valence, which when collectively observed can give an idea as to their current emotional state [23]. A positive outcome from an experiment such as this could add considerable weight to data gathered by subjective methods.

REFERENCES

[1] Dolby.com, 'Dolby Atmos', 2015. [Online]. Available: http://www.dolby.com/in/en/brands/dolby-atmos.html. [Accessed: 28- May- 2015].

[2] Headphonex.com, 'DTS HeadphoneX', 2015. [Online]. Available: http://www.headphonex.com/. [Accessed: 28- May- 2015].

[3] Razer, 'Razer Surround Personalized 7.1 Gaming Audio Software', 2015. [Online]. Available: http://www.razerzone.com/gb-en/surround. [Accessed: 28- May- 2015].

[4] Collins, Karen. *Playing with sound: A theory of interacting with sound and music in video games*. MIT Press, pp. 47, 2013.

[5] Richardson, John, Gorbman, Claudia, and Vernallis, Carol, eds. *The Oxford Handbook of New Audiovisual Aesthetics*. Oxford University Press, pp. 585-602, 2013.

[6] Kerins, Mark. *Beyond Dolby (stereo): cinema in the digital sound age*. Indiana University Press, pp. 322-324, 2010.

[7] Oldenburg, Aaron. "Sonic mechanics: Audio as gameplay." *The International Journal of Computer Game Research [Online journal]* 13, 2013.

[8] G. Nelva, 'Naughty Dog's The Last of Us Is the Most Awarded Game in Recorded History by Critics | DualShockers', Dualshockers.com, 2015. [Online]. Available: http://www.dualshockers.com/2014/01/24/the-last-of-us-is-the-most-awarded-game-in-history-by-critics/. [Accessed: 28- May- 2015].

[9] Awards.bafta.org, 'Games in 2014 | BAFTA Awards', 2015. [Online]. Available: http://awards.bafta.org/award/2014/games. [Accessed: 28- May- 2015].

[10] Audiogang.org, '2014 Awards', 2015. [Online]. Available: http://www.audiogang.org/awards/2014-awards/. [Accessed: 28- May- 2015].

[11] Zielinski, Slawomir K., et al. "Computer games and multichannel audio quality-The effect of division of attention between auditory and visual modalities."*Audio Engineering Society Conference: 24th International Conference: Multichannel Audio, The New Reality*. Audio Engineering Society, 2003.

[12] Lindau, Alexander, et al. "A Spatial Audio Quality Inventory (SAQI)." *Acta Acustica united with Acustica* 100.5, pp. 984-994, 2014.

[13] Recommendation, I. T. U. "General methods for the subjective assessment of sound quality." *ITU-R BS* (2003): 1284-1.

[14] Rumsey, Francis. *Spatial audio*. Taylor & Francis, 2001.

[15] Rumsey, Francis. "Spatial quality evaluation for reproduced sound: Terminology, meaning, and a scene-based paradigm." *Journal of the Audio Engineering Society* 50.9, pp. 651-666, 2002.

[16] Bech, Søren, and Zacharov, Nick. *Perceptual audio evaluation-Theory, method and application*. John Wiley & Sons, 2007.

[17] Scheffé, Henry. "An analysis of variance for paired comparisons." *Journal of the American Statistical Association* 47.259, pp. 381-400, 1952.

[18] David, Herbert Aron. *The method of paired comparisons*. Vol. 12. London, 1963.

[19] Recommendation, CCIR Draft New. "Multi-channel stereophonic sound system with and without accompanying picture." *International Telecommunication Union, BS*: 775-1.

[20] Jordan, Dominic William, and Peter, Smith. *Mathematical techniques*. Oxford University Press, pp. 898-899, 1997.

[21] Moore, Brian CJ, ed. *An introduction to the psychology of hearing*. Brill, 2012.

[22] Werner, Stephan, Liebetrau, Judith, and Sporer, Thomas. "Audio-visual discrepancy and the influence on vertical sound source localization." *Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on*. IEEE, 2012.

[23] Russell, James A. "A circumplex model of affect." *Journal of personality and social psychology* 39.6 1161, 1980.

FREQUENCY ESTIMATION OF THE FIRST PINNA NOTCH IN HEAD-RELATED TRANSFER FUNCTIONS WITH A LINEAR ANTHROPOMETRIC MODEL

Simone Spagnol,
Faculty of Ind. Eng., Mech. Eng. and Computer Science,
University of Iceland
Reykjavík, Iceland
spagnols@hi.is

Federico Avanzini,
Department of Information Engineering,
University of Padova
Padova, Italy
avanzini@dei.unipd.it

ABSTRACT

The relation between anthropometric parameters and Head-Related Transfer Function (HRTF) features, especially those due to the pinna, are not fully understood yet. In this paper we apply signal processing techniques to extract the frequencies of the main pinna notches (known as N_1, N_2, and N_3) in the frontal part of the median plane and build a model relating them to 13 different anthropometric parameters of the pinna, some of which depend on the elevation angle of the sound source. Results show that while the considered anthropometric parameters are not able to approximate with sufficient accuracy neither the N_2 nor the N_3 frequency, eight of them are sufficient for modeling the frequency of N_1 within a psychoacoustically acceptable margin of error. In particular, distances between the ear canal and the outer helix border are the most important parameters for predicting N_1.

1. INTRODUCTION

Most of the current binaural sound rendering techniques rely on the use of Head-Related Transfer Functions (HRTFs), i.e. filters that capture the acoustic effects of the human head [1]. HRTFs allow loyal simulation of the signal that arrives at the entrance of the ear canal as a function of the sound source spatial position. The classic solution best approximating free-field listening conditions involves the use of individual HRTFs measured on the listener himself with the addition of head tracking and artificial reverberation [2]. However, obtaining personal HRTF data for a vast number of users is only possible with expensive equipment and invasive recording procedures [3]. This is the reason why non-individual HRTFs, acoustically measured on anthropomorphic mannequins, are often preferred in practice. The drawback with non-individual HRTFs is that these likely never match with the listener's unique anthropometry, and especially the outer ear, resulting in frequent localization errors such as front/back reversals, elevation angle misperception, and inside-the-head localization [4].

In order to efficiently face such issues, several techniques for synthetic HRTF design have been proposed during the last two decades. In the authors' opinion, the most attractive is represented by structural HRTF models [5]. According to this approach, the most important effects involved in spatial sound perception (acoustic delays and shadowing due to head diffraction, reflections on pinna contours and shoulders, and so on) are isolated and modeled separately with a corresponding filtering element. The advantages of such an approach over alternative binaural rendering techniques are twofold:

1. adaptability to a specific subject, based on anthropometric quantities (head radius, pinna shape, shoulder width, and so on);
2. computational efficiency, as models are structured in smaller blocks each simulating one physical effect, allowing low-cost implementation and low-latency reproduction on many devices.

However, previous studies on the relation between acoustic effects and anthropometry - including applications of anthropometric regression methods to measured HRTF data [6, 7, 8, 9] - have produced mixed results, highlighting that many of these relations are not fully understood yet.

We can identify two reasons why these studies fail at explaining such relations. First, they typically do not take into account prior knowledge of the structural components that are responsible for localization cues, blindly applying classical machine learning techniques to long anthropometric feature vectors including irrelevant parameters that only have the effect of increasing clutter. Second, the former studies consider the whole HRTF or a dimensionally reduced version of it (e.g. via Principal Component Analysis) as the set of target variables, without applying any substantial preprocessing step in order to extract local HRTF features. In particular, it is known that local minima (notches) and maxima (peaks) in the transfer function are salient for detecting the most "individual" dimension, i.e. the elevation of a sound source [10].

This paper's main objective is to explore the relationship between the center frequencies of the three main notches in a set of frontal median-plane HRTFs and anthropometric parameters under the form of global pinna measurements (e.g. pinna height, concha width) as well as measurements that vary with the elevation angle of the sound source (i.e. distances between the ear canal and pinna edges). Indeed, the starting point of this paper (briefly reported in Section 2) is a previous work [11] that highlights how in median-plane HRTFs frequencies of the most prominent spectral minima are tightly related to the shape of the subject's pinna, and in particular to the above cited distances. Section 3 describes the methods for feature extraction and the anthropometric regression model, whose results are reported and discussed in Section 4. Section 5 concludes the paper.

2. A STRUCTURAL PINNA MODEL

The most relevant differences between the HRTFs of two subjects are due to different pinna features (shape, size, and orientation). The pinna has a fundamental role in shaping HRTFs thanks to two main acoustic phenomena, i.e., reflections and resonances. Consequently, the HRTF shows a sequence of peaks centered around the

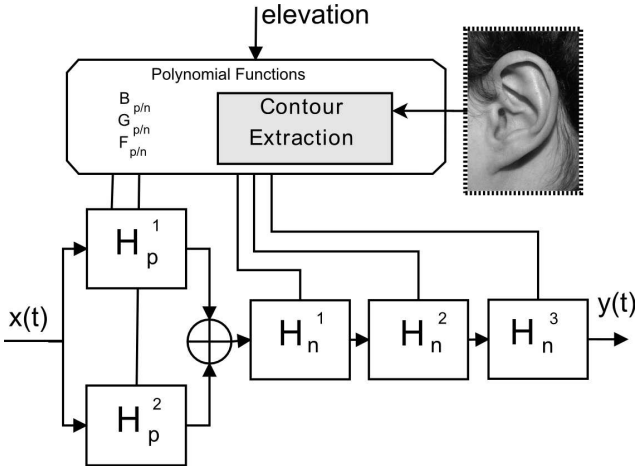


Figure 1: Schematic representation of the structural pinna model.

resonant frequencies and a sequence of notches located at all those frequencies where destructive interference between direct and reflected waves occurs. The spectral location of peaks and notches represents a pivotal cue to the characterization of the sound source's spatial position, in particular of its elevation [10].

As a matter of fact, Iida *et al.* [12] showed that a parametric HRTF recomposed using only the first peak in the HRTF spectrum coupled with the first two notches yields almost the same localization accuracy as the corresponding measured HRTF. Additional evidence in support of the lowest-frequency notches' relevance is given by Moore [13], who states that the threshold for perceiving a shift in the central frequency of a spectral notch is consistent with the localization blur (i.e., the angular threshold for detecting changes in the direction of a sound source) on the median plane. Also, Hebrank and Wright [14] judge increasing frontal elevation apparently cued by the increasing central frequency of a notch.

Previous literature suggests a number of solutions to synthetic modeling of the pinna-related component, known as Pinna-Related Transfer Function (PRTF). However, these models suffer from evident limits, e.g., the validity in an over restricted spatial region [15], and/or the absence of an explicit parametrization on the listener's anthropometry [16]. The authors propose a structural PRTF model composed of two filter blocks [11]. As Fig. 1 details, the first block (*resonant* block) includes two second-order peak filters placed in parallel, while the second block (*reflective* block) includes three second-order notch filters placed in series.

Similarly to previous works [17], the authors also studied the relation between notch frequencies in PRTFs and pinna geometry. A *ray-tracing* procedure on pinna images was exploited to map reflection points at a given distance from the reference ear-canal point, each of which is directly derived from a single notch frequency. The authors conclude that the use of negative reflection coefficients is crucial in determining notch frequencies. Therefore, the relation between notch frequency and reflection point-ear canal distance can be approximated by the following simple equation,

$$D_i(\phi) = \frac{c}{2F_i(\phi)}, \tag{1}$$

where c is the speed of sound, ϕ is the PRTF elevation angle, F_i is the center frequency of the i-th notch N_i, and D_i is the distance between the corresponding reflection point and the ear-canal point. Reflection points obtained from Eq. (1) were mapped on pinna images of a pool of experimental subjects, resulting in a close correspondence between reflection points and the three main contours, i.e. helix border, antihelix/concha inner wall, and concha border.

As a consequence, if we have an image of the pinna we can extract the above three contours, transform them into a sequence of polar coordinate pairs $(D_i(\phi), \phi)$ with respect to the ear-canal point, and derive from Eq. (1) notch frequencies for every desired elevation ϕ and for each of the three contours. The only independent parameter used in the model is indeed sound source elevation, which drives the evaluation of three polynomial functions (F_n^i, $i = 1, 2, 3$) that interpolate the obtained notch frequencies for a certain sampling step $\Delta\phi$. For what concerns the bandwidth and gain of notches as parameters, no clear relation with the pinna shape was found. The authors previously approximated these parameters, as well as resonance parameters, using average values from a population of subjects [18].

3. METHODS

The raw dataset consists of measured Head-Related Impulse Responses (HRIRs) from the CIPIC database [19], a public-domain database of high spatial resolution HRIRs measured at 1250 directions for 45 different subjects. Since this work involves subject anthropometry in the form of both numeric data (anthropometric parameters included in the CIPIC database) and a picture of their left or right pinna, we consider the 33 of them for which both are available for our analysis.

Consistently with the CIPIC database spatial grid, we take the interaural polar coordinate system as reference. We restrict our analysis to the frontal half of the median plane (azimuth angle $\theta = 0^\circ$), with the elevation angle ϕ varying from $\phi = -45^\circ$ to $\phi = 45^\circ$ at 5.625-degree steps (17 HRIRs per subject). We choose to consider the median plane because relative azimuthal variations up to at least $\Delta\theta = 30^\circ$ at fixed elevation cause very slight spectral changes in the pinna-related component of the HRTF [20], hence the model can be generalized to a wider set of azimuth values. Elevations higher than 45° were discarded because of the general lack of spectral notches in the corresponding HRTFs [21].

3.1. Notch frequency extraction

In order to get the relevant notch frequencies, we apply the ad-hoc signal processing algorithm by Raykar *et al.* [20] to each HRIR. Briefly, the algorithm computes the autocorrelation function of the linear prediction residual and extracts notch frequencies as the local minima of its group-delay function falling beyond a fixed threshold (heuristically set to -0.5 samples). Then, for each available elevation ϕ, the extracted notches are grouped in frequency tracks along adjacent elevations through the McAulay-Quatieri partial tracking algorithm [22], originally used to group sinusoidal partials along consecutive temporal windows according to their spectral location. The matching interval for the tracking procedure is set to $\Delta = 1$ kHz. The very same procedure for notch extraction and grouping was successfully used in a previous work [23].

Only tracks with 3 notches at least are preserved. If more than three tracks satisfying such a requirement are available, only the three longest tracks are considered and each frequency point labeled F_1, F_2, and F_3 in increasing order of average frequency. In those cases where a subject lacks a notch track (9 cases out of

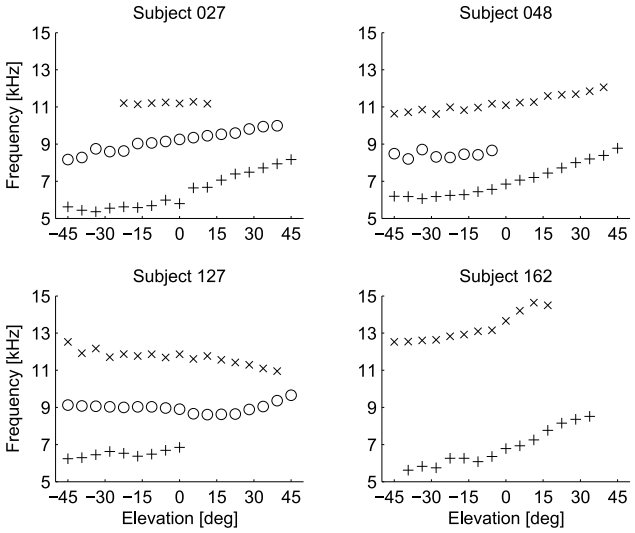


Figure 2: *Extracted notch tracks of four representative CIPIC subjects (N1:+, N2:∘, N3:×).*

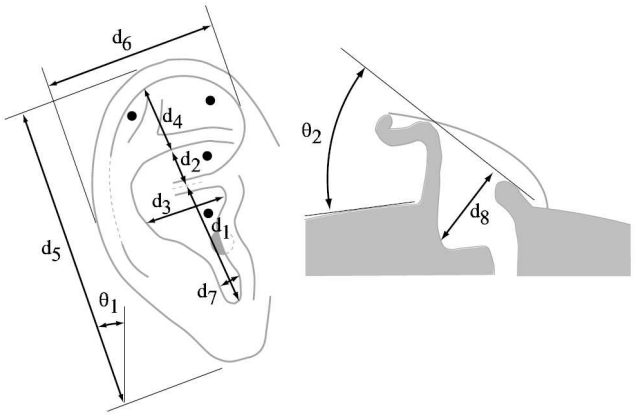


Figure 3: *The* 10 *anthropometric parameters of the pinna included in the CIPIC database (figure reproduced from [19]).*

33), labels are assigned according to the closest notch track frequency median among all subjects with three tracks. Figure 2 reports notch tracks of four representative subjects. Overall, the notch frequency extraction step yields 367 different observations for F_1, 401 for F_2, and 303 for F_3. Given the limited amount of data and the exploratory nature of this work, all observations will be used to train the following regression model, and 10-fold cross-validation will be performed to give an estimate of model fit.

3.2. Anthropometric feature extraction

Thirty-seven global anthropometric measurements for each one of the 33 considered subjects are available in the CIPIC database, 17 for the head and torso and 10 for each pinna. In this work, because of the focus on pinna notches, we consider the single pinna parameters only, which are reported in Fig. 3 and described in Table 1 for convenience. Table 1 also reports pairwise correlation values calculated on all available CIPIC subjects. Notice the over-

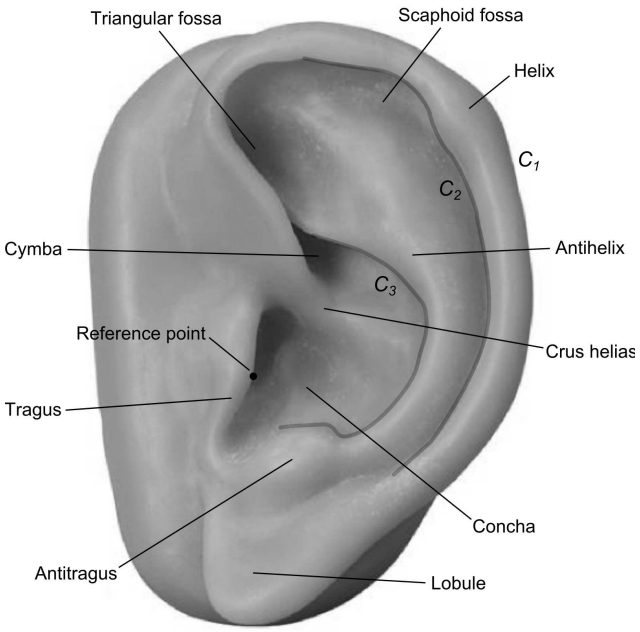


Figure 4: *Anatomy of the pinna and the three extracted pinna contours C_1, C_2, and C_3.*

all weak correlation values, apart from some cases where measurements overlap (e.g. $d_4 - d_5$, $d_3 - d_7$), which denotes an acceptable degree of orthogonality among the considered parameters.

In addition, elevation-dependent parameters strictly related to the ray-tracing rationale outlined in the previous Section are extracted according to the following procedure. The three contours corresponding to the outer helix border, the inner helix border, and the concha border/antitragus (see Fig. 4 for visual evidence of these contours) are traced by hand with the help of a pen tablet and stored as sequences of pixels.[1] Then, the point of maximum protuberance of the tragus is chosen as the reference ear-canal point for the computation of distance parameters. For each considered elevation angle $\phi \in [-45, 45]$, distances in pixels between the reference point and the point intersecting each traced pinna contour along the ray originating from the reference point with slope $-\phi$ are finally converted to centimeters based on the measuring tape reported in each picture close to the pinna and stored as $r_i(\phi)$, where $i \in \{1, 2, 3\}$ refers to the associated contour C_i.

The three parameters r_1, r_2, and r_3 are, as one may expect due to the roughly elliptic shape of the pinna, highly correlated. In particular, the correlation between r_1 and r_2 is 0.95 and these both correlate 0.75 with r_3. The very high correlation value between r_1 and r_2 can also be explained by the fact they both refer to the helix.

[1] Notice that these three contours do not all correspond to the three hypothesized reflection contours described in the previous Section. The reason we chose them is that in practical applications the same three contours can be robustly extracted through depth edge detection techniques based on multi-flash imaging [24]. However, in this work we chose manual tracing over automatic edge detection because single pictures were available only and because intensity-based edge detection methods (e.g. Canny, Sobel) fail in low-contrast areas such as those in the available pictures.

Table 1: *Pearson correlation coefficients between each pair of anthropometric parameters of the pinna.*

parameter	description	d_2	d_3	d_4	d_5	d_6	d_7	d_8	θ_1	θ_2
d_1	cavum concha height	-0.06	0.10	0.19	0.51	0.21	0.23	0.24	-0.04	0.20
d_2	cymba concha height		-0.02	0.33	0.47	0.13	0.11	0.30	0.02	-0.11
d_3	cavum concha width			0.03	0.19	0.47	0.59	0.27	0.23	0.02
d_4	fossa height				0.67	0.53	0.03	0.30	-0.06	-0.28
d_5	pinna height					0.52	0.22	0.44	-0.11	0.00
d_6	pinna width						0.16	0.45	0.09	-0.24
d_7	intertragal incisure width							0.15	-0.09	0.03
d_8	cavum concha depth								0.01	0.14
θ_1	pinna rotation angle									-0.12
θ_2	pinna flare angle									

3.3. Anthropometric regression

In order to investigate the dependence of F_1, F_2 and F_3 frequencies on anthropometry, multiple linear regression is performed on the thirteen measured parameters (d_i, $i = 1 \dots 8$, θ_j, $j = 1 \dots 2$, $r_k(\phi)$, $k = 1 \dots 3$, $\phi \in [-45, 45]$) for all subjects and elevations where the notch is available. Since our focus is on anthropometric parameters, the elevation angle ϕ is not considered as a regressor. We compute first a model for each notch including all variables; then, if the model is capable of significantly accounting for a large portion of variance in the corresponding notch frequency data, we design a more conservative and meaningful model in accordance with the following greedy algorithm.

1. Calculate pairwise correlations between notch frequencies and each anthropometric parameter.
2. Compute a univariate linear regression model with the parameter showing the highest unsigned correlation value in Step 1 as regressor and notch frequency as outcome.
3. Calculate pairwise correlations between residuals of the previous model and each of the remaining anthropometric parameters.
4. Compute a multivariate linear regression model by adding the parameter showing the highest unsigned correlation value in Step 3 to the list of regressors used in the previous model.
5. If the model significantly improves the previous instance at the $p = 0.05$ level (according to an analysis of variance), label the model as "good" and return to Step 3. Otherwise, output the last "good" model.

This model allows to check which variables significantly account for variation in the corresponding notch frequency, as well as the number of variables needed in order to have an acceptable approximation of the outcome.

4. RESULTS

Table 2 reports the results of the three multiple linear regression models (each called M_i in association with F_i, $i = 1, 2, 3$) with all 13 anthropometric parameters as regressors and each notch frequency as outcome. M_1 outperforms the other two models in terms of R^2 value, denoting a much stronger fit for notch N_1 than for N_2 or N_3. The very low R^2 values for both M_2 and M_3 let us conclude that the 13 parameters are not sufficient to explain tight relations between anthropometry and notches N_2 and N_3.

Let us now concentrate on model M_1. Slopes of most of the thirteen parameters, as well as the intercept, are highly significantly different from zero, with only the two rotation angles θ_1 and θ_2 resulting weak parameters as well as r_2, which is however tightly correlated with r_1 as previously discussed. Focusing on the most significant parameters ($p < 0.001$), coefficient signs reveal that both r_1 and r_3 have negative slope, indicating that as the distance between reference point and pinna contour increases F_1 decreases, as hypothesized in our previous ray-tracing model. Elevation-independent parameters have, by contrast, different signs; we will return to this point later on in this discussion.

Model M_1 suffers from considerable variance inflation related to some of its parameters. As an example, the Variance Inflation Factor (VIF) for parameters d_5, r_1 and r_2 is 6.8, 14.6, and 14.9 respectively, denoting the presence of unnecessary variables in the model. As a consequence, we build a more conservative model according to the algorithm outlined in Section 3.3. The algorithm chooses eight variables in the following order: r_1, d_6, d_1, r_3, d_2, d_3, d_8, d_7, with the model built at each step with the inclusion of a new variable significantly improving the previous model at the $p = 0.001$ level according to analysis of variance. The first chosen variable, r_1, accounts alone for a large percentage of variation, equal to $R^2 = 0.51$. This result can be related to the pinna model in Section 2, which assumed notch N_1 to be caused by reflections on the outermost pinna contour.

Again, results of the final model, which we call $\hat{M}_1$, are reported in Table 2. Notice how close the summary results are to those of M_1, confirming that the 8 chosen variables are sufficient in explaining the variance accounted for in M_1. Also, coefficient signs do not reverse passing from one model to the other, hence the previous observations on r_1 and r_3 still hold. For what concerns elevation-independent parameters, notice that the height parameters d_1 and d_2 carry a negative sign, indicating that an increase in height implies lower notch frequencies, in accordance with results by Middlebrooks [25]. By contrast, the width and depth parameters d_3, d_6, d_7 and d_8 have positive signs in 3 out of 4 cases.

Let us now turn to some considerations of psychoacoustic nature. The residual standard error of $\hat{M}_1$ is 0.59 kHz, meaning that our approximation introduces significant errors if used for predicting F_1, and the displacement of pivotal elevation cues such as pinna notches is known to have an impact on localization accuracy. For instance, it is known and verifiable in our data that 1-kHz shifts of N_1 can correspond to an increase/decrease of the elevation angle of 20° or more [14]. However, from previous literature [13]

Table 2: *Summary of the resulting linear regression models. For each model, the coefficient of each regressor and its level of significance (coefficient different from zero at level $p = 0.05$ (*), $p = 0.01$ (**), and $p = 0.001$ (***)) are reported, along with the RMS error and the R^2 value arising from cross-validation. Units of measurement are: cm ($d_1 - d_8$, $r_1 - r_3$), rad (θ_1, θ_2), kHz (outcomes $F_1 - F_3$).*

	M_1		$\hat{M}_1$		M_2		M_3	
RMS error [kHz]	0.59		0.59		0.75		1.27	
R^2	0.76		0.76		0.34		0.15	
parameter	coefficient	level	coefficient	level	coefficient	level	coefficient	level
d_1	-1.85	***	-1.48	***	-1.22	***	-1.61	**
d_2	-2.21	***	-1.70	***	-1.74	***	-3.44	**
d_3	-1.01	***	-1.03	***	0.74	**	0.16	
d_4	-0.51	**			-0.27		2.54	***
d_5	0.47	**			0.70	**	0.13	
d_6	1.76	***	1.79	***	0.11		-1.44	***
d_7	1.15	**	1.45	***	-1.59	***	2.55	**
d_8	0.74	***	0.79	***	-0.25		-1.44	**
θ_1	-0.07				-0.84	*	2.26	**
θ_2	-0.27				-1.06	*	-1.18	
r_1	-1.54	***	-1.16	***	-0.23		1.03	
r_2	0.42	*			-0.47	*	-1.43	*
r_3	-0.81	***	-0.78	***	-0.24		-0.12	
intercept [kHz]	9.84	***	10.44	***	11.19	***	17.21	***

we also know that two steady notches in the high-frequency range (around 8 kHz) differing in center frequency are distinguishable on average if such difference is around 10% of the lowest center frequency at least, independent of notch bandwidth.

If we extend the above assumption to the frequency range of N_1, we can detect all those notch frequencies predicted by $\hat{M}_1$ that exceed 10% of the corresponding extracted notch frequency F_1. This results in approximately 80% of predicted notch frequencies lying below such a threshold. If we take into account that different sources of error may exist in our data, for instance

- a completely automatic extraction of notch frequencies that may introduce artifacts during both the notch picking and tracking procedures, and
- the arbitrary placement of the ear-canal reference point in the computation of elevation-dependent parameters, which does not necessarily correspond to the microphone position in CIPIC measurements [11],

we can assess the overall fitness of our model. Still, having in regard that notch detectability heavily depends on stimulus intensity and intersubject variation [26], individual psychoacoustic tests are needed in order to ascertain whether the proposed approximation correlates to elevation performance in the median plane.

5. CONCLUSIONS

In this paper we studied linear models for approximating frequencies of pinna notches in frontal median-plane HRTFs from anthropometric parameters. Results for notch N_1 show an encouraging correspondence between anthropometry and HRTF features, confirming the possibility of predicting pinna notches from pictures of the ear. Even though the proposed model represents a step forward towards a full understanding of the physical mechanisms lying behind the generation of pinna notches, it bears some limitations.

First of all, the model is built from data included in the CIPIC HRTF database. Even though it is the most diffuse among the scientific community, the CIPIC database suffers from measurement error (such as left/right asymmetries [27]) and lack of documentation (e.g. no reference about microphone position) which likely contributed to increase the clutter in our data. More recent and documented databases such as the Aalto PRTF database [21] or the SYMARE database [28] will be used in future works in order to carry a more controlled analysis on the available data.

Secondly, the analysis was only conducted in the frontal half of the median plane. In order for results to be general, a wider and denser grid of HRTFs will need to be considered and the proposed notch extraction/tracking procedures to be accordingly tuned and extended to account for a 2-D (or possibly even 3-D, if distance dependence is included) representation of the notch data.

Last but not least, our model precludes the inclusion of possible nonlinear effects, which will be investigated in the future through the use of state-of-the-art machine learning techniques for nonlinear regression. The contingent availability of a large amount of data will allow the construction of proper training and test sets in order to have a technically sound data analysis.

6. ACKNOWLEDGMENTS

The authors wish to thank Mr. Silvio Galesso for his help with anthropometric data extraction. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 643636.[2] This work was supported by the research project Personal Auditory Displays for Virtual Acoustics, University of Padova, under grant No CPDA135702.

[2] Sound of Vision, `www.soundofvision.net`

7. REFERENCES

[1] C. I. Cheng and G. H. Wakefield, "Introduction to head-related transfer functions (HRTFs): Representations of HRTFs in time, frequency, and space," *J. Audio Eng. Soc.*, vol. 49, no. 4, pp. 231–249, April 2001.

[2] D. R. Begault, E. M. Wenzel, and M. R. Anderson, "Direct comparison of the impact of head tracking, reverberation, and individualized head-related transfer functions on the spatial perception of a virtual speech source," *J. Audio Eng. Soc.*, vol. 49, no. 10, pp. 904–916, October 2001.

[3] B. Xie, *Head-Related Transfer Function and Virtual Auditory Display*, J.Ross Publishing, Plantation, FL, USA, 2nd edition, June 2013.

[4] H. Møller, M. F. Sørensen, C. B. Jensen, and D. Hammershøi, "Binaural technique: Do we need individual recordings?," *J. Audio Eng. Soc.*, vol. 44, no. 6, pp. 451–469, June 1996.

[5] C. P. Brown and R. O. Duda, "A structural model for binaural sound synthesis," *IEEE Trans. Speech Audio Process.*, vol. 6, no. 5, pp. 476–488, September 1998.

[6] L. Li and Q. Huang, "HRTF personalization modeling based on RBF neural network," in *Proc. 38th IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP 2013)*, Vancouver, BC, Canada, May 2013, pp. 3707–3710.

[7] Q. Huang and L. Li, "Modeling individual HRTF tensor using high-order partial least squares," *EURASIP J. Adv. Signal Process.*, vol. 2014, no. 58, pp. 1–14, May 2014.

[8] P. Bilinski, J. Ahrens, M. R. P. Thomas, I. J. Tashev, and J. C. Platt, "HRTF magnitude synthesis via sparse representation of anthropometric features," in *Proc. 39th IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP 2014)*, Firenze, Italy, May 2014, pp. 4501–4505.

[9] F. Grijalva, L. Martini, S. Goldenstein, and D. Florencio, "Anthropometric-based customization of head-related transfer functions using Isomap in the horizontal plane," in *Proc. 39th IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP 2014)*, Firenze, Italy, May 2014, pp. 4506–4510.

[10] S. K. Roffler and R. A. Butler, "Factors that influence the localization of sound in the vertical plane," *J. Acoust. Soc. Am.*, vol. 43, no. 6, pp. 1255–1259, June 1968.

[11] S. Spagnol, M. Geronazzo, and F. Avanzini, "On the relation between pinna reflection patterns and head-related transfer function features," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 3, pp. 508–519, March 2013.

[12] K. Iida, M. Itoh, A. Itagaki, and M. Morimoto, "Median plane localization using a parametric model of the head-related transfer function based on spectral cues," *Appl. Acoust.*, vol. 68, no. 8, pp. 835–850, August 2007.

[13] B. C. J. Moore, S. R. Oldfield, and G. J. Dooley, "Detection and discrimination of spectral peaks and notches at 1 and 8 kHz," *J. Acoust. Soc. Am.*, vol. 85, no. 2, pp. 820–836, February 1989.

[14] J. Hebrank and D. Wright, "Spectral cues used in the localization of sound sources on the median plane," *J. Acoust. Soc. Am.*, vol. 56, no. 6, pp. 1829–1834, December 1974.

[15] P. Satarzadeh, R. V. Algazi, and R. O. Duda, "Physical and filter pinna models based on anthropometry," in *Proc. 122nd Conv. Audio Eng. Soc.*, Vienna, Austria, May 2007, pp. 718–737.

[16] K. J. Faller II, A. Barreto, and M. Adjouadi, "Augmented Hankel total least-squares decomposition of head-related transfer functions," *J. Audio Eng. Soc.*, vol. 58, no. 1/2, pp. 3–21, January/February 2010.

[17] P. Mokhtari, H. Takemoto, R. Nishimura, and H. Kato, "Pinna sensitivity patterns reveal reflecting and diffracting surfaces that generate the first spectral notch in the front median plane," in *Proc. 36th IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP 2011)*, Prague, Czech Republic, May 2011, pp. 2408–2411.

[18] M. Geronazzo, S. Spagnol, and F. Avanzini, "A head-related transfer function model for real-time customized 3-D sound rendering," in *Proc. INTERPRET Work., SITIS 2011 Conf.*, Dijon, France, November-December 2011, pp. 174–179.

[19] V. R. Algazi, R. O. Duda, D. M. Thompson, and C. Avendano, "The CIPIC HRTF database," in *Proc. IEEE Work. Appl. Signal Process., Audio, Acoust.*, New Paltz, New York, USA, October 2001, pp. 1–4.

[20] V. C. Raykar, R. Duraiswami, and B. Yegnanarayana, "Extracting the frequencies of the pinna spectral notches in measured head related impulse responses," *J. Acoust. Soc. Am.*, vol. 118, no. 1, pp. 364–374, July 2005.

[21] S. Spagnol, M. Hiipakka, and V. Pulkki, "A single-azimuth pinna-related transfer function database," in *Proc. 14th Int. Conf. Digital Audio Effects (DAFx-11)*, Paris, France, September 2011, pp. 209–212.

[22] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 4, pp. 744–754, August 1986.

[23] S. Spagnol, "On distance dependence of pinna spectral patterns in head-related transfer functions," *J. Acoust. Soc. Am.*, vol. 137, no. 1, pp. EL58–EL64, January 2015.

[24] S. Spagnol, D. Rocchesso, M. Geronazzo, and F. Avanzini, "Automatic extraction of pinna edges for binaural audio customization," in *Proc. IEEE Int. Work. Multi. Signal Process. (MMSP 2013)*, Pula, Italy, September-October 2013, pp. 301–306.

[25] J. C. Middlebrooks, "Individual differences in external-ear transfer functions reduced by scaling in frequency," *J. Acoust. Soc. Am.*, vol. 106, no. 3, pp. 1480–1492, September 1999.

[26] A. Alves-Pinto and E. A. Lopez-Poveda, "Detection of high-frequency spectral notches as a function of level," *J. Acoust. Soc. Am.*, vol. 118, no. 4, pp. 2458–2469, October 2005.

[27] S. Spagnol and F. Avanzini, "Anthropometric tuning of a spherical head model for binaural virtual acoustics based on interaural level differences," in *Proc. 21st Int. Conf. Auditory Display (ICAD)*, Graz, Austria, July 2015, pp. 204–209.

[28] C. Jin, P. Guillon, N. Epain, R. Zolfaghari, A. van Schaik, A. I. Tew, C. Hetherington, and J. Thorpe, "Creating the Sydney York morphological and acoustic recordings of ears database," *IEEE Trans. Multimedia*, vol. 16, no. 1, pp. 37–46, January 2014.

RELATIVE AUDITORY DISTANCE DISCRIMINATION WITH VIRTUAL NEARBY SOUND SOURCES

Simone Spagnol,
Faculty of Ind. Eng., Mech. Eng. and Computer Science,
University of Iceland
Reykjavík, Iceland
spagnols@hi.is

Erica Tavazzi,
Department of Information Engineering,
University of Padova
Padova, Italy
tavazzie@dei.unipd.it

Federico Avanzini,
Department of Information Engineering,
University of Padova
Padova, Italy
avanzini@dei.unipd.it

ABSTRACT

In this paper a psychophysical experiment targeted at exploring relative distance discrimination thresholds with binaurally rendered virtual sound sources in the near field is described. Pairs of virtual sources are spatialized around 6 different spatial locations (2 directions $\times 3$ reference distances) through a set of generic far-field Head-Related Transfer Functions (HRTFs) coupled with a near-field correction model proposed in the literature, known as DVF (Distance Variation Function). Individual discrimination thresholds for each spatial location and for each of the two orders of presentation of stimuli (approaching or receding) are calculated on 20 subjects through an adaptive procedure. Results show that thresholds are higher than those reported in the literature for real sound sources, and that approaching and receding stimuli behave differently. In particular, when the virtual source is close (< 25 cm) thresholds for the approaching condition are significantly lower compared to thresholds for the receding condition, while the opposite behaviour appears for greater distances (≈ 1 m). We hypothesize such an asymmetric bias to be due to variations in the absolute stimulus level.

1. INTRODUCTION

Spatial auditory features can be rendered through headphones by processing an input sound with a pair of left/right filters, each simulating all the linear transformations undergone by the acoustic signal during its path from the sound source to the corresponding listener's eardrum. These filters are known in the literature as Head-Related Transfer Functions (HRTFs), formally defined as the ratio between the acoustic pressure produced by a sound source at the eardrum, and the free-field pressure that would be produced by the same sound source at the listener's head center [1]. By this definition, and due to the fact that spherical wavefronts become progressively planar for increasing distances, HRTFs are approximately distance-independent in the so-called *far field* (i.e. for distances greater than 1 m from the center of the head) as opposed to the *near field* (i.e. less than 1 m from the center of the head) [2].

In order for a Virtual Auditory Display (VAD) to produce perceptually convincing results over headphones, dense and accurate sets of HRTFs are needed [3]. Unfortunately, several technical challenges often limit the availability of such data. Collecting individual HRTF sets of a human subject requires an anechoic room, in-ear microphones, and a loudspeaker moving around the subject in order to measure responses at different directions. As a consequence many real-world applications typically use generic HRTF sets (e.g., measured on a mannequin), which lack important features that depend on individual anthropometry [4, 5]. Even more important for the scope of this paper, HRTFs are typically measured at one single distance in the far field, whereas near-field HRTFs as previously said are distance-dependent and should thus be measured at various distances for subsequent interpolation.

As a consequence, near-field HRTF databases have more demanding requirements in terms of both measuring times and memory usage. Moreover they require the measurement system to accommodate for controlled variations of loudspeaker-subject distance. Measurement errors are also larger, as very small head movements can substantially alter the speaker direction. Because of these difficulties, very few databases of near-field HRTFs are available. Qu *et al.* [6] collected and validated one such database that includes the responses of a KEMAR[1] at 14 elevation angles, 72 azimuth angles, and 8 distances, for a total of 12688 HRTFs.

Still, even if near-field HRTFs are not available, a proper near-field VAD can be reconstructed by applying an ILD correction to a set of far-field HRTFs. This is what the Distance Variation Function (DVF) method by Kan *et al.* [7] specifically does: multiplying the far-field individual HRTF magnitude by a function that takes into account the pressure ratio between a near-field and the corresponding isodirectional far-field sound source observed on the surface of a rigid sphere [8]. Thanks to the introduction of a proper ILD, such a method was found to be more effective in conveying absolute distance information with respect to a simple $1/r$ intensity scaling of the far-field display, especially at very near distances (< 40 cm).

[1] Knowles Electronics Manikin for Acoustic Research, one of the most commonly used mannequins for non-individual HRTF measures.

An open question is to what extent the use of the DVF method in a VAD is able to convey relative, rather than absolute, distance information, and whether such information is symmetric with respect to the order of presentation of two isodirectional virtual stimuli or not. Our starting point is a previous work [9] where we compared the DVF method to a low-order filter approximation of itself [10] and found different error statistics between different orders of presentation as a collateral result. Thus, in this paper our aim is to investigate, through an ad-hoc designed adaptive psychophysical experiment, whether individual perceptual discrimination thresholds for two isodirectional virtual stimuli created through the DVF method vary with the order of presentation and/or with the location of the virtual stimuli themselves.

2. BACKGROUND

2.1. Auditory distance estimation

Our ability to estimate the physical distance of a sound source is influenced by a number of factors [11]. Sound intensity is the first cue taken into account: the weaker the intensity, the farther the source should be perceived. Under anechoic conditions, the intensity of a sound source decays of 6 dB for each doubling distance and can thus be predicted by a $1/r$ pressure attenuation law [12], where r is the distance between source and receiver. Having a certain familiarity with the involved sound is, however, a fundamental requirement: if the sound is unfamiliar then intensity cues work only on a relative basis [13]. The just noticeable difference (jnd) in the relative distance between two isodirectional sound sources can indeed be directly related to the 5% intensity jnd [12], even though higher jnd's (up to 50%) have been reported for very near sound sources [11]. When the intensity cue is not available relative distance discrimination severely degrades [12].

In anechoic conditions with a familiar sound source, absolute distance is better estimated when the source is lateral to the subject (especially on his interaural axis) and worse when the source is in the median plane [14]. On the other hand, if the environment is reverberant then the proportion of reflected to direct energy (known as *R/D ratio*) works as a stronger absolute cue for distance than intensity [15]. Also, by gradually approaching the sound source to the listener's head in the near field it was observed that relevant additional distance cues such as a low-frequency spectral boost and a dramatic increase of the interaural level difference (ILD) across the whole spectrum for lateral sources arise [16].

When the sound source is virtually rendered and presented binaurally with a pair of measured near-field HRTFs, both directional localization and absolute distance estimation typically degrade. Still, Brungart and Simpson [17] found a significant correlation between simulated and perceived distance on the interaural axis using generic KEMAR HRTFs and no intensity/reverberation cues. By contrast, if the DVF method is used, when intensity cues are removed performances severely degrade, confirming that the intensity cue is still dominant in near-field VADs [7].

2.2. The DVF method

The DVF method is based on the analytical formulation of the spherical head model, whose transfer function (i.e. the ratio between the pressure p_S that a point source generates on an observation point on the surface of the sphere, and the free-field pressure p_{ff}) we refer to as *spherical transfer function (STF)*. In this formulation, each considered spatial location of the sound source is specified by two coordinates: the *incidence angle* α, i.e. the angle between rays connecting the center of the sphere to the source and the observation point, and the distance r to the center of the sphere, which can also be expressed in relation to the sphere radius a as $\rho = r/a$ (*normalized distance*). For each $\rho > 1$, the STF can be evaluated by means of the following function [8]:[2]

$$STF(\mu, \alpha, \rho) = -\frac{\rho}{\mu} e^{-i\mu\rho} \sum_{m=0}^{\infty} (2m+1) P_m(\cos\alpha) \frac{h_m(\mu\rho)}{h'_m(\mu)}, \quad (1)$$

where μ is the *normalized frequency*, defined as

$$\mu = f \frac{2\pi a}{c}, \quad (2)$$

and c is the speed of sound[3].

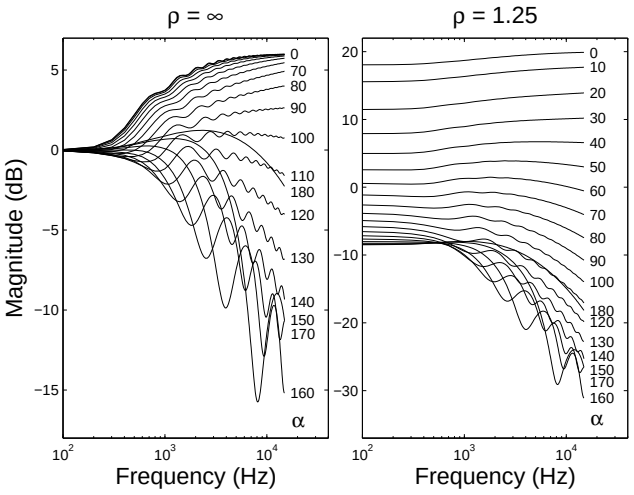


Figure 1: Far-field and near-field Spherical Transfer Functions: $\rho \to \infty$ (left panel) and $\rho = 1.25$ (right panel).

Figure 1 shows the magnitude of the so calculated transfer function for 19 incidence angle values and two distances, $\rho \to \infty$ (far field) and $\rho = 1.25$ (near field). Notice that in the near field the response increases on the ipsilateral side and decreases on the contralateral side, even at low frequencies. This effect explains the aforementioned ILD boost at small distances across the whole frequency range. Also notice that the infinite sum in Eq. (1) does not allow a direct computation of $STF(\mu, \alpha, \rho)$, while computation of spherical Hankel functions and Legendre polynomials requires high computational costs. The solution to these shortcomings is provided by a recursive algorithm [18] where the latter functions are computed iteratively, allowing a relatively fast evaluation.

In a previous work [19], the authors used Principal Component Analysis (PCA) in order to study how incidence angle and distance affect STF variability. Results indicate that after the first basis vector which retains the average behaviour of the STF, those from the second onwards provide each a description of the rippled high-frequency behaviour of contralateral STFs, which varies according to the incidence angle. However, distance dependence clearly

[2] Here P_m and h_m represent, respectively, the *Legendre polynomial* of degree *m* and the *m*th-order *spherical Hankel function*. h'_m is the derivative of h_m with respect to its argument.

[3] Considering dry-air conditions at $20°$C temperature, $c = 343.2$ m/s.

arises when comparing the average gain of far-field and near-field STFs. In light of this result, the STF at a given near-field distance ρ_n can be represented as a far-field STF (at distance ρ_f) multiplied by a correcting term. This corresponds to the *intensity-scaled DVF* as defined by Kan *et al.* [7]. The proper DVF including intensity information needs a further correction by a term equal to the ratio of the far-field distance to the near-field distance, accounting for the differences in the free-field pressures at the two reference points:

$$DVF(\mu, \alpha, \rho_n, \rho_f) = \frac{STF(\mu, \alpha, \rho_n)}{STF(\mu, \alpha, \rho_f)} \times \frac{\rho_f}{\rho_n}. \quad (3)$$

Once the DVF for a given near-field location (θ, ϕ, ρ) is known (where azimuth θ and elevation ϕ uniquely define an α value depending on the used coordinate system), it can be applied to any far-field HRTF to obtain the corresponding near-field HRTF approximation as

$$\widetilde{HRTF}(\mu, \theta, \phi, \rho_n) = DVF(\mu, \alpha, \rho_n, \rho_f) \times HRTF(\mu, \theta, \phi). \quad (4)$$

It could be questioned whether analytical DVFs objectively reflect distance-dependent patterns in real measured HRTFs of human subjects. As a matter of fact, a non-analytical DVF (derived from the ratio between a near-field HRTF and a far-field HRTF) is likely to result more and more sensitive to geometric features of the head as the sound source approaches and, since the sphere can be considered as a simple scatterer, it could become an increasingly worse approximation of the real near-field effects. However, we know that the spherical model from which the DVF emerges closely matches typical measured HRTF patterns in the low frequency range (< 1 kHz) [16] where near-field cues are prominent, and accurately predicts the RMS pressure at the near ear as a function of distance for both medial and lateral sources, although slightly underestimating ILD [20]. Thus, the most relevant features of the near field shall be preserved.

3. EXPERIMENTAL DESIGN

In order to investigate relative auditory distance discrimination with virtual sound sources binaurally rendered through the DVF method presented above, a psychophysical experiment was conducted. Pairs of isodirectional virtual sources at two different distances were used as experimental stimuli, and the subject's task was to estimate which of the two sounds was closer. The novelty of this work with respect to previous works with near-field VADs and/or the DVF method lies indeed in the fact that relative, rather than absolute, localization judgments were asked to experimental subjects.

We used KEMAR HRTFs measured in the far field (distance $r_{ff} = 1.6$ m from the center of the manikin's head) from the PKU&IOA database [6] as the reference far-field virtual auditory display. Similarly to previous works [17, 21], non-individual HRTFs were primarily chosen as the far-field display in order to simulate a feasible scenario for practical applications where individual HRTFs are typically not available. Although non-individual HRTFs are known to be the source of localization errors such as front/back reversals [22], elevation angle misperception [23], and inside-the-head localization [24], distance estimation was found not to significantly change when switching from the individual HRTF to a non-individual one [25]. The choice of the PKU&IOA database was due for the sake of consistency with previous experiments [9]. Also similarly to the previously cited works [17, 21], no reverberation was introduced in order to have more control on anechoic distance cues such as intensity and ILD. As a consequence, the R/D ratio cue was not available to experimental subjects.

3.1. Subjects and apparatus

Twenty subjects (7 female and 13 male) participated in the experiment on a voluntary basis. Subjects' ages ranged from 22 to 49 years (mean $= 27.2$, SD $= 6.8$). All subjects reported normal hearing defined as thresholds no greater than 25 dB HL in the range of 125 Hz to 8 kHz according to an audiometric screening based on an adaptive maximum likelihood procedure [26].

The experiment took place inside a dark Sound Station Pro 45 silent booth. The experimental subject sat on a chair in front of a small table holding a keyboard whose direction arrow keys up and down were colored blue and red, respectively. The subject wore a pair of Sennheiser HDA 200 headphones (frequency response $20 - 20$k Hz, impedence 40Ω) plugged to a Roland Edirol AudioCapture UA-101 external audio card working at a sampling rate of 48 kHz. The compensation filter proposed by Lindau and Brinkmann [27] was used to compensate headphone responses.

A PC screen was also present in front of the subject, but it was turned off during the experimental sessions in order to avoid visual distraction. The screen could be optionally turned on during breaks to show a countdown to the following block of trials. Keyboard, audio card and screen were all plugged to a PC placed on the floor running the control software implemented in MATLAB.

3.2. Stimuli

All stimuli used as sound source signal a a 400-ms uniformly distributed white noise with 30-ms onset and offset linear ramps. This signal was used in order to facilitate comparisons with relative distance localization results with real sound sources by Ashmead *et al.* [12] and to avoid familiarity issues. The average measured amplitude of the raw signal at the entrance of the ear canal was approximately 60 dB(A). Spatialized sounds were then created by filtering the sound source signal through a pair of near-field HRTFs obtained through the DVF method, where parameter a (head radius) was fixed to the standard 8.75 cm value [28].

The virtual sound source was simulated on the horizontal plane in two different directions, labeled as

- L (lateral), with the source location pseudo-randomly chosen between right ($\theta = 90°$) and left ($\theta = 270°$), and
- M (medial), i.e. with the source located behind ($\theta = 180°$).

The latter location was preferred to a directly ahead source because of the potentially significant number of front/back reversals ascribable to non-individual HRTFs [17] and in order to avoid possible associations with visual anchors. For each direction, we fixed three reference distance values, labeled as

- N (near-field), 25 cm away from the center of the head;
- H (halfway), 50 cm away from the center of the head;
- F (far-field), 100 cm away from the center of the head.

Having fixed a certain direction and reference distance value, virtual stimuli corresponding to the reference distance (e.g. 50 cm) and a lower distance (e.g. 40 cm) were proposed in sequence to the experimental subject in either order, labeled as

Table 1: *Relative distance discrimination thresholds [%] of the* 20 *experimental subjects for each condition.*

ID	sex	age	NLR	NLA	HLR	HLA	FLR	FLA	NMR	NMA	HMR	HMA	FMR	FMA
01	M	30	24.6	0.0	18.5	4.5	0.0	17.5	30.0	0.0	10.5	6.7	0.0	23.5
02	F	24	22.3	0.0	17.1	6.2	4.9	21.3	24.7	0.0	13.2	6.3	0.0	21.8
03	M	23	23.1	0.0	20.3	10.0	0.0	25.6	26.0	0.0	9.0	12.4	1.4	25.3
04	F	25	24.3	0.0	9.3	15.2	0.0	24.9	27.8	0.0	17.2	24.5	0.0	27.9
05	M	27	28.7	0.0	24.1	5.7	13.9	25.7	30.0	0.0	25.4	3.5	17.7	20.6
06	M	23	27.7	0.0	26.2	0.0	15.3	7.8	30.0	0.0	16.8	0.0	13.5	15.1
07	M	30	27.1	0.0	17.7	0.0	18.9	19.5	25.9	0.0	18.0	0.0	12.8	20.4
08	M	49	22.5	0.0	17.9	0.0	0.0	16.9	22.7	0.0	9.9	10.2	0.0	23.9
09	M	24	29.1	0.0	25.1	9.4	0.0	18.0	32.0	0.0	15.6	10.2	7.6	21.9
10	M	27	18.9	0.0	16.3	0.0	6.5	22.2	26.3	0.0	14.7	14.6	11.5	29.9
11	M	25	19.5	0.0	29.6	0.0	25.9	15.1	29.7	0.0	28.5	6.9	17.5	19.5
12	F	23	22.1	0.0	20.3	0.0	0.0	26.8	31.0	0.0	24.7	4.0	0.0	22.8
13	F	23	22.5	0.0	21.1	5.3	3.5	20.7	28.0	0.0	16.7	16.1	15.2	26.3
14	M	42	20.5	0.0	13.6	0.0	14.1	15.9	25.0	0.0	13.3	0.0	8.1	14.9
15	M	23	23.1	0.0	14.5	0.0	14.9	14.7	20.1	9.5	9.1	11.7	9.2	16.2
16	F	22	21.3	0.0	16.3	0.0	9.9	14.8	28.1	0.0	13.0	0.0	6.1	13.1
17	F	25	27.0	0.0	23.1	13.5	25.1	13.5	22.8	17.4	28.3	18.2	14.4	24.6
18	M	27	26.3	0.0	20.1	0.0	0.0	17.5	30.1	0.0	18.0	4.5	1.0	25.6
19	M	24	17.2	2.7	14.1	11.1	0.0	23.0	31.9	0.0	20.2	0.0	2.8	19.6
20	F	28	27.9	0.0	27.2	11.6	13.5	26.9	29.5	0.0	27.1	14.2	0.0	29.0
mean	-	-	23.8	0.1	19.6	4.6	8.3	19.4	27.6	1.3	17.4	8.2	6.9	22.1

- R (receding), e.g. $40 - 50$ cm, and
- A (approaching), e.g. $50 - 40$ cm,

with a 500 ms pause separating the two sounds.

3.3. Protocol

The combination of 3 reference distances, 2 directions, and 2 orders gave rise to 12 different experimental conditions. The goal of the experiment was to adaptively determine through 12 different sequences of trials the individual discrimination threshold of the two stimuli in each condition.

The subject wore the headphones and received instructions from a recorded voice generated through a Text-To-Speech software. At each trial, the next stimulus pair in one of the active sequences (picked in pseudo-random order) was presented, as we will shortly explain. The subject was instructed to report whether he perceived the second stimulus nearer or farther than the first, by pressing the red or blue key respectively. The recorded voice also signaled the beginning and the end of each block of trials, where the maximum number of trials for each block was 200, inviting the subject to take a mandatory 3-minute break between them. Each subject underwent a short training session (10 similar trials, with no feedback on answer accuracy) just before the experimental session. The average total duration of the experiment was 45 minutes.

The adaptive procedure was based on the algorithm proposed by Ashmead *et al.* [12] and runs as follows. Having fixed one of the 12 conditions, the initial adaptive (lower) distance in the first trial of the sequence is chosen by reducing the reference distance by 20%. The following trials are determined by moving the adaptive distance point in 1% steps with respect to the reference distance according to a *1-down, 1-up* algorithm up to the fifth reversal (i.e., incorrect answer), and a *2-down, 1-up* algorithm for the following trials [29]. For instance, if the reference distance is N (25 cm) and the order is A (approaching), the second stimulus is set at 20 cm in the first trial and subsequently moves in 0.25 cm steps, approaching the reference distance if the subject perceives the correct order of presentation (i.e., if the red key is pressed) and receding otherwise (i.e., if the blue key is pressed), up to the fifth reversal. From then onwards, the second stimulus approaches the reference distance if and only if two correct answers in a row are given, but keeps receding at each single reversal. Each sequence (condition) ends either at the twentieth reversal or when the adaptive distance reaches the reference distance (0% difference).

Individual discrimination thresholds were then computed by averaging the differences (expressed as percentage of the reference distance) between the two distances surrounding reversals 6 to 20. If the sequence ended because the adaptive distance reached the reference distance, we considered the total number of reversals in that sequence, n_r. If $n_r > 5$, the threshold was similarly set to the average distance difference surrounding reversals 6 to n_r; otherwise, it was set to zero.

4. RESULTS AND DISCUSSION

While complete results of all subjects are reported in Table 1, the barplot in Fig. 2 summarizes the mean and standard deviation of the above defined individual thresholds for each of the 12 experimental conditions. In both Table 1 and Fig. 2 each condition is labeled with a three-character string, reporting from left to right labels of the reference distance (N, H, or F), the direction (L or M), and the order (R or A).

The average threshold among all conditions is around 13%. If we consider the reference point in the median plane at 1 m, i.e. the same location used by Ashmead *et al.* for investigating discrimination thresholds for real sound sources [12], the average of our two corresponding conditions FMR and FMA is 14.51%, with SD

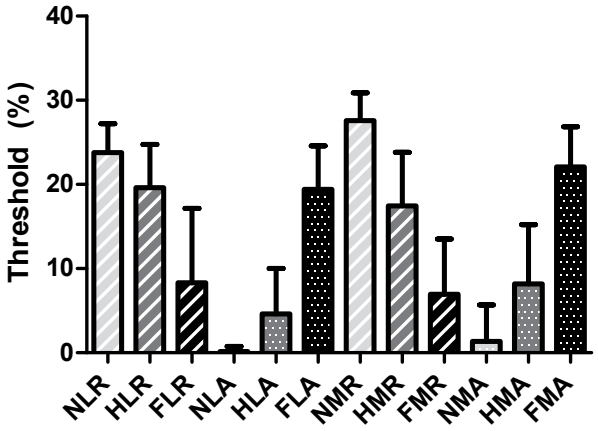


Figure 2: Mean and standard deviation [%] of relative distance discrimination thresholds for the 12 experimental conditions.

equal to 5.67%. Such higher values with respect to those found by Ashmead *et al.* (mean $= 5.73\%$, SD $= 2.26\%$) are supposed to be due to the use of virtual sources, where spatial information is inevitably lost with respect to the real case.

If we fix both reference distance and direction and compare orders R and A, we clearly observe opposite trends confirming an asymmetric bias in the perception of receding and approaching stimuli. As a matter of fact, discrimination thresholds are clearly lower for approaching stimuli than for receding stimuli in the near field, and lower for receding stimuli than for approaching stimuli towards the far field. Such an evidence, which already comes out clear from Fig. 2, is confirmed by nonparametric paired t-tests: all pairs of conditions differing in order only are significantly different at the $p = 0.01$ significance level.

The significantly higher thresholds for receding stimuli at close distances is in accordance with the results of a localization experiment with real near-field sources by Simpson and Stanton [30], who reported a higher distance jnd for receding than approaching sources especially at closer distances. The authors hypothesize this phenomenon to reflect an auditory counterpart of visual looming, an effect for which we are selectively tuned in favour of perceiving approaching stimuli as opposed to receding ones. However, they do not find an opposite trend for farther distances. The reason of our findings may be searched instead in the perception of the intensity cue. As reported by Olsen and Stevens [31], the perceived loudness change in pairs of discrete sound stimuli is significantly higher when the pair is presented in order of increasing level (i.e., approaching) than of decreasing level (i.e., receding) in the higher intensity region ($70-90$ dB, where our N conditions fall), whereas such discrepancy is exactly mirrored in the lower intensity region ($50 - 70$ dB, where our F conditions fall) where the perceived loudness change of decreasing pairs is higher.

We also observe higher average thresholds for receding stimuli than for approaching stimuli. In particular, notice that thresholds for receding stimuli are higher than thresholds for approaching stimuli in the opposite distance range, see e.g. conditions NLR and FLA, or FMR and NMA. This can be again related to a sort of correlation with the perceived stimulus level.

By contrast, we do not observe significant differences between directions L and M, except for the near-field receding conditions NLR and NMR, with lateral directions exhibiting lower thresholds. Such an effect is in accordance with results by Kan *et al.* [7] who found some absolute distance discrimination even with intensity cues removed in the lateral region within the $10 - 20$ cm distance range, and can be attributed to the relevance of the ILD cue in the nearest field as opposed to farther distances. Again, all the found differences are statistically significant according to nonparametric paired t-tests with significance level set to $p = 0.01$.

The latter effect is not found for near-field approaching conditions NLA and NMA which, however, both score an exceptionally high number of zero thresholds. It is interesting to notice how stimuli that should be in principle indistinguishable (such as two stimuli separated by a 1% or 2% distance difference) are instead almost always perceived as approaching by the vast majority of subjects. The reason for this should be found again in the previously discussed bias towards approaching stimuli in the near field.

5. CONCLUSIONS AND PERSPECTIVES

Near-field VADs have a plethora of possible applications, ranging from immersive virtual environments to speech applications [3, 32]. Results of the psychophysical experiment reported in this paper confirm the presence of an asymmetric perceptual bias for virtual sound sources created through the DVF method, whose relative distance discrimination thresholds heavily depend on the order of presentation (approaching or receding source) and on source distance. Such a bias is hypothesized to be due to the perception of the intensity cue.

Our results both confirm previous findings on auditory distance perception and complement the results of Kan *et al.* [7] on near-field distance perception with the DVF method, being based on relative - rather than absolute - judgments and applied to generic - rather than individual - far-field HRTFs. In order to investigate in more detail the found perceptual effects, further experiments where the overall level of presentation is roved or fixed at different reference intensities are planned. If such experiments were to support our hypothesis that intensity is the reason for the observed bias, these would also help understand at what reference intensity individual thresholds for approaching and receding sources roughly coincide, i.e., the turning point of the asymmetry. In addition, our experimental protocol could be applied to the case of real sound sources, with the aim of evaluating whether the found perceptual bias is due to limitations in near-field VADs or still holds in the real world.

6. ACKNOWLEDGMENTS

The authors wish to thank all the people who took part to the experiment for their precious collaboration. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 643636.[4] This work was supported by the research project Personal Auditory Displays for Virtual Acoustics, University of Padova, under grant No CPDA135702.

[4] Sound of Vision, `www.soundofvision.net`

7. REFERENCES

[1] C. I. Cheng and G. H. Wakefield, "Introduction to head-related transfer functions (HRTFs): Representations of HRTFs in time, frequency, and space," *J. Audio Eng. Soc.*, vol. 49, no. 4, pp. 231–249, April 2001.

[2] S. Spagnol, "On distance dependence of pinna spectral patterns in head-related transfer functions," *J. Acoust. Soc. Am.*, vol. 137, no. 1, pp. EL58–EL64, January 2015.

[3] D. S. Brungart, "Near-field virtual audio displays," *Presence*, vol. 11, no. 1, pp. 93–106, February 2002.

[4] S. Spagnol, M. Geronazzo, and F. Avanzini, "On the relation between pinna reflection patterns and head-related transfer function features," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 3, pp. 508–519, March 2013.

[5] S. Spagnol and F. Avanzini, "Anthropometric tuning of a spherical head model for binaural virtual acoustics based on interaural level differences," in *Proc. 21st Int. Conf. Auditory Display (ICAD 2015)*, Graz, Austria, July 2015, pp. 204–209.

[6] T. Qu, Z. Xiao, M. Gong, Y. Huang, X. Li, and X. Wu, "Distance-dependent head-related transfer functions measured with high spatial resolution using a spark gap," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 17, no. 6, pp. 1124–1132, August 2009.

[7] A. Kan, C. Jin, and A. van Schaik, "A psychophysical evaluation of near-field head-related transfer functions synthesized using a distance variation function," *J. Acoust. Soc. Am.*, vol. 125, no. 4, pp. 2233–2242, April 2009.

[8] W. M. Rabinowitz, J. Maxwell, Y. Shao, and M. Wei, "Sound localization cues for a magnified head: Implications from sound diffraction about a rigid sphere," *Presence*, vol. 2, no. 2, pp. 125–129, Spring 1993.

[9] S. Spagnol and F. Avanzini, "Distance rendering and perception of nearby virtual sound sources with a near-field filter model," 2015, submitted for publication.

[10] S. Spagnol, M. Geronazzo, and F. Avanzini, "Hearing distance: A low-cost model for near-field binaural effects," in *Proc. EUSIPCO 2012 Conf.*, Bucharest, Romania, September 2012, pp. 2005–2009.

[11] P. Zahorik, D. S. Brungart, and A. W. Bronkhorst, "Auditory distance perception in humans: a summary of past and present research," *Acta Acustica united with Acustica*, vol. 91, no. 3, pp. 409–420, May/June 2005.

[12] D. H. Ashmead, D. LeRoy, and R. D. Odom, "Perception of the relative distances of nearby sound sources," *Percept. Psychophys.*, vol. 47, no. 4, pp. 326–331, April 1990.

[13] P. D. Coleman, "Failure to localize the source distance of an unfamiliar sound," *J. Acoust. Soc. Am.*, vol. 34, no. 3, pp. 345–346, March 1962.

[14] M. B. Gardner, "Distance estimation of 0° or apparent 0°-oriented speech signals in anechoic space," *J. Acoust. Soc. Am.*, vol. 45, no. 1, pp. 47–53, 1969.

[15] D. H. Mershon and J. N. Bowers, "Absolute and relative cues for the auditory perception of egocentric distance," *Perception*, vol. 8, no. 3, pp. 311–322, 1979.

[16] D. S. Brungart and W. M. Rabinowitz, "Auditory localization of nearby sources. Head-related transfer functions," *J. Acoust. Soc. Am.*, vol. 106, no. 3, pp. 1465–1479, September 1999.

[17] D. S. Brungart and B. D. Simpson, "Auditory localization of nearby sources in a virtual audio display," in *Proc. IEEE Work. Appl. Signal Process., Audio, Acoust.*, New Paltz, New York, USA, October 2001, pp. 107–110.

[18] R. O. Duda and W. L. Martens, "Range dependence of the response of a spherical head model," *J. Acoust. Soc. Am.*, vol. 104, no. 5, pp. 3048–3058, November 1998.

[19] S. Spagnol and F. Avanzini, "Real-time binaural audio rendering in the near field," in *Proc. 6th Int. Conf. Sound and Music Computing (SMC09)*, Porto, Portugal, July 2009, pp. 201–206.

[20] B. G. Shinn-Cunningham, "Distance cues for virtual auditory space," in *Proc. 1st IEEE Pacific-Rim Conf. on Multimedia*, Sydney, Australia, December 2000, pp. 227–230.

[21] G. Parseihian, C. Jouffrais, and B. F. G. Katz, "Reaching nearby sources: Comparison between real and virtual sound and visual targets," *Front. Neurosci.*, vol. 8, pp. 1–13, September 2014.

[22] E. M. Wenzel, M. Arruda, D. J. Kistler, and F. L. Wightman, "Localization using nonindividualized head-related transfer functions," *J. Acoust. Soc. Am.*, vol. 94, no. 1, pp. 111–123, July 1993.

[23] H. Møller, M. F. Sørensen, C. B. Jensen, and D. Hammershøi, "Binaural technique: Do we need individual recordings?," *J. Audio Eng. Soc.*, vol. 44, no. 6, pp. 451–469, June 1996.

[24] G. Plenge, "On the differences between localization and lateralization," *J. Acoust. Soc. Am.*, vol. 56, no. 3, pp. 944–951, September 1974.

[25] P. Zahorik, "Distance localization using nonindividualized head-related transfer functions," *J. Acoust. Soc. Am.*, vol. 108, no. 5, pp. 2597, November 2000.

[26] D. M. Green, "A maximum-likelihood method for estimating thresholds in a yes-no task," *J. Acoust. Soc. Am.*, vol. 93, no. 4, pp. 2096–2105, April 1993.

[27] A. Lindau and F. Brinkmann, "Perceptual evaluation of headphone compensation in binaural synthesis based on non-individual recordings," *J. Audio Eng. Soc.*, vol. 60, no. 1/2, pp. 54–62, January 2012.

[28] R. V. L. Hartley and T. C. Fry, "The binaural location of pure tones," *Phys. Rev.*, vol. 18, no. 6, pp. 431–442, December 1921.

[29] H. Levitt, "Transformed up-down methods in psychoacoustics," *J. Acoust. Soc. Am.*, vol. 49, no. 2, pp. 467–477, 1971.

[30] W. E. Simpson and L. D. Stanton, "Head movement does not facilitate perception of the distance of a source of sound," *Am. J. Psych.*, vol. 86, no. 1, pp. 151–159, March 1973.

[31] K. N. Olsen and C. J. Stevens, "Perceptual overestimation of rising intensity: is stimulus continuity necessary?," *Perception*, vol. 39, no. 5, pp. 695–704, May 2010.

[32] F. Avanzini, L. Mion, and S. Spagnol, "Personalized 3D sound rendering for content creation, delivery, and presentation," in *NEM Summit 2009*, Saint-Malo, France, September 2009, pp. 12–16.

BLOCK-ORIENTED MODELING OF DISTORTION AUDIO EFFECTS USING ITERATIVE MINIMIZATION

Felix Eichas, Stephan Möller, Udo Zölzer

Department of Signal Processing and Communications,
Helmut-Schmidt-Universität
Hamburg, Germany
felix.eichas@hsu-hh.de

ABSTRACT

Virtual analog modeling is the process of digitally recreating an analog device. This study focuses on analog distortion pedals for guitarists, which are categorized as stompboxes, because the musician turns them on and off by stepping on the switch. While some of the current digital models of distortion effects are circuit-based, this study uses a signal-based approach to identify the device under test (DUT). An algorithm to identify any distortion effect pedal in any given setting by input-output (I/O) measurements is proposed. A parametric block-oriented `Wiener-Hammerstein` model for distortion effects and the corresponding iterative error minimization procedure are introduced. The algorithm is implemented in `Matlab` and uses the `Levenberg-Marquardt` minimization procedure with boundaries for the parameters.

1. INTRODUCTION

Since the first distortion stompbox had been introduced in the 1960s, these effects became very popular amongst guitarists. Some are willing to pay horrendous prices for original vintage effect pedals, others have a huge collection of distortion effects. With the aid of system identification these devices can be digitally reproduced by virtual analog modeling, providing all advantages of digital systems. Identifying analog distortion circuits and building circuit based models to capture their characteristics has widely been done in the context of virtual analog modeling.

In [1–5] circuit based approaches were used to model distortion effects. Nodal analysis is used to derive a state-space-system describing the original circuit. The state-space-system is extended to be able to handle nonlinear circuit elements. However, complete knowledge of the circuit-schematics and all characteristics of the nonlinear elements are required for this method to be applied. If the circuit-schematic of a certain device is not accessible, expensive reverse-engineering and high quality measurements would be needed to derive a digital model. Therefore a simple technique, based on input-output (I/O) measurements would be desirable to get a quick snapshot of the DUT's characteristics. An approach based on I/O measurements was already used in [6–8]. The authors use a modified swept-sine technique, originally introduced by [9], to identify an overdrive effect pedal, which is described by a block-oriented Hammerstein model. Unfortunately, the results from [6–8] could not be reproduced accurately enough from the information given in the paper for a detailed comparison.

To the authors knowledge, there does not exist an objective measure which describes the perceptual correlation between digital model and reference system output. Most of the current objective metrics to evaluate audio content, like PEAQ [10], were designed to rate the sound degradations of low bit-rate audio codecs. This work does not focus on finding an acceptable error metric but designing a proper parametric model and the corresponding identification procedure for modeling of distortion effects. The optimization is based on iterative error minimization between the parametric, block-oriented Wiener-Hammerstein model and the DUT.

In 2008 Kemper introduced a patent describing his model and identification routine for identifying nonlinear guitar amplifiers. He uses a block-oriented Wiener-Hammerstein model to emulate the characteristics of an analog guitar amplifier by analyzing the statistical distribution of pitches and volumes of the identification signal. The filters and the nonlinearity are identified by an identification procedure of his own design, analyzing small and high signal levels separately [11].

This paper is structured as follows. The model is described in Section 2. Section 3 explains the identification process. The results are described in Section 4 and Section 5 concludes this paper.

2. THE MODEL

The basic idea behind the model used in this study is to have a parametric model, which is flexible enough to adapt to many distortion effects but still simple enough to be computationally efficient. The structure of a distortion effect can be described by a Wiener-Hammerstein model. This model consists of linear-time-invariant (LTI) blocks and a nonlinear block. The blocks are ordered in series where the nonlinear block is lined by two LTI blocks. The LTI blocks are filters, which are shown in Fig. 1 as H_1

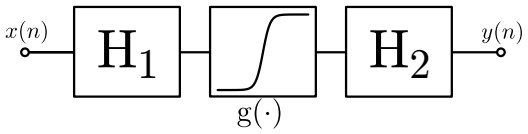
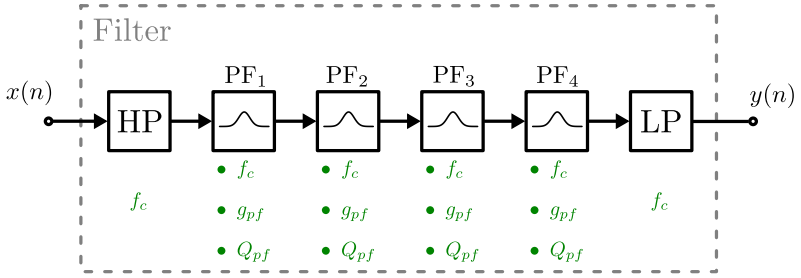


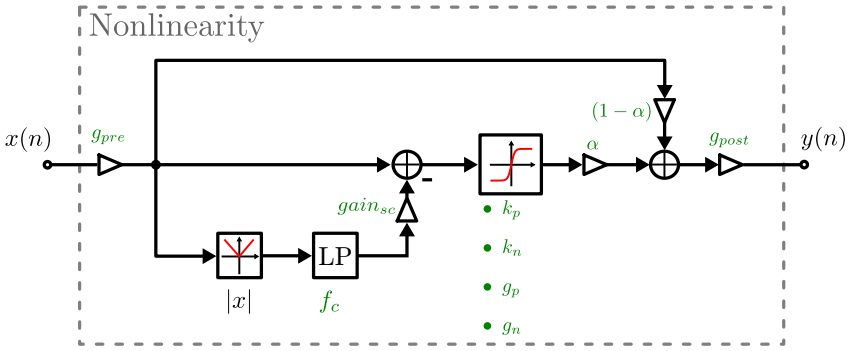
Figure 1: *Block diagram of a Wiener-Hammerstein model.*

and H_2, and the nonlinear block is basically a mapping function, mapping the level of the input signal to an output level, according to the nonlinear function $g(\cdot)$, which simulates the nonlinear behavior of the distortion effect. $x(n)$ denotes the input and $y(n)$ the output signal.

Block-oriented Wiener-Hammerstein models are successfully used in commercial products due to their flexibility and expandability. Fractal Audio Systems calls a Wiener-Hammerstein sys-



(a) *Block diagram of one filter.*



(b) *Block diagram of the nonlinearity.*

Figure 2: *Diagrams for the single blocks of the Wiener-Hammerstein model.*

tem consisting of two filters and a distorting nonlinearity the "fundamental paradigm of electric guitar tone" [12]. They extend this model with more linear and nonlinear blocks to include frequency responses of speakers and model the different nonlinear stages (preamp, power amp) of an analog guitar amplifier.

2.1. LTI Blocks

The filters of the Wiener-Hammerstein model were designed to be flexible but still stay stable in the identification process. Hence, the parameters which will be varied during optimization are not the coefficients of a Direct Form II filter structure, because the identification procedure will not converge if the filter coefficients yield an unstable filter. Instead, they are expressed as the parameters of a simple band-limited equalizer. Figure 2 (a) shows the structure of one LTI block of the Wiener-Hammerstein model. The input signal $x(n)$ is processed by a high-pass filter, a series of four peak filters and finally by a low-pass filter which yields the output signal $y(n)$. The adjustable parameters of the filters are expressed in terms of cutoff frequency f_c for the low-pass and high-pass filters. The peak filters can also be modified in terms of gain and Q-factor. All filters are second order IIR filters and their coefficients are computed according to [13].

All parameters are aligned in a parameter vector $\mathbf{p}_{\text{lti}} = [f_{c,hp}, f_{c,pf1}, g_{pf,1}, Q_{pf1}, \ldots, f_{c,lp}]^T$.

2.2. Nonlinear Block

The nonlinear block of the Wiener-Hammerstein model is shown in Fig. 2 (b). The input signal $x(n)$ is multiplied with a pre-gain g_{pre} and then fed into the lower side-chain which acts as a very simple envelope detector. The absolute value of the signal is computed, low-pass filtered and multiplied with the side-chain gain $gain_{sc}$. This signal is then subtracted from the direct path after the pre-gain multiplication. Subsequently it is fed into the actual mapping function, which is controlled by four parameters. Thereafter the signal can be mixed with the pre-amplified input signal via the parameter α. Finally the signal is multiplied by the post-gain g_{post} to yield the output signal $y(n)$. All parameters for the nonlinear block are aligned in the parameter vector $p_{nl} = [g_{pre}, f_c, g_{sc}, k_p, k_n, g_p, g_n, \alpha, g_{post}]^T$.

The side-chain envelope detector was added to the nonlinear block because the used mapping function is memoryless. But this is not adequate to model the time-variant behavior of e.g. vacuum tube distortion circuits. The bias-point of a vacuum tube amplification stage changes slightly, depending on the signal history, which leads to a DC-offset of the output, amongst other effects. This is emulated by subtracting the lower side-chain envelope signal from the direct path, before the mapping function is applied [14].

2.2.1. *Mapping Function*

The flexibility of the nonlinear block is based on the mapping function. If we would chose a mapping function based on one hyperbolic tangent, it would not be very flexible. In some cases a moderate slope around the zero-crossing of the mapping function and, at the same time, a sharp transition into the saturated region are required, because it resembles the behavior of analog circuit components. A simpler mapping function could not model that behavior accurately. Therefore, we propose an alternate mapping function,

consisting of three hyperbolic tangents which are concatenated at the location denoted by k_p for the positive part of input levels and at k_n for the negative part. The tanh functions above or respectively below k_p and k_n are modified so that they have the same slope as the middle part of the function at the connection points, shown in Eq. 1.

$$m(x) = \begin{cases} \tanh(k_p) - \left[\frac{\tanh(k_p)^2-1}{g_p}\tanh(g_p x - k_p)\right] & \text{if } x > k_p \\ \tanh(x) & \text{if } -k_n \leq x \leq k_p \\ -\tanh(k_n) - \left[\frac{\tanh(k_n)^2-1}{g_n}\tanh(g_n x + k_n)\right] & \text{if } x < -k_n \end{cases} \quad (1)$$

The parameters g_p and g_n control the smoothness of the transition between saturated region and linear center part. For high values of g_p and g_n the transition is very sharp, for low values it is smooth and for very small values and correctly chosen values of k_x and k_p it behaves like a linear function.

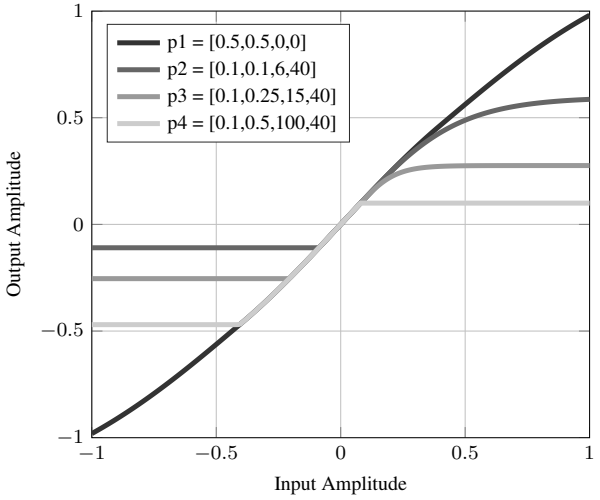


Figure 3: *Modified tanh function with the parameters $p = [k_p, k_n, g_p, g_n]$. Gain values for g_p and g_n are in dB.*

Figure 3 illustrates the modified tanh function. The darkest curve for parameter set $p1$ shows a nearly linear mapping function with relatively high values for $k_n > 0.5$ and $k_p > 0.5$ and low values for $g_n < 1\,\text{dB}$ and $g_p < 1\,\text{dB}$. For positive input amplitudes $p2$, $p3$ and $p4$ have steadily increasing gain values g_p and the same connection point $k_p = 0.1$. This changes the shape of the nonlinear mapping function. The gain g_n was kept constant at $g_n = 40$ dB for negative input amplitudes, while the connection parameter k_n was shifted for $p2$, $p3$ and $p4$. Positive and negative sections of each curve could be interchanged by simply changing the corresponding gain and connection parameters.

3. IDENTIFICATION

The concept of iterative error minimization is shown in Fig. 4. The same input signal is sent through the digital model and the reference system, $\mathbf{y}(n)$ denotes the desired output from the DUT, while $\hat{\mathbf{y}}(\mathbf{p}, n)$ denotes the model output, which is not only dependent on the input samples, but also on a set of parameters $\mathbf{p}$, which were introduced in Sec. 2. If the model is nonlinear for at least one

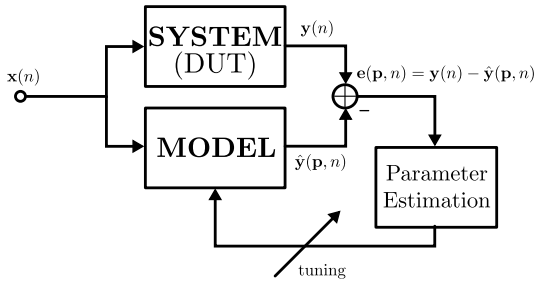


Figure 4: *Block diagram of iterative error minimization.*

parameter, it has to be identified iteratively [15]. The error signal $\mathbf{e}(\mathbf{p}, n) = \mathbf{y}(n) - \hat{\mathbf{y}}(\mathbf{p}, n)$ is calculated by subtracting the model output from the reference output. The parameter estimation algorithm calculates a new set of parameters, which are applied to the model in order to minimize the error between digital model and analog system according to a cost-function C, in our case least-squares, $C(\mathbf{p}) = \sum_{n=1}^{N} \mathbf{e}(\mathbf{p}, n)^2$. Where N is the length of the input signal in samples. The parameter estimation method used in this work is the Levenberg-Marquardt algorithm. This algorithm combines the advantages of gradient-descent and Gauss-Newton method [16, 17].

Before the identification procedure can be started, the reference signals need to be recorded. For this purpose a high quality audio interface (RME Fireface UC ™) was used, which is controlled via `Matlab`. The DUT is placed between output and input of the audio interface. Before the actual input signals are send through the DUT, the interface is calibrated by sending a test signal $x(n) = \sin\left(2\pi \frac{f_0}{f_s} n\right)$ with an amplitude of 0 dBFS, while the DUT is in bypass mode. The fundamental frequency is $f_0 = 1\,\text{kHz}$. The output gain of the interface was adjusted, so that an amplitude of 1 corresponds to 1 V at the output of the audio interface.

3.1. Nonlinear Identification

There are two different input signals for the identification of the linear and nonlinear parts of the Wiener-Hammerstein model. The input signal,

$$x_{nl}(n) = a(n) \cdot \sum_{i=1}^{M} \sin\left(2\pi \cdot \left[\mathbf{f_i} \cdot \sin\left(2\pi \frac{\mathbf{f_{mod,i}}}{f_s} n\right)\right]\right), \quad (2)$$

for the identification of the nonlinear block is created by summing several sine waves with different fundamental frequencies. Where $a(n)$ is a scaling function, creating a logarithmically rising amplitude from $-60\,\text{dBFS}$ to $0\,\text{dBFS}$ to emphasize lower signal levels. $\mathbf{f} = (50, 100, \ldots, 900, 1000)\,\text{Hz}$ is a vector containing all the desired frequencies that roughly cover the range of notes that can be played on a guitar and $\mathbf{f_{mod,i}}$ is a vector containing the modulation frequencies, which range from $f_{\text{mod,min}} = 1\,\text{Hz}$ to $f_{\text{mod,max}} = 10\,\text{Hz}$ and are used to avoid destructive and constructive interference which affect the envelope of the signal. M is the amount of sine waves which are summed and then normalized to achieve a maximum amplitude of 1.

When the parameters of a nonlinear model are optimized by an iterative error minimization algorithm, like the Levenberg-Marquardt

method, it is very important to have an appropriate set of initial parameters. The minimization algorithm might converge into a local minimum if the wrong set of initial parameters is chosen. Thus, every possible combination of the parameters g_{pre}, k_p, k_n, g_p and g_n is tested on a coarse grid by calculating the sum of squares $C(\mathbf{p})$ for each combination. The set with the lowest sum of squares is used as the initial parameter set for the identification process.

Because the parameters g_{pre}, k_p, k_n, g_p and g_n have the most influence on the envelope of the input signal, this procedure helps finding a starting point, which is most likely to converge into the global minimum of the cost function. During the iterative error minimization, only the parameters of the parameter vector $\mathbf{p}_{nl}$ are optimized. The parameters of the two LTI blocks are fixed and can not be changed during this identification step. The parameters in the $\mathbf{p}_{lti}$ vectors are set to yield a neutral filter characteristic in the audible frequency range for both filters. After the optimization $\mathbf{p}_{nl}$ is saved for further use.

3.2. Filter Identification

The input signal for the parameter optimization of the LTI blocks is white noise with signal levels below -50 dBFS, because we assume, that for low signal levels the nonlinear part of the reference system operates in its linear region and $\tanh(x) \approx x \quad \text{for} \quad |x| \ll 1$ is also true for the nonlinear block of the digital model.

In this case optimization of time domain error signals is challenging because the signal can look different in time domain, due to deviating phase characteristics of simulation and reference system, but is still perceived as similar for the human ear. For this reason the output signals $\mathbf{y}(n)$ and $\hat{\mathbf{y}}(\mathbf{p}, n)$ need special treatment before the actual minimization procedure can be started.

First the saved parameters from the identification of the nonlinear block are loaded and used during filter parameter optimization. This helps identifying both filters of the model. If the filter's parameters would be identified before the nonlinear parameters in the first optimization step, characteristics of the first filter could be optimized onto the second filter and vice versa even though the overall frequency response is retained.

The time domain output sequence for the white noise identification input is recorded. The power spectral density (PSD) of the output is computed by calculating a 16384-point discrete Fourier transform (DFT) with a hop size of 4096 samples. All calculated spectra are averaged and multiplied by its complex conjugate to yield the PSD. But the frequency resolution of a PSD or DFT respectively does not correspond to the perceptual resolution of the human ear. For this reason the semi-tone spectrum is calculated from the PSD by averaging the frequency bins corresponding to one note. The first note is A0 in the sub contra octave which has a frequency of $f_{0,A0} = 27.5$ Hz, which is one note below the lowest note on a standard tuning 5-string bass. This is done for reference and digital model respectively before the error signal is calculated.

The initial values for the identification procedure are chosen in such a way that the filter is flat and the cutoff frequencies of high- and low-pass filters are set to $f_{c,HP} = 10$ Hz and $f_{c,LP} = 18$ kHz. After the filter parameters are adapted, they are stored for further use.

3.3. Overall Identification

In this final step the stored parameter vectors for both LTI blocks and the nonlinear block are loaded and used as the initial parameter set in the final parameter vector

$$\mathbf{p} = \begin{pmatrix} \mathbf{p}_{\text{lti1}} \\ \mathbf{p}_{\text{lti2}} \\ \mathbf{p}_{\text{nl}} \end{pmatrix}. \tag{3}$$

The Levenberg-Marquardt algorithm is started and now all parameters of the model can be modified to refine the results from the previous optimization runs. The input signal for this step is the same as for the optimization of the nonlinear parameters, described in Subsec. 3.1. The identification is not done solely in time domain. This approach helps finding an initial parameter set for the error minimization which is likely to converge into the global minimum of the cost function. The spectrogram (8192-point DFT, 4096 hop size) of the output of the analog system and the digital simulation is calculated and then vectorized as the minimization algorithm is not able to process error signals which are not in vectorial form. By optimizing over the spectrogram error signal the necessary information for adapting the filter and nonlinear parameters is included.

4. RESULTS

As stated in Subsec. 3.2 it is challenging to find an objective error measure for the perceived differences between to audio signals. For this reason the error between simulation and reference is shown in time-domain as well as frequency-domain.

4.1. Time-Domain Error

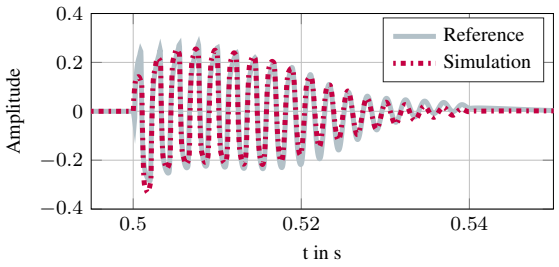
A possible objective error measure would be the time domain error,

$$e_{y\hat{y}} = \frac{\sum_{n=1}^{N} (y(n) - \hat{y}(\mathbf{p}, n))^2}{N}, \tag{4}$$

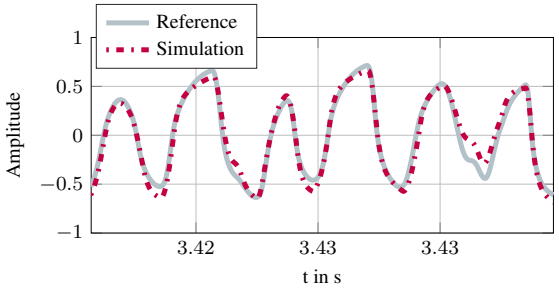
where N is the overall length in samples of the output signals. The error becomes zero if the signals are completely the same. But in certain cases, the time domain error is quite high but the perceived (subjective) difference is hard to hear. In general however, the time domain error gives us an estimate about how close the model can recreate the DUT. Nevertheless this is no reliable metric to characterize the perceptual difference between two audio signals. Figure 5 shows the comparison between time domain signals of DUT and the identified model. The DUT was a *Hughes & Kettner* - Tube Factor, which is basically a tube-preamp in stompbox format with a 12AX7 vacuum tube. Figure 5 (a) shows the response of digital model and reference system to an exponentially decaying sine input with 440 Hz. The maximum amplitude of the input was 1. The digital model reproduces the analog system quite well, except from the transient part at the beginning of the signal. For the recorded electrical guitar signal, depicted in Fig. 5 (b), the simulation follows the measured curve closely but there are still some differences for certain frequency components, which may be a result of the simple nonlinear block, where one nonlinear function is used for all frequency components of the input signal. Another way of determining how well the identification worked, is by comparing the transfer functions of both, reference and simulation.

4.2. Time and Frequency-Domain Error

Figure 6 shows the frequency response of a *Jim Dunlop* - Fuzz Face fuzz pedal in comparison to the frequency response of its



(a) 40 ms 440 Hz *sin with exponentially decaying amplitude.*



(b) *Excerpt from recorded guitar signal.*

Figure 5: *Comparison of the time-domain signals for the Hughes & Kettner Tube Factor and the identified model.*

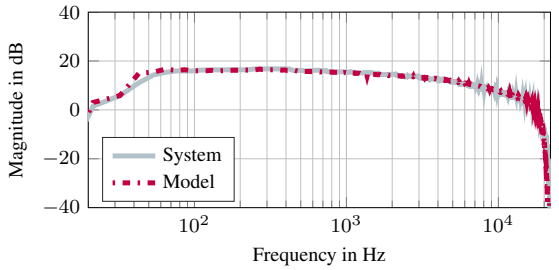
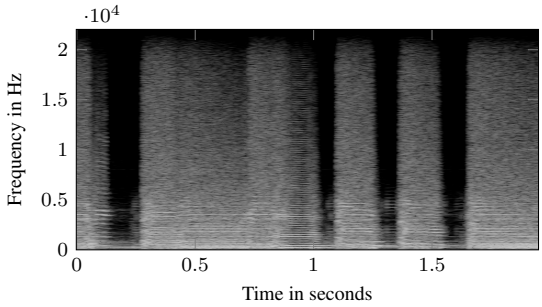
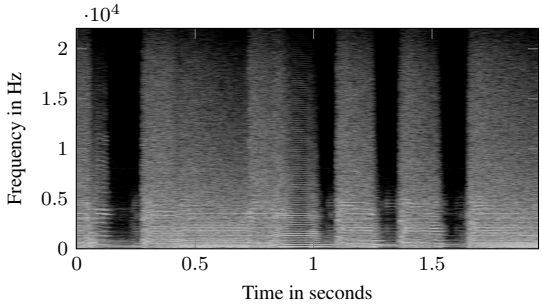


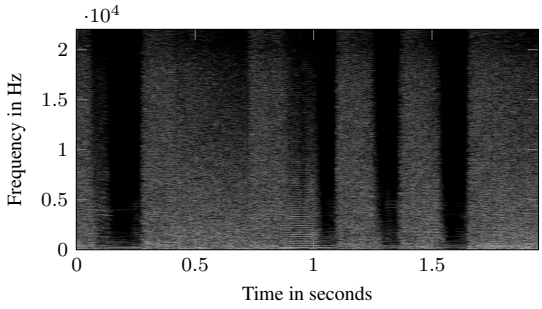
Figure 6: *Comparison of the frequency response of analog reference and digital model for a Jim Dunlop - Fuzz Face.*



(a) Reference



(b) Simulation



(c) Error

Figure 7: *Spectrograms of reference system and digital model output and the error spectrogram.*

digital model. The model deviates from the reference system by less than 1 dB, but becomes more inaccurate for low frequencies (below 60 Hz) and for frequencies above 18 kHz.

The spectrograms of digital model and analog system output are shown in Fig. 7. The input to both reference system and digital model was a recording of an electrical guitar playing fast high chords, as customary for funk music. The DUT was a *Hughes & Kettner* - Tube Factor. The harmonics generated by the DUT and its digital representation resemble each other well. The error spectrogram, shown in Fig. 7 (c), shows the difference of the absolute values of the previous spectrograms. The overall error energy is much lower than the signal energy of reference system and digital model.

4.3. Aliasing

The sampling frequency was set to $f_s = 48$ kHz for identification and runtime operations. The upper plot of Fig. 8 shows the frequency response of the digital model to a 1500 Hz sine wave. The aliasing, caused by the nonlinear block of the Wiener-Harmmerstein model is clearly visible due to the distinct peaks between the main peaks for fundamental frequency and the harmonics. For this reason resampling was introduced. The output signal of the first LTI block is upsampled by resampling factor L, then processed by the nonlinear block and finally the output of the nonlinear block is downsampled by factor L. The lower plot of Fig. 8 shows the response of the digital model to a 1500 Hz sine wave with resampling factor $L = 8$. The effects of aliasing are now nearly diminished.

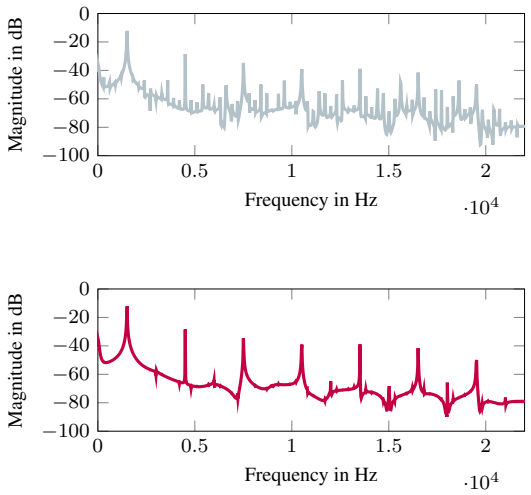


Figure 8: *Response of the digital model to a* 1500 *Hz sine. Above: without oversampling. Below: with* 8 *times oversampling.*

4.4. Auditory Impression

Although no formal listening test was conducted, the subjective auditory impression of the proposed model is quite satisfying. In some cases the difference between simulation and reference output is still audible but only for a trained listener. Different input signals as well as digital model and analog system outputs can be found online. Please visit [18] for listening examples.

5. CONCLUSIONS

This work proposed a method to identify and model nonlinear analog distortion effects. LTI filter blocks and a nonlinear block of a Wiener-Hammerstein model, are introduced. The identification routine is described and the model is able to emulate any distortion pedal in a given setting. For many effects the results from the model are nearly indistinguishable from the analog device itself. But this method still has several drawbacks, which should be addressed in the future. First the search for the initial parameter set is still carried out on a coarse grid, because the computational effort rises drastically if the grid resolution or the amount of tested parameters increases. This may cause the identification algorithm to converge into a local minimum instead of the global minimum. Furthermore it is essential to study more perceptually motivated error metrics, e.g. PEAQ, to find a comparable and reliable error metric.

6. REFERENCES

[1] D. Yeh, J.S. Abel, and J.O. Smith, "Automated physical modeling of nonlinear audio circuits for real-time audio effects: Part 1 - theoretical development," in *IEEE Trans. Audio, Speech, and Language Process.*, May 2010, vol. 18, pp. 203–206.

[2] D. Yeh and J.O. Smith, "Simulating guitar distortion circuits using wave digital and nonlinear state-space formulations," in *Proc. Digital Audio Effects (DAFx-08)*, Espoo, Finland, Sept. 1-4, 2008, pp. 19–26.

[3] J. Macak, *Real-time Digital Simulation of Guitar Amplifiers as Audio Effects*, Ph.D. thesis, Brno University of Technology, 2011.

[4] K. Dempwolf, *Modellierung analoger Gitarrenverstärker mit digitaler Signalverarbeitung*, Ph.D. thesis, Helmut-Schmidt-Universität, 2012.

[5] M. Holters and U. Zölzer, "Physical modelling of a wah-wah effect pedal as a case study for application of the nodal dk method to circuits with variable parts," in *Proc. Digital Audio Effects (DAFx-11)*, Paris, France, Sept. 19-23, 2011, pp. 31–35.

[6] A Novak, L. Simon, P. Lotton, and J. Gilbert, "Chebyshev model and synchronized swept sine method in nonlinear audio effect modeling," in *Proc. Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 6-10, 2010.

[7] A. Novak, L. Simon, F. Kadlec, and P. Lotton, "Nonlinear system identification using exponential swept-sine signal," *Instrumentation and Measurement, IEEE Trans. on*, vol. 59, no. 8, pp. 2220–2229, 2010.

[8] R. Cauduro Dias de Paiva, J. Pakarinen, and V. Välimäki, "Reduced-complexity modeling of high-order nonlinear audio systems using swept-sine and principal component analysis," in *Audio Engineering Society Conference: 45th International Conference: Applications of Time-Frequency Processing in Audio*, Mar 2012.

[9] A. Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique," in *Audio Engineering Society Convention 108*, Paris, France, 2000.

[10] International Telecommunication Union, "Bs.1387: Method for objective measurements of perceived audio quality," url, Available online at `http://www.itu.int/rec/R-REC-BS.1387` — accessed June 1st 2015.

[11] C. Kemper, "Musical instrument with acoustic transducer," US Patent: US20080134867 A1, July 2008.

[12] Fractal Audio Systems, "Multipoint Iterative Matching and Impedance Correction Tevhnology (MIMIC™)," Tech. Rep., Fractal Audio Systems, April 2013.

[13] U. Zölzer, *DAFx - Digital Audio Effects*, John Wiley and Sons, 2011.

[14] J. Pakarinen and D. T. Yeh, "A review of digital techniques for modeling vacuum-tube guitar amplifiers," *Computer Music Journal*, vol. 33, no. 2, pp. 85–100, 2009.

[15] T. Strutz, *Data fitting and uncertainty: a practical introduction to weighted least squares and beyond*, Vieweg and Teubner, 2010.

[16] K. Levenberg, "A method for the solution of certain problems in least squares," *Quarterly of applied mathematics*, vol. 2, pp. 164–168, 1944.

[17] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial & Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[18] "Listening Examples," Website, Available at `http://www2.hsu-hh.de/ant/webbox/audio/eichas/dafx15/audio_examples_local_version.html` — accessed June 4th 2015.

APPROXIMATING NON-LINEAR INDUCTORS USING TIME-VARIANT LINEAR FILTERS

Giulio Moro
Centre for Digital Music
Queen Mary University of London
London, United Kingdom
g.moro@qmul.ac.uk

Andrew P. McPherson
Centre for Digital Music
Queen Mary University of London
London, United Kingdom
a.mcpherson@qmul.ac.uk

ABSTRACT

In this paper we present an approach to modeling the non-linearities of analog electronic components using time-variant digital linear filters. The filter coefficients are computed at every sample depending on the current state of the system. With this technique we are able to accurately model an analog filter including a non-linear inductor with a saturating core. The value of the magnetic permeability of a magnetic core changes according to its magnetic flux and this, in turn, affects the inductance value. The cutoff frequency of the filter can thus be seen as if it is being modulated by the magnetic flux of the core. In comparison to a reference non-linear model, the proposed approach has a lower computational cost while providing a reasonably small error.

1. INTRODUCTION

This work investigates how it is possible to use linear, time variant filters in order to introduce non-linearities in a digital signal processing system. The idea is to use time-varying infinite impulse response filters whose coefficients are updated at every time sample according to the state of the system at the previous time sample. This approach is applied here to solve electronic circuits with non-linear components in the digital domain and can be used as a building block for Virtual Analog applications.

Non-linear DSP systems are governed by non-linear equations that have to be solved iteratively at a non-negligible computational cost [1, 2]. Adding a new non-linear equation to an existing system increases the computational cost even further and requires sometimes to partially re-design the existing system [3]. Non-iterative ways to solve such systems have been proposed which rely on pre-computed tables [4]. The lookup tables method does not scale up easily to systems with multiple non-linearities as this increases the dimensionality of the table, increasing memory usage and computational cost.

The method presented in this paper replaces the non-linear equations with a time-varying linear filter, which is by itself less expensive in terms of computations and can, potentially, be expanded to replace systems with multiple non-linearities with higher-order filters. We do not expect the output of our approximated model to be a sample-by-sample replica of the results obtainable with more accurate simulations, but we expect it to be close enough that the loss in accuracy is justified by improvements in execution speed and scalability.

While general criteria that determine the stability of stationary recursive filters are well defined in the literature [5], criteria to assess the stability of time-variant linear filters can be studied and defined only under some specific conditions. Existing work mainly focuses on bounded input-bounded output stability [6, 7] and transient suppression [8]. However, these methods are not readily applicable when filter coefficients are changing at every sample. Recent work proves that stable time-varying behavior can be obtained using state variable filters [9]. However, as in this paper we deal with passive first-order filters, we chose to use a different filter topology.

In [10], time-varying coefficients are used to introduce a clipping function in the feedback loop of an IIR filter, in order to reproduce the behaviour of analog voltage controlled filters. In [11] it is shown that this method affects the frequency response of a resonant filter by increasing its bandwidth and moving its centre frequency. The use of IIR coefficients varying on a sample-by-sample basis has been exploited previously in [12] where feedback amplitude modulation is used for sound synthesis and in [13] where time-varying fractional delays are used to model non-linear vibrating strings.

In this paper we will present a physically-informed model for an inductor, with its characteristic non-linearity caused by the saturation of the magnetic core [14]. Non-linear differential equations and a state-space model to solve the non-linear transformer are presented in [15], whereas a Wave Digital Filter approach can be found in [16].

From a physical standpoint, the saturation of the core in an inductor affects the present value of its inductance. From this consideration we will build an infinite impulse response (IIR) linear filter whose coefficients are updated at every time step using the actual value of the inductance given by the current saturation state of its core. This will produce a delay-free loop which we will address using a variation on the classic 1-sample delay approach, widely used in the literature ([17, 18]), and linearizing the system around the operating point. D'angelo recently discussed the linearization of a non-linear system around an operating point to solve a transistor ladder filter [19], generalizing the delay-free loops resolution method in [20].

The physics of the non-linear inductor is reviewed in Section 2. Section 3 will present a reference non-linear discrete-time model for the inductor which will be used to evaluate the approximated model presented in Section 4. Results and discussion follow in Section 5 and Section 6 respectively.

2. PHYSICS OF INDUCTORS

An inductor is a passive component with inductive behavior, usually built using a coil of wire winded on a core made of ferromagnetic material, such as ferrite. While inductors are often modeled as linear components, most real inductors exhibit non-linear

behavior caused primarily by the progressive saturation of their ferromagnetic core. The distortion caused by the core occurs primarily for signals with large currents and low frequencies.

We present here a simplified model for the inductor which models the saturation of the core but does not take into account losses, hysteresis and parasitic parameters.
From Faraday's laws we have, for a solenoid:

$$\frac{dB}{dt} = \frac{V_L}{NS} \tag{1}$$

where B is the magnetic flux density in the inductor core, V_L is the voltage across the inductor, S is the area of the section of the core and N is the number of turns in the inductor. Ampere's law gives the magnetizing force H for a solenoid traversed by a current I_L as: [21]

$$H = \frac{NI_L}{l} \tag{2}$$

and l is the length of the induction path. In an ideal inductor there is a linear relation between the flux density and the magnetizing force:

$$B = \mu H \tag{3}$$

where μ is the absolute magnetic permeability of the core, defined as:

$$\mu = \mu_0 \mu_i \tag{4}$$

where μ_0 is the vacuum permeability and $\mu_i \geq 1$ is the relative permeability of the magnetic core. In the case of a ferromagnetic core, however, the magnetic flux density cannot be increased above a certain value. This value is called magnetic flux density saturation and depends on the material and geometry of the core.

For low flux density levels, Eq. (3) is valid and the inductor can be considered as a linear component. As the core approaches the saturation level B_{sat}, the relation between H and B becomes non linear, the magnetic characteristics of the core change from those of a ferromagnetic material to those of a paramagnetic material and the value of μ progressively changes from being $\mu = \mu_i \cdot \mu_0$ when $B = 0$ to being approximately $\mu = \mu_0$ when $B = B_{sat}$. The Fröhlich-Kennelly relation gives the following relation between B and H for a ferromagnetic core: [22]

$$B = \frac{H}{c + b|H|} \tag{5}$$

where b and c are defined as

$$b = \frac{1 - \sqrt{\frac{1}{\mu_i}}}{B_{sat}}$$
$$c = \frac{1}{\mu_0 \mu_i}$$

The B-H relation described by these formulas is shown in Fig. 1. For small values of $|H|$ and/or large values of B_{sat}, Eq. (5) is equivalent to Eq. (3), thus explaining the linear behaviour at low currents.
The inductance L of a solenoid is derived by Ampere's law as: [21]

$$L = \frac{\mu N^2 S}{l} \tag{6}$$

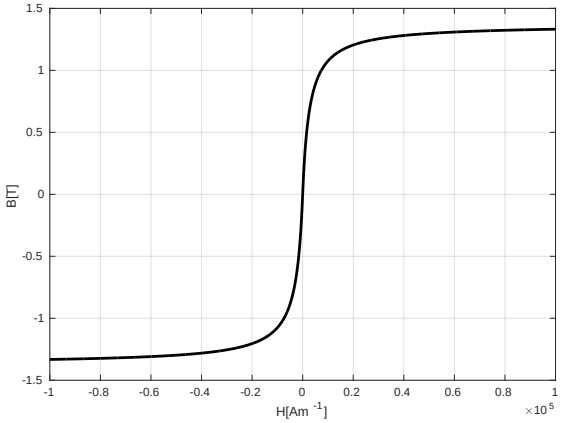


Figure 1: Anhysteretic B-H curve according to the Fröhlich model for an inductor with a ferrite core ($u_i = 400$, $B_{sat} = 1.3$)

where N, S, l, μ are the physical parameters of the inductor described above. As denoted by Eqs. (3) and (5), μ is not constant and its value can drop by several orders of magnitude as the core progressively saturates and this is reflected directly on the value of the inductance through Eq. (6).

3. DISCRETIZATION OF THE NON-LINEAR INDUCTOR

In order to solve a circuit including a non-linear inductor in the discrete-time domain, Eqs. (1), (2) and (5) have to be discretized. This is straightforward for Eqs. (2) and (5):

$$H[n] = \frac{N}{l} I_L[n] \tag{7}$$

$$B[n] = \frac{H[n]}{c + b|H[n]|} \tag{8}$$

while Eq. (1) requires an integration formula. The solution to an equation of the form:

$$\frac{dx}{dt} = f(x, t) \tag{9}$$

is given in the discrete-time domain by the backward Euler formula as: [23]

$$x[n] = x[n-1] + Tf(x[n]) \tag{10}$$

where T is the sampling period of the discrete-time system. This formula, when applied to Eq. (1), yields:

$$B[n] = B[n-1] + \frac{T}{NS} V_L[n] \tag{11}$$

Combining Eqs. (7), (8) and (11), we obtain:

$$V_L[n] = \frac{NS}{T} \left(\frac{\frac{N}{l} I_L[n]}{c + \frac{bN}{l} |I_L[n]|} - B[n-1] \right) \tag{12}$$

Where $V_L[n]$ is the voltage across an inductor at a time instant n given the current through it $I_L[n]$ and the magnetic field at the previous time instant $B[n-1]$.

3.1. High pass filter

We now consider the circuit in Fig. 2. In this circuit low frequencies from the input V_i will find an easier path to ground through the inductor than high frequencies, therefore, considering node V_o as the output, the circuit will act as a high pass filter. For the circuit in Fig. 2, the current through the inductor is:

$$I_L[n] = \frac{V_i[n] - V_o[n]}{R} \tag{13}$$

while the voltage across the inductor is:

$$V_L[n] = V_o[n] \tag{14}$$

Substituting these values in Eqs. (7), (11) and (12) gives:

$$B[n] = B[n-1] + \frac{T}{NS} V_o[n] \tag{15}$$

$$V_o[n] = \frac{NS}{T} \left(\frac{\frac{N}{l} \frac{V_i[n] - V_o[n]}{R}}{c + \frac{bN}{l} |\frac{V_i[n] - V_o[n]}{R}|} - B[n-1] \right) \tag{16}$$

Combining Eqs. (15) and (16) we obtain the following system equation for the discretized version of the circut in Fig. 2 with a non-linear inductor.

$$V_o^2[n]\beta[n] - V_o[n]\gamma[n] - \delta[n] = 0 \tag{17}$$

with

$$\beta[n] = bNTk[n]$$
$$\gamma[n] = clRT + bNTk[n]V_i[n] + N^2S - bN^2Sk[n]B[n-1]$$
$$\delta[n] = clNRSB[n-1] - bN^2Sk[n]B[n-1]V_i[n] + N^2SV_i[n]$$

in which, k is the sign of the current through the inductor at time instant n, $V_i[n]$ is the voltage input to the system and $B[n-1]$ is computed at each time step using Eq. (15). Eq. (17) is – strictly speaking – a non-polynomial equation as it contains $k = sign(V_o - V_i)$ and should then be solved using iterative numerical approaches (e.g. Newton Method). On the other hand, it can be considered as two distinct second-order polynomials – one with $k = 1$ and one with $k = -1$. Solving these two polynomials will produce four solutions for V_o, of which one and only one will be real and therefore acceptable.

The following schedule can thus be used to find the output $V_o[n]$ of the system for every n:

1. Solve Eq. (17) as explained above to obtain $V_o[n]$. For $n = 0$ assume $B[n-1] = 0$
2. Compute $B[n]$ using Eq. (15)

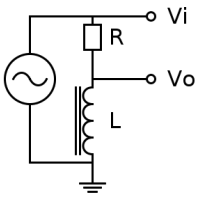


Figure 2: Passive high pass filter

3.2. Low pass filter

We now consider the circuit in Fig. 3. In this circuit high frequencies from the input V_i will be attenuated while passing through the inductor more than low frequencies, therefore, considering node V_o as the output, the circuit will act as a low pass filter. In this case the current through and the voltage across the inductor are, respectively:

$$I_L[n] = \frac{V_o[n]}{R} \tag{18}$$
$$V_L[n] = V_i[n] - V_o[n] \tag{19}$$

Substituting these values in Eqs. (7) and (11) and going through passages similar to those described in Section 3.1 we obtain:

$$V_o^2[n]\epsilon[n] + V_o[n]\zeta[n] + \eta[n] = 0 \tag{20}$$

with

$$\epsilon[n] = bNTk[n]$$
$$\zeta[n] = bN^2Sk[n]B[n-1] - clRT + bNTk[n]V_i[n] - N^2S$$
$$\eta[n] = clNRSB[n-1] + clRTV_i[n]$$

where $k[n] = sign(V_o[n])$ and for which the same considerations made above for the resolution of Eq. (17) are valid.

4. APPROXIMATION OF THE NON-LINEAR INDUCTOR WITH A VARIABLE INDUCTANCE

The reference model presented in Section 3 solves the non-linear electronic circuits proposed using non-linear equations. A different approach is presented in this section which solves the same circuits using time-varying linear filters informed by the physical behaviour of the non-linearity under exam.

In Section 2 we showed that the change in the permeability of the core of an inductor as it approaches saturation affects the effective inductance of the core. The non-linear behaviour is modeled here using a time-varying value for the inductance which is, for each time instant, determined by the current value of the core permeability.

The incremental magnetic permeability of a ferromagnetic material is the rate of change of magnetic flux density with respect to the magnetizing force and is given, in its differential definition, by: [24]

$$\mu_{inc} = \frac{dB}{dH} \tag{21}$$

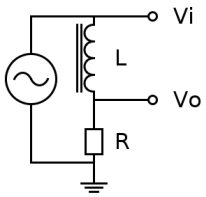


Figure 3: Passive low pass filter

For a real inductor, according to the Fröhlich model, μ_{inc} is given combining this with Eq. (5):

$$\mu_{inc} = \frac{dB}{dH} = \frac{c}{(c + b|H|)^2} \tag{22}$$

Remembering Eqs. (7) and (22), the time-discrete formulation for μ_{inc} is:

$$\mu_{inc}[n] = \frac{c}{(c + \frac{bN}{l}|I_L[n]|)^2} \tag{23}$$

The equation of the inductance of a solenoid as given by Eq. (6) is discretized as:

$$L[n] = \frac{\mu_{inc}[n] N^2 S}{l} \tag{24}$$

Which, combined with Eq. (23), gives:

$$L[n] = \frac{cN^2 S}{l(c + \frac{bN}{l}|I_L[n])^2} \tag{25}$$

4.1. High pass filter

If we consider the inductor in Fig. 2 to be ideal, the cutoff frequency F_c of the filter can be computed from the values of its electronic components as:

$$F_c = \frac{R}{2\pi L} \tag{26}$$

We can discretize the circuit under exam using the well-known bilinear transform. If the component values were time-invariant, the z-transform of the first-order highpass filter would be:

$$H(z) = \frac{2L - 2Lz^{-1}}{RT + 2L + z^{-1}(RT - 2L)} \tag{27}$$

If we now consider the inductor to have a saturating core, the value of L actually changes at every sample, according to Eq. (25). Comparing Eqs. (25) and (26) it emerges that as the current through the inductor increases, the actual inductance value is decreased and consequently the cutoff frequency increases.

As a consequence, the filter coefficients in Eq. (27) will also change over time. Considering the time-varying elements, the finite difference equation for this system is, therefore:

$$V_o[n] = b_0[n]V_i[n] + b_1[n]V_i[n-1] - a_1[n]V_o[n-1] \tag{28}$$

where:

$$\begin{aligned} b_0[n] &= 2L[n]/(RT + 2L[n]), \\ b_1[n] &= -2L[n]/(RT + 2L[n]), \\ a_1[n] &= (RT - 2L[n])/(RT + 2L[n]) \end{aligned} \tag{29}$$

For the circuit in Fig. 2, $L[n]$ depends on the instantaneous current through the inductor, as given by Eq. (13), which, in turn, depends on $V_o[n]$. Therefore $L[n]$ cannot be computed before $V_o[n]$ and it cannot appear in the right hand side of Eq. (28). This constraint leads to an uncomputable loop, also known as delay-free loop [25]. To eliminate the delay free loop we must use an approximate value for $I_L[n]$ which does not depend on $V_o[n]$.

We can estimate a value $V_o'[n] \approx V_o[n]$ by linearizing the output signal around time instant $n-1$ and estimating the value of $V_o'[n]$ using linear extrapolation. Given the discrete-time differentiation of $V_o[n]$

$$\Delta V_o[n] = V_o[n] - V_o[n-1] \tag{30}$$

we can write

$$V_o'[n] = V_o[n-1] + \Delta V_o[n-1] \tag{31}$$

Given a parameter $\alpha \in [0, 1]$, we can define an estimated value $V_o'(n, \alpha)$ for the output voltage at every time instant between $n-1$ and n by linearly interpolating between $V_o[n-1]$ and $V_o'[n]$ as:

$$V_o'(n, \alpha) = \alpha \cdot V_o[n-1] + (1 - \alpha) \cdot V_o'[n] \tag{32}$$

which, for $\alpha = 1$ equals $V_o[n-1]$ and for $\alpha = 0$ equals $V_o'[n]$. In order to appropriately compute the current through the inductor, the input and output voltage must be considered at the same instant in time, therefore we also define $V_i'(n, \alpha)$ as the linear interpolation between $V_i[n-1]$ and $V_i[n]$, parametrized by α:

$$V_i'(n, \alpha) = \alpha \cdot V_i[n-1] + (1 - \alpha) \cdot V_i[n] \tag{33}$$

Now we can compute approximate value for $I_L[n]$ parametrized by α by replacing V_o with $V_o'(n, \alpha)$ and V_i with $V_i'(n, \alpha)$ in Eq. (13):

$$I_{L_\alpha}[n] = \frac{V_i'(n, \alpha) - V_o'(n, \alpha)}{R} \tag{34}$$

4.2. Low pass filter

The z-transform of the low pass filter in Fig. 3:

$$H(z) = \frac{RT + RTz^{-1}}{RT + 2L + z^{-1}(RT - 2L)} \tag{35}$$

We can derive the equations for the filter coefficients similarly to what has been done in the previous paragraph. In the case of a real inductor, the value of $L[n]$ is, again, time-varying and it depends on the current through the inductor $I_L[n]$ for each instant n.

As such current is not known in advance, we need to use an approximate value for $I_L[n]$ when computing the filter coefficients. Analogously to what has been done for the high pass filter in Section 4.1, using the same formulas for linear extrapolation as in Eq. (31) and linear interpolation as in Eq. (32) to obtain $V_o'(n, \alpha)$, we can write the estimated current through the inductor, parametrized by α, by replacing V_o with $V_o'(n, \alpha)$ in Eq. (18):

$$I_{L_\alpha}[n] = \frac{V_o'(n, \alpha)}{R} \tag{36}$$

5. RESULTS

For the evaluation of the approximated saturating inductor model we created a digital model of the high pass circuit in Fig. 2 and solved it using both the model involving non-linear equations described in Section 3, used as a reference, and the approximate model introduced in Section 4. Results for the low pass circuit in Fig. 3 are not explicitly reported here for brevity, but the findings are very similar to those outlined below for the high pass filter.

We performed our tests using the parameter values listed in Table 4 over all of the possible combinations of the following parameters:

Input level [V]	**Input signal** 15Hz	45Hz	89Hz	179Hz	238Hz	953Hz	3810Hz	7620Hz	15240Hz	19050Hz	Noise	Bass	Guitar
1	0.034	0.041	0.041	0.041	0.041	0.041	0.041	0.041	0.041	0.041	0.042	0.041	0.041
10	0.024	0.039	0.041	0.041	0.041	0.041	0.041	0.041	0.041	0.041	0.053	0.046	0.041
50	0.203	0.193	0.026	0.039	0.040	0.041	0.041	0.042	0.042	0.042	0.172	0.115	0.043
100	0.397	0.669	0.572	0.025	0.035	0.041	0.042	0.042	0.042	0.042	0.330	0.256	0.046
200	0.706	1.360	1.750	1.515	0.501	0.041	0.042	0.042	0.043	0.043	0.470	1.130	0.058

Table 1: High-pass filter: frequency-domain RMS error (%) for different signals and input voltages, with $\alpha = 1$ and sampling rate=48kHz

Sampling rate [kHz]	**Input signal** 15Hz	45Hz	89Hz	179Hz	238Hz	953Hz	3810Hz	7620Hz	15240Hz	19050Hz	Noise	Bass	Guitar
12	2.703	4.944	6.032	4.412	0.943	0.165	0.175	-	-	-	3.425	3.740	0.221
24	1.390	2.621	3.295	2.688	0.775	0.082	0.084	0.085	-	-	1.152	2.123	0.114
48	0.706	1.360	1.750	1.515	0.501	0.041	0.042	0.042	0.043	0.043	0.470	1.130	0.058
96	0.356	0.694	0.908	0.811	0.286	0.020	0.021	0.021	0.021	0.021	0.434	0.579	0.030
192	0.179	0.351	0.463	0.421	0.153	0.010	0.010	0.010	0.010	0.010	0.279	0.289	0.015
384	**0.090**	**0.177**	**0.234**	**0.214**	**0.079**	**0.005**	**0.005**	**0.005**	**0.005**	**0.005**	**0.154**	**0.142**	**0.008**

Table 2: High-pass filter: frequency-domain RMS error (%) for different signals and sampling rates with $\alpha = 1$ and input level=200V

α	**Input signal** 15Hz	45Hz	89Hz	179Hz	238Hz	953Hz	3810Hz	7620Hz	15240Hz	19050Hz	Noise	Bass	Guitar
0	1.062	2.072	2.600	2.410	0.904	0.211	1.475	11.821	533.597	533.597	51.414	1.019	0.622
0.25	0.758	1.480	1.891	1.684	0.582	0.153	0.780	5.894	479.974	479.974	45.540	0.620	0.538
0.5	0.538	1.055	1.378	1.135	0.341	0.095	0.250	1.537	336.812	336.812	37.176	**0.485**	0.448
0.75	**0.517**	**1.006**	**1.319**	**1.067**	**0.316**	**0.031**	0.064	0.854	211.559	211.559	20.818	0.742	0.311
1	0.706	1.360	1.750	1.515	0.501	0.041	**0.042**	**0.042**	**0.043**	**0.043**	**0.470**	1.130	**0.058**

Table 3: High-pass filter: frequency-domain RMS error (%) for different signals and values of α with sampling rate=48kHz and input level=200V

1. sampling frequencies:
   12kHz, 24kHz, 48kHz, 96kHz, 192kHz, 384kHz.
2. audio signals:
   (a) Sine waves at frequencies:
   15Hz, 45Hz, 89Hz, 179Hz, 238Hz, 953Hz, 3810Hz, 7620Hz, 15240Hz, 19050Hz, length 10 seconds
   (b) White noise sample, length 10 seconds
   (c) Electric bass guitar sample, length 6.6 seconds
   (d) Electric guitar sample, length 3.1 seconds
3. a set of amplitudes: 1V, 10V, 50V, 100V, 200V
4. a set of values for the linear interpolation parameter α: 0.25, 0.5, 0.75, 1

We skipped tests on sine waves whose frequency was above the Nyquist frequency of the sampling rate. For sampling frequencies of 48kHz and above the result signals have been bandlimited to a maximum frequency of 20kHz before computing error figures. Given a sampling rate, an input signal and an amplitude, the outputs of the approximate model for each different α have been compared to the output of the reference model.

$R[\Omega]$	μ_i	$B_{sat}[T]$	N	$S[cm^2]$	$l[cm]$
100	400	1.3	1000	1	2

Table 4: Physical parameters used in the simulation.

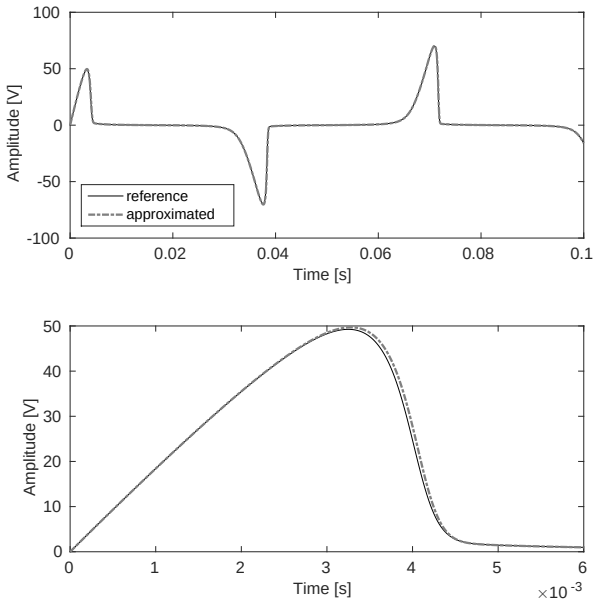


Figure 4: Time domain waveforms for the reference and approximated high pass filter for sampling frequency 48kHz, signal frequency 15Hz, input level 200V, $\alpha = 1$. Large time scale (top) and detail (bottom)

Fig. 4 displays the time domain voltage signal of a 15Hz sinusoid of peak amplitude 200V processed through the high pass filter. The time domain waveforms are very similar. Fig. 4 (bottom) shows that only by zooming in on the time axis we can notice the difference: the drop in voltage caused by the saturation of the core is slightly delayed in the approximated waveform. This can be easily explained considering that in the approximated model the value for the flux density is computed based on the value of the current at the previous time sample, which causes an inherent delay in the response. Fig. 5 shows that the flux density is in fact

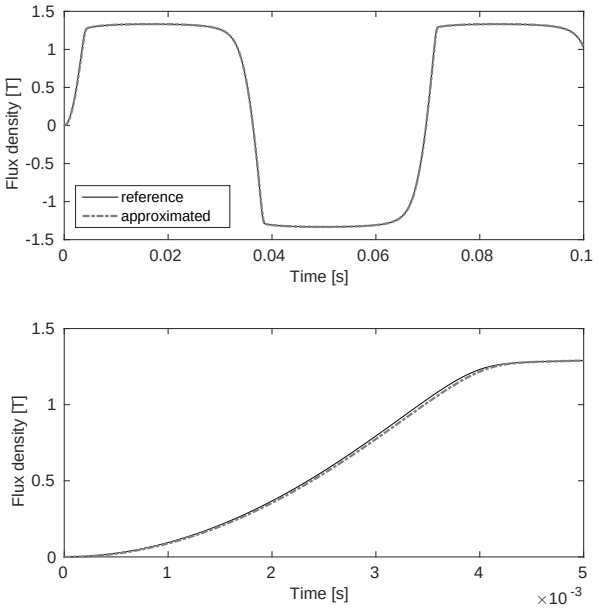


Figure 5: Time domain waveforms of the flux density for the signal in Fig. 4. Large time scale (top) and detail (bottom).

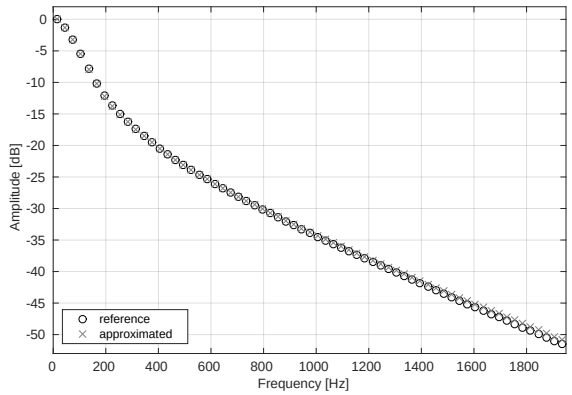


Figure 6: Peaks of the spectrum of the signal in Fig. 4

slightly delayed with respect to the reference.

The RMS error between the two time-domain voltage signals is 2.45%. This rather large error figure is justified by the fact that the discrepancy between the two waveforms occurs around a rapid voltage drop. As human perception of an audio signal is linked more closely to its frequency content than to its time-domain representation, we find it more relevant to reference the THD, THD+N and frequency domain RMS error figures for the purposes of this evaluation.

Fig. 6 displays the reference and approximated signal in the frequency domain. They are very similar, with a total RMS error in the frequency domain as small as 0.71%. While the match is almost perfect for the lower harmonics, a small discrepancy arises

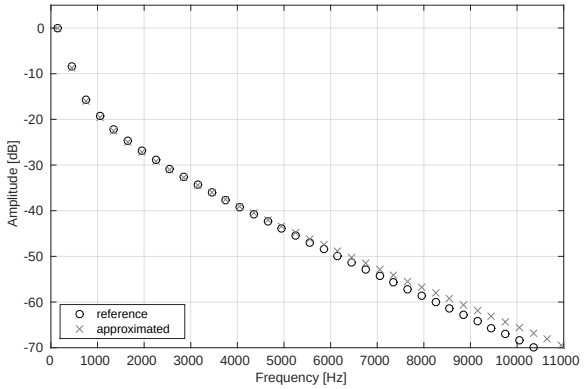


Figure 7: Peaks of the spectrum of a 150Hz sinusoid processed through the high pass filter circuit, sampled at 48kHz

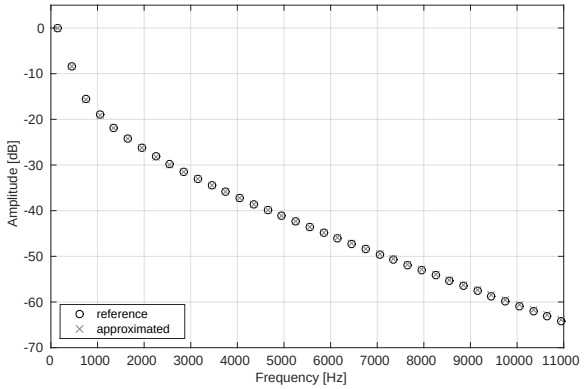


Figure 8: Peaks of the spectrum of a 150Hz sinusoid processed through the high pass filter circuit, sampled at 384kHz

from the 90th harmonic (1350Hz and above). No noise or spurious frequencies have been introduced by the approximated model, as denoted by the fact that the THD and the THD+N values are exactly the same.

Performing the same analysis on a 150Hz signal gives a time-domain RMS error of 8.63% and a frequency-domain RMS error of 1.69%. The spectrum of the signal is shown in Fig. 7. The discrepancies in the spectral amplitudes begin to arise from about 4950Hz (33rd harmonic). Again, no spurious frequencies or noise are added.

By increasing the sampling frequency to 384kHz (8 times oversampling), for the 150Hz sinewave we obtain that the time-domain RMS error is cut down to 1.1% and the frequency-domain RMS error is 0.24%. The increase in the sampling rate reduced the effect of the unit delay used in the approximation. The frequency response of this oversampled signal is displayed in Fig. 8. At every time step the value of the inductance $L[n]$ changes according to an estimated value of the current through the inductor, as explained in Section 4.1. Tables 1 to 3 show the frequency domain error figures for each of the test signals. Table 1 shows that the increase in the amplitude of the input signal causes larger errors. This is expected as to a larger amplitude corresponds a faster saturation

of the core and therefore a larger distortion of the output, with a steeper voltage drop in the time domain. Table 2 shows that results are greatly improved by increasing the sampling frequency. The error reduction is proportional to $1/F_s$.

The α parameter, as given by Eq. (34), determines the balance between the weight given to the value $I[n-1]$ of the current measured at the previous time step and the estimated value $I'[n]$ of the current through the inductor at the current time instant, when computing the value $L[n]$ of the inductance at the current time instant. When $\alpha = 0.5$ the estimated value used for the current through the inductor is the average value of the current over the time interval between $n-1$ and n and this is, in theory, the best choice to be used in the computation of $L[n]$, as long as the estimated value $I'[n]$ is reasonably close to the actual value $I[n]$.

Test results in Table 3 show that most of the times $\alpha = 0.75$ performs better than $\alpha = 0.5$ and also that both these values perform poorly when the signal contains significant amounts of energy at higher frequencies. This can be explained by the fact that as the frequency of the signal increases, the linear extrapolation becomes less accurate and $\alpha = 0.75$ performs better than $\alpha = 0.5$ because the overshoot caused by the estimate is mitigated by giving less weight to it. The value of $\alpha = 1$, corresponding to no linear extrapolation being used, was found to be the one that gives best results for a signal of arbitrary frequency content. This effectively corresponds to the introduction of a 1-sample delay so that the filter coefficients are entirely based on the system state at the previous sample instant.

6. DISCUSSION

In this paper we introduced a way to solve a high pass filter circuit containing a non-linear inductor using a time-varying IIR filter, whose block diagram is shown in Fig. 9. The model is physically informed and exploits the fact that the actual inductance value for a solenoid changes according to the saturation of its core. As pointed out in 4.1, these changes affect the frequency response of the filter by modulating its cutoff frequency.

6.1. Performance

The time-variant IIR filters used here to emulate the behavior of a non-linear inductor produced results comparable to the reference model and they did not exhibit any inherent instability. The time domain error due to the intrinsic delay in the approximation does not affect negatively the perceived sound, as it produces rather small error figures in the frequency-domain. For certain combinations of parametes (e.g. large μ_i, large input voltage) ringing and overshoot effects have been observed, due to the sudden change in the filter coefficients. These effects can be attenuated by hard limiting the slew rate of the filter coefficients, imposing a maximum-change-per-sample limit, or otherwise suppressing the transient using one of the techniques proposed in [8].

The use of linear extrapolation to compute an estimate of the inductance value at the current time sample did not improve the results. On the other hand, the model produces good results when not using linear extrapolation, with frequency-domain errors below 1.75% in all the cases under exam and below 1.2% when tested with real-world audio signals.

Inductors mostly saturate at low frequencies, therefore the use of oversampling is not a requirement when modeling this type of non-linearity, as the higher partials generated by the distortion are

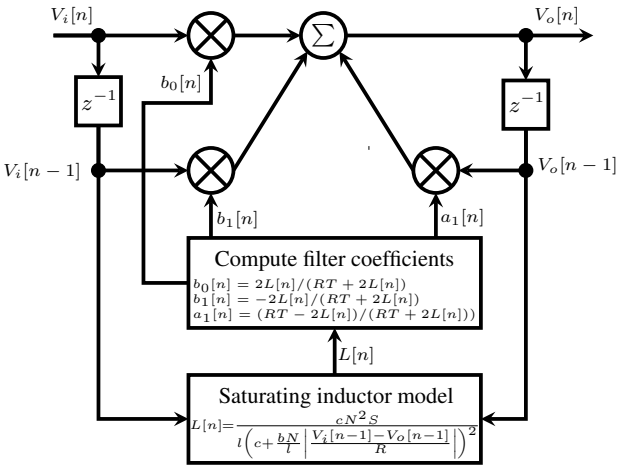


Figure 9: Block diagram of the digital filter for the circuit in Fig. 2, when $\alpha = 1$.

likely to be below the Nyquist frequency even for a sampling rate of 48kHz. Despite this general consideration, the choice of the oversampling factor has to be evaluated on a case-by-case basis, according to the characteristics of the inductor (e.g. saturation flux density), of the circuit (e.g. presence of other non linear elements and filters) and of the expected frequency content of the input signal. Nevertheless, the accuracy of the model takes advantage of oversampling, which reduces the effects of the 1-sample delay and gives better results overall.

As outlined above, Eq. (17) is a particular case which can be solved with lower computational cost than most non-linear equations found in DSP systems. This considered, solving the high pass circuit using the non-linear model and Eq. (17) requires 13 multiplies, 12 additions and 2 square roots per sample. Solving the same circuit with the time-variant IIR filter in Section 4 requires 5 multiplies, 4 additions and 2 divisions per sample. As on modern CPUs the execution time of square roots is greater or equal than the one for divisions, the time-variant IIR model turns out to have a lower computational cost than the non-linear one. Improvements in speed can become even greater when a similar approach is used to model non-linearities which are otherwise solved through computationally-expensive transcendental functions.

The stability and performance of time-variant IIR models have to be evaluated on a case-by-case basis. For instance, modeling of a diode clipper circuit as a time-variant resistor has been attempted by the authors which led to a conditionally-working model that requires oversampling and other adjustments to prevent DC drift of the output.

6.2. Applications

The idea that is at the base of this research, that is the use of time-variant linear filters with recursive coefficient computation to implement non-linearities, proved to be not only achievable, but also well suited for the emulation of a real electronic component, the inductor. This model has been successfully used to extend existing systems, without requiring major re-designs. For instance, it was used to add the non-linearities of the inductor to the wah-wah

pedal model based on the DK-method presented in [26] by simply replacing the static inductance in the circuit with a time-varying one. What emerged from the simulation of the wah-wah pedal is that the current through the inductor was too small to cause audible saturation, when using for the inductor parameters similar to the ones of a real wah-wah inductor. By introducing fictitious physical parameters for the inductor, we allowed the input signal to drive it into saturation. As a result we obtained increased harmonic distortion and a shift of the cutoff frequency of the filter.

The inductor model presented is not complete yet, as a full model of the inductor would require at least to add the hysteresis of the magnetic core. From what we have seen so far, it is reasonable to think that this additional step will not add much to the complexity of the model.

7. REFERENCES

[1] J. Macak, J. Schimmel, and V. Välimäki, "Real-time guitar preamp simulation using modified blockwise method and approximations," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, pp. 20, 2011.

[2] D. T. M. Yeh, "Automated physical modeling of nonlinear audio circuits for real-time audio effects;part II: BJT and vacuum tube examples," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 4, pp. 1207–1216, May 2012.

[3] K. Meerkötter and R. Scholz, "Digital simulation of nonlinear circuits by wave digital filter principles," in *Circuits and Systems, 1989., IEEE International Symposium on*. IEEE, 1989, pp. 720–723.

[4] S. Petrausch and R. Rabenstein, "Wave digital filters with multiple nonlinearities," in *Signal Processing Conference, 2004 12th European*, Sept 2004, pp. 77–80.

[5] J. O. Smith, *Introduction to Digital Filters with Audio Applications*, W3K Publishing, http://www.w3k.org/books/, 2007.

[6] G. Stoyanov and M. Kawamata, "Variable digital filters," *J. Signal Processing*, vol. 1, no. 4, pp. 275–289, 1997.

[7] J. Laroche, "On the stability of time-varying recursive filters," *Journal of the Audio Engineering Society*, vol. 55, no. 6, pp. 460–471, 2007.

[8] V. Välimäki and T.I. Laakso, "Suppression of transients in time-varying recursive filters for audio signals," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, 1998*, May 1998, vol. 6, pp. 3569–3572 vol.6.

[9] A. Wishnick, "Time-varying filters for musical applications," in *Proceedings of the 17th International Conference on Digital Audio Effects (DAFx-14)*, 2014, pp. 69–76.

[10] D Rossum, "Making digital filters sound "analog"," pp. 30–33, 1992.

[11] V. Välimäki, S. Bilbao, J. O. Smith, J. S. Abel, J. Pakarinen, and D. Berners, "Virtual analog effects," *DAFX: Digital Audio Effects, Second Edition*, pp. 473–475, 2011.

[12] J. Kleimola, V. Lazzarini, V. Valimaki, and J. Timoney, "Feedback amplitude modulation synthesis," *EURASIP Journal on Advances in Signal Processing*, , no. 434378, 2011.

[13] J. Pakarinen, V. Välimäki, and M. Karjalainen, "Physics-based methods for modeling nonlinear vibrating strings," *Acta Acustica united with Acustica*, vol. 91, no. 2, pp. 312–325, 2005.

[14] D. C. Jiles, J. B. Thoelke, and M. K. Devine, "Numerical determination of hysteresis parameters for the modeling of magnetic properties using the theory of ferromagnetic hysteresis," *Magnetics, IEEE Transactions on*, vol. 28, no. 1, pp. 27–35, 1992.

[15] J. Macak, "Nonlinear audio transformer simulation using approximation of differential equations," *Elektrorevue*, vol. 2, no. 4, December 2011.

[16] R. C. D. de Paiva, J. Pakarinen, V. Välimäki, and M. Tikander, "Real-time audio transformer emulation for virtual tube amplifiers," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, no. 1, pp. 347645, 2011.

[17] T. Stilson and J. Smith, "Analyzing the moog vcf with considerations for digital implementation," in *Proceedings of the 1996 International Computer Music Conference, Hong Kong, Computer Music Association*, 1996.

[18] A. Huovilainen, "Nonlinear digital implementation of the moog ladder filter," in *Proc. Int. Conf. on Digital Audio Effects (Naples, Italy, October 2004)*, 2004, pp. 61–4.

[19] S. D'Angelo and V. Välimäki, "Generalized moog ladder filter: Part ii–explicit nonlinear model through a novel delay-free loop implementation method," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 22, no. 12, pp. 1873–1883, 2014.

[20] A. Härmä, "Implementation of recursive filters having delay free loops," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. IEEE, 1998, vol. 3, pp. 1261–1264.

[21] D. J. Griffiths, *Introduction to electrodynamics*, vol. 3, Prentice hall Upper Saddle River, NJ, 1999.

[22] D. C. Jiles, *Introduction to Magnetism and Magnetic Materials, Second Edition*, Taylor & Francis, 1998.

[23] D. T. M. Yeh, *Digital implementation of musical distortion circuits by analysis and simulation*, Ph.D. thesis, Stanford University, 2009.

[24] "Incremental magnetic permeability," in *Computer Science and Communications Dictionary*, pp. 763–763. Springer US, 2001.

[25] G. Borin, G. De Poli, and D. Rocchesso, "Elimination of delay-free loops in discrete-time models of nonlinear acoustic systems," *Speech and Audio Processing, IEEE Transactions on*, vol. 8, no. 5, pp. 597–605, 2000.

[26] M. Holters and U. Zölzer, "Physical modelling of a wah-wah effect pedal as a case study for application of the nodal DK method to circuits with variable parts," in *Proceedings of the 14th International Conference on Digital Audio Effects DAFx11*, 2011, pp. 19–23.

DIGITIZING THE IBANEZ WEEPING DEMON WAH PEDAL

Chet Gnegy, Kurt James Werner

Center for Computer Research in Music and Acoustics (CCRMA), Stanford University
660 Lomita Drive, Stanford, CA 94305, USA
[chet|kwerner]@ccrma.stanford.edu

ABSTRACT

Being able to transform an analog audio circuit into a digital model is a big deal for musicians, producers, and circuit benders alike. In this paper, we address some of the issues that arise when attempting to make such a digital model. Using the canonical state variable filter as the main point of interest in our schematic, we will walk through the process of making a signal flow graph, obtaining a transfer function, and making a usable digital filter. Additionally, we will address an issue that is common throughout virtual analog literature; reducing the very large expressions for each of the filter coefficients. Using a novel factoring algorithm, we show that these expressions can be reduced from thousands of operations down to tens of operations.

1. INTRODUCTION

The *Weeping Demon* may be one of the most versatile wah pedals on the market. Much like the *Dunlop Crybaby*, the *Weeping Demon* offers control over the center frequency and Q of the filter. Due to the design of the *Weeping Demon*'s filter circuit, there is independent control over both of these features, as well as a controllable low range boost and a mode switch that changes the frequency range of the wah making it more suitable for the bass guitar. In this work, we model the filter circuit by obtaining its transfer function parameterized by its electrical components. The transfer function is then digitized via the bilinear transform [1][1].

The transfer function is obtained by reverse-engineering the circuit, which happens to be very similar to the canonical state variable filter (SVF) [2] consisting of four op-amp circuits in feedback with each other. The low impedance output of each op-amp allows us to treat each op-amp circuit independently and form a block diagram consisting of adds and multiplies. The transfer function at any node of the circuit can be obtained via Mason's rule. SVFs, which will be discussed in more detail later, have the interesting property that the high-, band-, and low-passed outputs are produced at each of three op-amps. In the *Weeping Demon*, a fourth op-amp combines the band- and low-pass outputs to produce a resonant low-pass filter, as is common for wah pedals.

Unfortunately, the transfer function that we obtain from Mason's rule has extremely complicated coefficients. In the case of the *Weeping Demon*, a product-of-sum coefficient can have as many as one hundred addends, each consisting of about five multiplications. We present a method of polynomial simplification capable of reducing the number of add/multiply operations for a single coefficient from several hundred down to 50 or fewer. Using common subexpression extraction, a typical compiler trick, we can reduce this even further.

[1] https://ccrma.stanford.edu/~jos/pasp/Bilinear_Transformation.html

Wah pedals have previously been studied in virtual analog. Models based on fitting biquads to measured filter responses can capture a wah pedal's basic global behavior[2], but they lack the detail of virtual analog physical models. Holters and Zölzer studied the *Dunlop Crybaby* in a nodal DK framework, handling efficient parameter update in the context of changing coefficients by exploiting a Woodbury identity—their technique is applicable to state-space systems where the number of variable parts is low compared to the total number of parts [3]. This is closely related to a technique used by Dempwolf *et al.*, who handle complicated coefficient updates by exploiting a certain minimized matrix formulation [4]. Falaize-Skrzek and Hélie studied the Crybaby pedal from a port-Hamiltonian perspective [5].

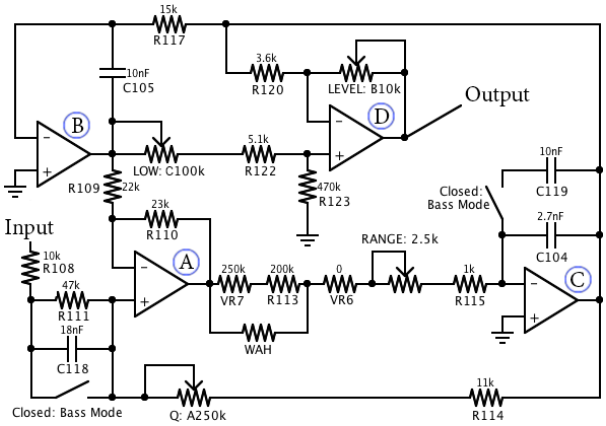


Figure 1: *The filter stage for the* Weeping Demon *with all components labeled.*

2. MODELING THE CIRCUIT

The filter stage of the *Weeping Demon* is shown in Figure 1. The switching circuit, which offers a bypass mode that turns on once the pedal has not been used for some programmed amount of time, will not be covered in this paper, mainly because there is no need for this in the digital model. The implementation of a timer that turns the effect off after a given time is trivial in software. Perhaps the most interesting bit about this circuit is the choice of sensors used to detect the angle of the foot pedal. Rather than using the common choice, a potentiometer, an optical sensor is used. An

[2] https://ccrma.stanford.edu/realsimple/faust_strings/Adding_Wah_Pedal.html

opaque fin is mounted to the bottom of the moving foot piece directly between an LED and a photoresistor. As the pedal is rocked, the fin allows more light to pass from the LED to the photoresistor, decreasing its resistance. The advantage of this is that unlike a potentiometer, the optical element will not accumulate dirt and become noisy with time and use. The modeling of the optical element will be discussed in section 3.

2.1. Overview

Prior to entering the shown circuit, the signal is passed through a buffer stage. It consists of two cascaded emitter followers, which provide a low impedance signal to the filter. The emitter follower stages act as a high-pass filter with cutoff of about 4 Hz. Because this is way below the audio range, will exclude it from the model.

The coupled network of op-amps shown is known as a state variable filter (SVF). It is rather complicated as a whole, but broken into components, it is much more easily understood. This decomposition will be used when deriving the transfer function. The first important observation is that op-amps B and C (labeled in the figure) are configured as integrators. The other two op-amps, A and D, are differential amplifiers. This is nearly enough information to dive into finding the transfer function, but it is insufficient for understanding how the circuit works.

We will start our analysis by noting that the important mechanics of the circuit can be realized by removing op-amp D from the circuit completely. This is because there is no feedback from D to any of the other stages. We can therefore ignore op-amp D, and remove the paths containing R_{120} and the potentiometer labeled LOW. This leaves us with a differential amplifier with non-inverting inputs from the filter input and from the output of C. Let us assume for now that the output of C is not feeding back to A. Equivalently, set the resistance of the potentiometer labeled Q to be arbitrarily high. We will relax this assumption later. This leaves with two cascaded integrators whose output is providing negative feedback to A. In essence, we are subtracting the original signal from itself.

The output of A and the output of B have a constant phase relationship: they are always 180 degrees out of phase. This should be easy to see, because the output of B appears at the inverting terminal of A. The integrators, B and C, are simply low-pass filters, each with a cutoff associated with some RC time constant. Without the feedback from B to A (basically ignoring the effects of A entirely), we can expect the output of B to have a low-pass characteristic. Now, considering the feedback, we note that we are adding an inverted, low-passed version of the input signal to itself. At low frequencies, we should expect a very minimal output of A. At high frequencies, the inverting terminal of A is very small in magnitude, so we see a high-pass behavior at the output of A. An important step in getting comfortable with this circuit may be to realize that by twice integrating a second order high-pass filter (A), we obtain a low-pass filter (C). As we reintroduce the feedback from the band-pass filter (B), we bring with it lower, more reasonable Q values [2].

2.2. Signal Flow Diagram

Using some basic topological information and superposition, we will represent the circuit as a block diagram with only adds and multiplies. First, lets simplify the component values, combining series and parallel elements into impedance terms, Z_i. We can see the reduced circuit in Figure 2. This will simplify the arithmetic quite a bit. The substitutions are shown in Table 1. It should be clear that we will derive a separate set of equations based on whether the mode switch is open or closed, designating Bass or Normal mode.

To figure out the global filtering properties of the SVF, it is useful to consider the op-amp stages individually. For each op-amp stage, we can derive how each of the other stages and the input voltage V_{in} contribute. These contributions, based on combined impedance values from Table 1 and the op-amp configurations, are summed together by superposition. This derivation, which is shown in detail for each stage in the next section, yields a global signal flow graph, as shown in Figure 3. We will show the derivation for the gains in the graph (the K_i's in Figure 3) in the next section.

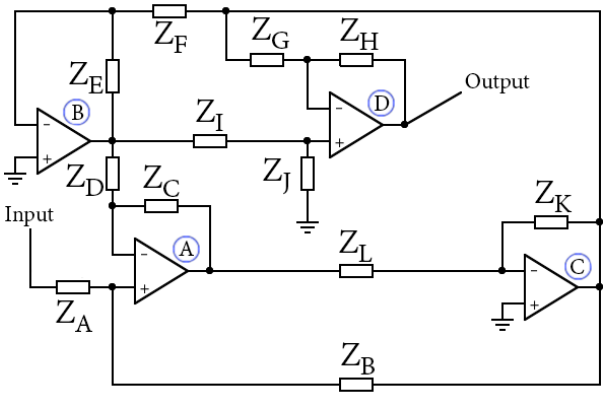


Figure 2: *The filter stage for the* Weeping Demon *with the components combined into impedance terms.*

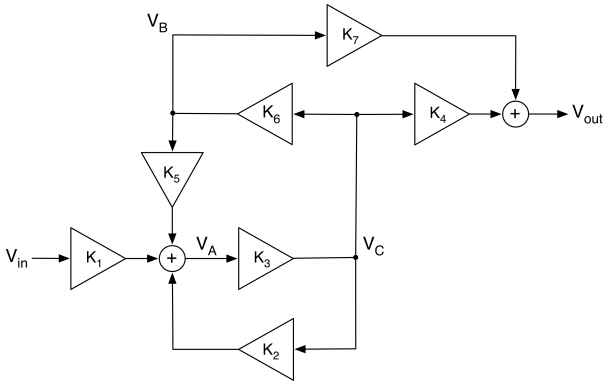


Figure 3: *A signal flow graph view of the SVF.*

2.3. Operational Amplifier Stages

Let's look at the four operational amplifier stages of the *Weeping Demon* Filter individually. These are the input summer, the first integrator, the second integrator, and the output summer.

Z_i	**Combined Components**	
	Bass	**Normal**
Z_A	R_{108}	$R_{108} + R_{111} \| \| \frac{1}{sC_{118}}$
Z_B	$R_Q + R_{114}$	
Z_C	R_{110}	
Z_D	R_{109}	
Z_E	$\frac{1}{sC_{105}}$	
Z_F	R_{117}	
Z_G	R_{120}	
Z_H	R_{LEVEL}	
Z_I	$R_{\text{LO}} + R_{112}$	
Z_J	R_{123}	
Z_K	$\frac{1}{s(C_{104}+C_{119})}$	$\frac{1}{s(C_{104})}$
Z_L	$(R_{\text{VR7}} + R_{113}) \| \| R_{\text{WAH}} + R_{\text{VR6}} + R_{115} + R_{\text{RANGE}}$	

Table 1: *Substitutions for Z_i. Due to the switch in the circuit, we have different components for Z_A and Z_K depending on whether Bass or Normal mode is currently being used.*

Each stage can be analyzed by assuming ideal op-amps, invoking superposition, and using the equations for inverting amplifier and difference amplifiers. Ideal op-amps have zero output impedance—their outputs can be treated as ideal voltage sources. Furthermore, they are linear, so the output of each op-amp can be found by summing its response to each voltage source input. When finding the response to one voltage source input, the others are shorted. Under this condition, each op-amp acts as either a differential amplifier (with only non-inverting input) or an inverting amplifier to each input.

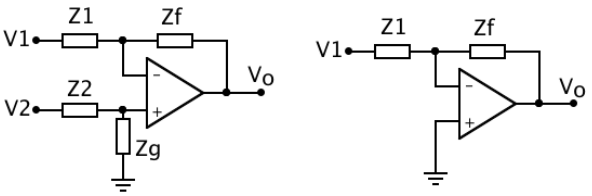


Figure 4: *Differential (left) and inverting (right) amplifiers with generalized impedances.*

Differential and inverting amplifiers with generalized impedances are shown in Figure 4. The output of each amplifier depends on the ratios between connected impedances. The output of a differential amplifier is given by:

$$V_o = -\frac{Z_f}{Z_1} V_1 + \left(\frac{Z_1 + Z_f}{Z_1}\right)\left(\frac{Z_g}{Z_g + Z_2}\right) V_2 \qquad (1)$$

and the output of an inverting amplifier is given by:

$$V_o = -\frac{Z_f}{Z_1} V_1. \qquad (2)$$

In all of our cases, when considering superposition, the inverting input V_1 of the differential amplifier will be grounded, so a simpler version of Eqn. (1) applies:

$$V_o = \left(\frac{Z_1 + Z_f}{Z_1}\right)\left(\frac{Z_g}{Z_g + Z_2}\right) V_2. \qquad (3)$$

Table 2 shows the gains for the transfer function, op-amp, input voltage, configuration (**i**nverting or **d**ifferential), and impedances for each case of superposition.

op	TF	V_{in}	i/d	Z_1	Z_2	Z_f	Z_g
A	K_1	V_{in}	d	Z_D	Z_A	Z_C	Z_B
A	K_5	V_B	i	Z_D		Z_C	
A	K_2	V_C	d	Z_D	Z_B	Z_C	Z_A
B	K_6	V_C	i	Z_F		Z_E	
C	K_3	V_A	i	Z_L		Z_K	
D	K_7	V_B	d	Z_G	Z_I	Z_H	Z_J
D	K_4	V_C	i	Z_G		Z_H	

Table 2: *For each case of superposition, configuration and impedances seen by the op-amps.*

2.3.1. Input Summer

The input summer is the circuitry around op-amp A that combines V_{in}, the input to the SVF, with feedback from the first and second integrators (V_C and V_B). By superposition, we can get its output in terms of the partial transfer functions K_1, K_5, and K_2,

$$V_A = K_1 V_{\text{in}} + K_5 V_B + K_2 V_C, \qquad (4)$$

where K_1, K_5, and K_2 are found in terms of the combined impedances using Eqns. (2)–(3) with values from Table 2 as appropriate:

$$K_1 = \left(\frac{Z_C + Z_D}{Z_C}\right)\left(\frac{Z_B}{Z_B + Z_A}\right) \qquad (5)$$

$$K_5 = -\frac{Z_C}{Z_D} \qquad (6)$$

$$K_2 = \left(\frac{Z_C + Z_D}{Z_D}\right)\left(\frac{Z_A}{Z_A + Z_B}\right). \qquad (7)$$

In a standard SVF, this input stage would be purely resistive—the magnitude response would be flat and the resistances chosen to get the desired circuit response. In the *Weeping Demon*, there is a reactive component when the pedal is set in "normal mode." Since this is part of Z_A, it has implications for K_1 and K_2—how the summer affects the input signal V_{in} and also the feedback from the first integrator V_C. We can speculate about why this capacitor is put in place for "normal mode." Perhaps for a guitar input, the circuit designers wanted to damp down low frequencies for noise reasons; perhaps it is just an ad hoc voicing choice.

The output of the input summer (A) is the high-pass output of the SVF.

2.3.2. First Integrator

The first integrator is the circuitry around op-amp C that integrates the output V_A of the input summer. Its output V_C is applied to the second integrator, through a feedback path to the non-inverting input of the input summer, and to the inverting input of the output

summer. In terms of the circuit generalized impedances, its output is

$$V_C = K_6 V_A, \tag{8}$$

where

$$K_6 = -\frac{Z_K}{Z_L}. \tag{9}$$

Impedance Z_K changes depending on whether the pedal is set in "normal mode" or "bass mode." An extra capacitor is placed in parallel for "bass mode," shifting the response of the integrator. The integrator response is also shifted when the wah pedal's position changes, changing R_{WAH} and hence R_L; this is mechanism by which R_{WAH} shifts the output resonant filter's center frequency.

The output of the first integrator (V_C) is the band-pass output of the SVF.

2.3.3. Second Integrator

The second integrator is the circuitry around op-amp B that integrates the output V_C of the first integrator. Its output V_B is applied through a feedback path to the inverting input of the input summer and to the output summer. In terms of the circuit generalized impedances, its output is

$$V_B = K_3 V_C, \tag{10}$$

where

$$K_3 = -\frac{Z_E}{Z_F}. \tag{11}$$

The output of the second integrator (B) is the low-pass output of the SVF.

2.3.4. Output Summer

The output summer sums contributions from the band-pass (V_C) and low-pass (V_B) outputs of the SVF. By superposition, its output is:

$$V_{\text{out}} = K_7 V_B + K_3 V_C \tag{12}$$

where

$$K_7 = \left(\frac{Z_G + Z_H}{Z_G}\right)\left(\frac{Z_J}{Z_I + Z_J}\right) \tag{13}$$

$$K_3 = -\frac{Z_H}{Z_G}. \tag{14}$$

This combination of the band-pass (C) and low-pass(B) SVF outputs forms a resonant low-pass filter, a typical goal of wah pedal design. We note that it would be possible to modify the output summer circuitry (or the digital model) so that the output summer combined different SVF outputs, yielding other filter configurations like resonant high-pass, etc.

3. MODELING THE OPTICAL ELEMENT

The optical element that controls the center frequency of the wah resonance is the most difficult element in the circuit to model. Whereas resistors and capacitors in the circuit are well modeled in the audio band by their ideal generalized impedances and operational amplifiers can be considered ideal, the effect of the optical element is more complex.

The current–voltage (i–v) characteristics of the stationary LED and photoresistor pair depends in complex ways on the internal geometry of the pedal as well as the electrical characteristics of the LED and photoresistor. The optical/geometric properties of the enclosure are too complex to predict from first principles. Lacking datasheets or documentation for the LED and photoresistor, and even access to a good model of this photoresistor's behavior, the behavior of this pair of components is difficult to predict. So, we make recourse to black-box modeling and fit a model to measured data.

When the foot pedal is rocked, the black, metal fin that divides the LED and the photoresistor moves, and is no longer an obstruction. We cannot expect to get a useful measurement of the mapping from pedal angle to resistance with the pedal disassembled because of the ambient light for the room. We instead cut the copper traces around the photoresistor and solder wires to each of its terminals. The pedal is reassembled with the wires running out of the pedal through a small hole near the battery holder. Making no assumptions about the linearity of this element with respect to any of its parameters, we set out to measure its i–v transfer characteristic as a function of pedal angle, θ. The photoresistor is connected in series with a 33kΩ resistor via the long wires as seen in Figure 5. We sweep the voltage V_{DC} across the series connection, recording the voltage across the photoresistor ($V_{DC} - v_R$, where v_R is voltage across the 33kΩ resistance) as well as the current through the circuit, $v_R/33k\Omega$. θ is increased in increments of 2°, as measured from the rotational axis. For each θ, we sweep V_{DC} from 0V up to around 5V. Figure 6 shows the result.

For each angle θ, we observe an approximately straight line in the i–v characteristic, indicating that the photoresistor can be accurately modeled as linear for any given θ value. The slope of this line gives us the resistance of the component. The relationship between θ and the resistance of the photoresistor is shown in Figure 7. We can now model the resistance $R_{\text{WAH}}(\theta)$ as the pedal is rocked by fitting a curve to these points.

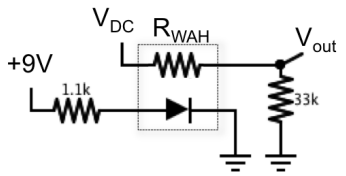


Figure 5: *Setup to find $R_{WAH}(\theta)$.*

4. THE TRANSFER FUNCTION

Now that we have a signal flow chart, we can get our transfer function. It is of interest to express the transfer function as a ratio of two polynomials in s, as seen in Equation 15.

$$H(s) = \frac{b_m s^m + \cdots + b_2 s^2 + b_1 s + b_0}{a_n s^n + \cdots + a_2 s^2 + a_1 s + a_0} \tag{15}$$

Both Matlab and Python symbolic libraries are used to obtain and reduce the transfer function. We do this via Mason's gain law [6, 7], the same treatment seen in modeling the TR-808 cowbell [8] and as Kramer demonstrates generically [9]. In short, we leverage a Matlab function written by Rob Walton[3] that converts

[3] `http://www.mathworks.com/matlabcentral/fileexchange/22-mason-m`

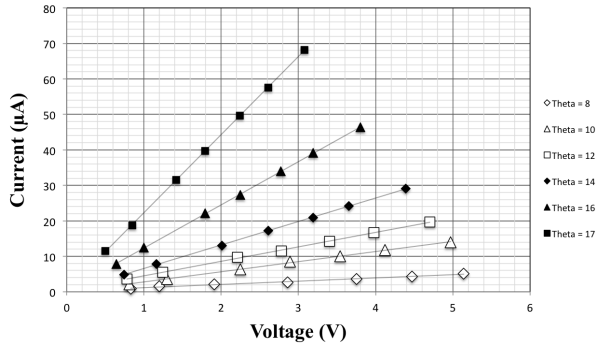


Figure 6: *Current–voltage characteristic for different pedal positions. $\theta = 6^\circ$ and $\theta = 8^\circ$ produced the same curve indicating that the internal geometry didn't change much. The maximum angle without putting a lot of pressure on the foot pedal was $\theta = 17^\circ$.*

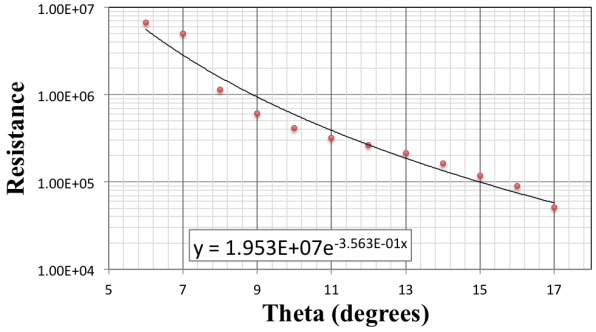


Figure 7: *The curve produced for $R_{WAH}(\theta)$. A curve fitted equation is shown. Different curve fits would change the behavior of the model significantly. The presented equation was used for simplicity and because it matched the real pedal decently well. After some experimentation, a higher order fit was not deemed necessary.*

a netlist for a network of summations and gains to a single transfer function. This can be done for any chosen output of the filter, but we perform our analysis for V_{out}. The transfer function seen in Equation 16 expresses the transfer characteristic from input to output in terms of the gain coefficients, K_i. The impedance values, Z_i in Table 2, can now be substituted back for the $K's$, and in turn, the symbols designating the component values.

$$H(s) = \frac{V_{\text{out}}}{V_{\text{in}}} = \frac{-K_1K_3(K_4 + K_6K_7)}{K_2K_3 + K_3K_5K_6 - 1} \tag{16}$$

We do this using the `simplifyFraction` function in the Matlab symbolic library. The `simplifyFraction` function reduces a fraction such that the greatest common divisor of the numerator and denominator is 1. Without this step, the equation is quite messy and does not necessarily contain only positive powers of s. Once we have done that, we are left with an analog filter with coefficients that can be expressed in the form shown in Equation 15. In this form, it is clear that the normal mode filter is third order, and the bass mode filter is only second order.

The result is something familiar from other works in virtual analog: the coefficients are very complicated. Two examples of this are [8] and [10], in which Werner *et al.* demonstrate a discretization of a band-pass filter from the *TR-808* cowbell and Yeh and Smith discretizes the '59 *Fender Bassman Tone Stack*. Each of these works models a circuit with a relatively low component count and results in coefficients that require, in their presented form, 82 and 280 operations. The *Weeping Demon* in normal mode features a whopping 3917 operations in expanded form. Even the partially factored result given by Matlab contains nearly 400 operations. The worst of these coefficients, a_2, in the form given by Matlab, is shown:

$$\begin{aligned}
a_2 = {} & R_{120}(R_{\text{LO}} + R_{122} + R_{123})C_{105}R_{117}(\\
& C_{104}R_{\text{Q}}R_{109}R_{113}R_{115} + C_{104}R_{\text{Q}}R_{109}R_{113}R_{\text{RANGE}} + \\
& C_{104}R_{108}R_{109}R_{113}R_{115} + C_{104}R_{109}R_{111}R_{113}R_{115} + \\
& C_{104}R_{109}R_{113}R_{114}R_{115} + C_{104}R_{109}R_{113}R_{114}R_{\text{RANGE}} + \\
& C_{118}R_{108}R_{110}R_{111}R_{113} + C_{104}R_{108}R_{109}R_{113}R_{\text{RANGE}} + \\
& C_{104}R_{109}R_{111}R_{113}R_{\text{RANGE}} + C_{118}R_{108}R_{109}R_{111}R_{113} + \\
& C_{104}R_{\text{Q}}R_{109}R_{113}R_{\text{VR6}} + C_{104}R_{109}R_{111}R_{\text{RANGE}}R_{\text{VR7}} + \\
& C_{104}R_{\text{Q}}R_{109}R_{\text{RANGE}}R_{\text{VR7}} + C_{104}R_{\text{Q}}R_{109}R_{113}R_{\text{WAH}} + \\
& C_{104}R_{\text{Q}}R_{109}R_{115}R_{\text{WAH}} + C_{104}R_{108}R_{109}R_{113}R_{\text{VR6}} + \\
& C_{104}R_{108}R_{109}R_{115}R_{\text{VR7}} + C_{104}R_{109}R_{111}R_{113}R_{\text{VR6}} + \\
& C_{104}R_{109}R_{111}R_{115}R_{\text{VR7}} + C_{104}R_{109}R_{113}R_{114}R_{\text{VR6}} + \\
& C_{104}R_{109}R_{114}R_{115}R_{\text{VR7}} + C_{118}R_{108}R_{109}R_{111}R_{\text{VR7}} + \\
& C_{118}R_{108}R_{110}R_{111}R_{\text{VR7}} + C_{104}R_{\text{Q}}R_{109}R_{\text{RANGE}}R_{\text{WAH}} + \\
& C_{104}R_{108}R_{109}R_{\text{RANGE}}R_{\text{VR7}} + C_{104}R_{\text{Q}}R_{109}R_{115}R_{\text{VR7}} + \\
& C_{104}R_{109}R_{114}R_{\text{RANGE}}R_{\text{VR7}} + C_{104}R_{108}R_{109}R_{113}R_{\text{WAH}} + \\
& C_{104}R_{108}R_{109}R_{115}R_{\text{WAH}} + C_{104}R_{109}R_{111}R_{113}R_{\text{WAH}} + \\
& C_{104}R_{109}R_{111}R_{115}R_{\text{WAH}} + C_{104}R_{109}R_{113}R_{114}R_{\text{WAH}} + \\
& C_{104}R_{109}R_{114}R_{115}R_{\text{WAH}} + C_{118}R_{108}R_{109}R_{111}R_{\text{WAH}} + \\
& C_{118}R_{108}R_{110}R_{111}R_{\text{WAH}} + C_{104}R_{108}R_{109}R_{\text{RANGE}}R_{\text{WAH}} + \\
& C_{104}R_{109}R_{111}R_{\text{RANGE}}R_{\text{WAH}} + C_{104}R_{\text{Q}}R_{109}R_{\text{VR6}}R_{\text{WAH}} + \\
& C_{104}R_{\text{Q}}R_{109}R_{\text{VR6}}R_{\text{VR7}} + C_{104}R_{109}R_{114}R_{\text{RANGE}}R_{\text{WAH}} + \\
& C_{104}R_{\text{Q}}R_{109}R_{\text{VR7}}R_{\text{WAH}} + C_{104}R_{108}R_{109}R_{\text{VR6}}R_{\text{VR7}} + \\
& C_{104}R_{109}R_{111}R_{\text{VR6}}R_{\text{VR7}} + C_{104}R_{109}R_{114}R_{\text{VR6}}R_{\text{VR7}} + \\
& C_{104}R_{108}R_{109}R_{\text{VR6}}R_{\text{WAH}} + C_{104}R_{108}R_{109}R_{\text{VR7}}R_{\text{WAH}} + \\
& C_{104}R_{109}R_{111}R_{\text{VR6}}R_{\text{WAH}} + C_{104}R_{109}R_{111}R_{\text{VR7}}R_{\text{WAH}} + \\
& C_{104}R_{109}R_{114}R_{\text{VR6}}R_{\text{WAH}} + C_{104}R_{109}R_{114}R_{\text{VR7}}R_{\text{WAH}} \\
&)
\end{aligned}$$

5. COMPARISON TO SPICE AND REAL DATA

When we compare the (analog) frequency response to the SPICE model, they are a perfect match. The digital frequency response, which is obtained via a third order bilinear transform [11], is shown in Figure 8 alongside the SPICE model. As expected, we see a sharp roll-off associated with the frequency warping characteristic of the bilinear transform near the Nyquist limit. This gives us assurance that we are modeling the schematic effectively. Indeed, we see that when we compare using any set of parameters, we have a model that is consistent with SPICE.

Though a full discussion is outside the scope of this paper, the match between the model and measurements of the real pedal is not as good. In general, the match is only approximate, showing discrepancies in center frequency, Q, and overall gain of the transfer function. This error can potentially be ascribed to limitations in our model of the optical element, which shows sensitivity to curve-fitting of the $R_{\text{WAH}}(\theta)$ relationship. 5% changes in curve

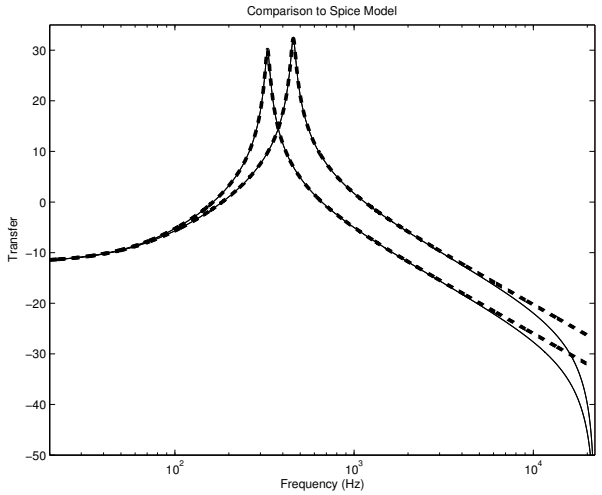


Figure 8: *Comparison to the SPICE Model. The curve for our model is the solid line, and the SPICE model is shown as a dashed line. The curves are shown for a pair of different parameter values. The curves are no longer overlapping in the high frequencies because of the frequency warping due to the bilinear transform.*

parameters resulted in changes of R_{WAH} on the order of tens of megaohms. Board-tracing errors are also possible, but unlikely since both authors independently traced the same schematic, and the schematic shows such a close match to a standard and sensible design (the SVF). Component, manufacturing, and trimpot tuning tolerances seem to be a likely source of error. Comparisons between 3 different *Weeping Demon* pedals showed that each pedal had a very different voicing, and that 2 of the 3 even exhibited instability when the Q knob was turned past about 2 o'clock. This was discovered within minutes of taking the pedals out of the box and is strongly in support of either a wide degree of variation between pedals or just a poor filter design.

Though we lack a strong case that the digital model matches the real-world circuit with a high degree of accuracy, the digitization scheme remains valid for the traced schematic, as shown by the correspondence with SPICE. In fact, a main finding of this paper is not the digital model itself, but a more general technique for minimized computation effort of digital filter coefficients which will be presented in the next section.

6. COEFFICIENT REDUCTION

The reduction process is done in two steps and was implemented using the symbolic Python library, SymPy [12]. First, there is a factoring step, and second is the common subexpression extraction step. To further motivate the need for the factoring step, it is important to mention that neither Matlab nor Python's symbolic packages would provide adequate simplification to the some of the more difficult factoring problems. If there was, say, a resistance value that was included in every term of a long coefficient computation, however, it would successfully factor it out. As a result, the coefficients returned by Matlab are not completely expanded and do have minimal amounts of factoring. The coefficient above, a_2, is an example of this. Our algorithm provides a much more satisfactory result, though no claims of optimality will be made.

The factoring step is a recursive algorithm that is shown in pseudocode in Algorithm 1. Before getting into the details of the main algorithm, it is necessary to introduce some supplementary functions (most of which are built into SymPy).

```
// Expands any parenthetical groupings;
function expand(expression)

// Finds all symbols that are included in 'expression';
function getVariables(expression)

// Finds the order of 'expression' with respect to 'var';
function orderOf(expression, var)

// Collects the coefficients of 'expression' for all power;
// of the variable 'var';
function collectTerms(expression, var)

// Picks a variable from a list 'vars' given some heuristic;
function chooseVar(vars)

// The main factoring algorithm;
function factored(exp)
    terms = [];
    // get all variables contained in exp;
    vars = getVariables (exp);
    if length(vars) ≤ 1 then
        terms.append( (exp,1) );
    else
        pickVar = chooseVar (vars);
        N = orderOf (exp, pickVar);
        // pows: array of descending powers of pickVar;
        // coeffs: coefficients to terms in pows;
        (coeffs, pows) = collectTerms (exp, pickVar);
        for i=0 to N do
            coeffs[i] = factored (coeffs[i]);
            terms.append( (pows[i], coeffs[i]) );
        end
    end
    recombine = 0;
    for i=0 to length(terms) do
        recombine += terms[i][0]*terms[i][1]
    end
    return recombine;
```

Algorithm 1: *The factoring algorithm used to reduce the expressions to a more computationally efficient form.*

The first, `expand`, is used to remove all parenthetical expressions by expansion. For example, $f(a,b,c) = (a+b)(c+b)$ is expanded to get $ac + ab + bc + b^2$. If we would like to find the variables involved in $f(a,b,c)$, we can access them via the `getVariables` function. `getVariables` will, in this case, return the list $[a,b,c]$.

The `orderOf` function returns the highest power of an expression with respect to a single variable.

The `collectTerms` function is used to factor out a single variable. For example, `collectTerms`$(f(a,b,c),b)$ will return two lists; the first of which containing the coefficients for each power of b contained in $f(a,b,c)$, and the second containing those powers of b. For $f(a,b,c)$ and (b), the returned lists are $[1, a+c, ac]$ and $[b^2, b, 1]$. The inner product of these two lists will, by definition, give an expression that is mathematically equivalent to $f(a,b,c)$. However, without additional expansion,

they are not identically equivalent and the inner product of the result is guaranteed to have *at most* the same number of operations as $f(a, b, c)$.

Algorithm 1 shows the process by which we reduce the equations. The basic premise of the algorithm is that we pick a variable v_i from the expression using some heuristic and factor it out. This choice is accomplished using the `chooseVar` function, whose implementation will be discussed shortly. We then factor by performing `collectTerms` on the expression. For each coefficient of v_i's powers (including $v_i^0 = 1$), we recurse. A list of the terms that are being factored is kept. Once each coefficient to v_i's powers is factored, we will recombine them using an inner product as previously mentioned, noting that this form is guaranteed to have, in the worst case, just as many operations as it started with. Once the expression is a function of only a single variable, nothing more can be factored out and the recursion ceases.

For a given expression that is completely expanded, we are guaranteed to get a result that is at least as good as the original. However, we note that the choice of v_i will change the amount of possible simplification. Thus, a good heuristic for choosing v_i is needed to ensure that partially factored input will still be improved by this algorithm. Three heuristics were tested for the `chooseVar` implementation: choosing the first variable found, choosing the most common variable, and choosing the least common variable. The most common variables is taken to mean "most common from the original set of expressions" and not from the current subexpression (the symbol, exp, from Algorithm 1). In the event that the most common variable is not in the subset, $vars$, it chooses the most common variable from the expression that does appear in $vars$. The same method was used for the least common variable heuristic. Choosing the first variable is subject to whatever ordering that Matlab may put on the variables, but we will assume that this heuristic is making a fairly arbitrary choice. Of the three, choosing the most common variable performed the best on each of the tested sets of coefficients and will be used when reporting results, followed closely by the arbitrary choosing of the first variable.

Once the factoring algorithm has completed, the common subexpression extraction (CSE) step is performed. This step relies completely on the `cse` function built into SymPy. This function takes an expression or group of expressions and replaces any calculations that happen multiple times with a temporary variable whose value is computed only once. This creates several more expressions than we started with, but the total number of operations will be reduced with every substitution. The algorithm then stores a copy of the expressions and counts the operations. The results at each step are shown in Table 3. The calculations for the coefficients are shown in Equations 17 and 18. The initial state of the coefficients (partially factored or not) was not observed to change the final operation counts. By inspection, we see that the *Weeping Demon* could be simplified slightly further, saving a few operations. For instance, the b_2 coefficient contains the term $x_2x_3x_5$, which should have been replaced with x_6. This is clearly a fault of the CSE algorithm, but it does not change the fact that taking the approach of factoring and using CSE, even with off-the-shelf software, leads to dramatic reductions in the required computation.

It is interesting to note that we see much larger reductions from the fully expanded form of the *Weeping Demon* equations than for the tone stack and cowbell. It is speculated that the reason is due to the isolation between op-amps in the circuit topology. The tone stack was an impedance network with no "stages" to be treated separately. Similarly, the band-pass filter of the cowbell was only a single op-amp. There was less to condense and no isolated stages to prevent terms from combining in complicated ways.

Operations Count				
	Initial	Expanded	Factored	CSE
WD: Normal Mode	383	3917	140	66
WD: Bass Mode	75	2410	64	40
Tone Stack	280	312	160	86
808 Cowbell	82	82	57	32

Table 3: *The number of operations for the analog coefficients of several virtual analog models. Initial counts for the Weeping Demon use the forms given by Matlab and initial counts for the Fender Bassman tone stack and TR-808 cowbell are for the form of the equations presented in each authors' original work. Expanded operation counts are obtained by completely expanding each coefficient. The operation count after the factoring algorithm shows modest improvement and finally after the CSE step the operation count is much lower.*

$$
\begin{aligned}
x_0 &= R_{\text{LO}} + R_{122} + R_{123} \\
x_1 &= C_{105}R_{111}R_{117}x_0 \\
x_2 &= R_{\text{Q}} + R_{114} \\
x_3 &= R_{109} + R_{110} \\
x_4 &= R_{113} + R_{\text{VR7}} \\
x_5 &= R_{\text{WAH}} + x_4 \\
x_6 &= x_2x_3x_5 \\
x_7 &= C_{118}R_{111}R_{123} \\
x_8 &= C_{105}R_{117} \\
x_9 &= x_0x_8 \\
x_{10} &= R_{108} + x_2 \\
x_{11} &= R_{115} + R_{\text{RANGE}} + R_{\text{VR6}} \\
x_{12} &= R_{\text{WAH}}(x_{11} + x_4) + x_{11}x_4 \\
x_{13} &= R_{110}R_{111} \\
x_{14} &= C_{118}R_{108}x_5 \\
x_{15} &= C_{104}x_{12} \\
x_{16} &= R_{120}x_0x_5 \\
x_{17} &= R_{108} + R_{111} \\
\boldsymbol{b_2} &= C_{118}R_{\text{LEVEL}}x_1x_2x_3x_5 \\
\boldsymbol{b_1} &= x_6(R_{\text{LEVEL}}(x_7 + x_9) + R_{120}x_7) \\
\boldsymbol{b_0} &= R_{123}x_6(R_{\text{LEVEL}} + R_{120}) \\
\boldsymbol{a_3} &= C_{104}C_{118}R_{109}R_{120}x_1x_{10}x_{12} \\
\boldsymbol{a_2} &= R_{120}x_9(R_{109}(R_{111}(x_{14} + x_{15}) + x_{10}x_{15}) + x_{13}x_{14}) \\
\boldsymbol{a_1} &= x_{16}(C_{118}x_{10}x_{13} + x_{17}x_3x_8) \\
\boldsymbol{a_0} &= R_{110}x_{16}(x_{17} + x_2)
\end{aligned}
\tag{17}
$$

$$
\begin{aligned}
x_0 &= R_{\text{LO}} + R_{122} + R_{123} \\
x_1 &= C_{105}R_{117}x_0 \\
x_2 &= R_{\text{Q}} + R_{114} \\
x_3 &= R_{109} + R_{110} \\
x_4 &= R_{113} + R_{\text{VR7}} \\
x_5 &= R_{\text{WAH}} + x_4 \\
x_6 &= R_{108} + x_2 \\
x_7 &= R_{115} + R_{\text{RANGE}} + R_{\text{VR6}} \\
\boldsymbol{b_1} &= R_{\text{LEVEL}}x_1x_2x_3x_5 \\
\boldsymbol{b_0} &= R_{123}x_2x_3x_5(R_{\text{LEVEL}} + R_{120}) \\
\boldsymbol{a_2} &= R_{109}R_{120}x_1x_6(C_{104} + C_{119})(R_{\text{WAH}}(x_4 + x_7) + x_4x_7) \\
\boldsymbol{a_1} &= C_{105}R_{108}R_{117}R_{120}x_0x_3x_5 \\
\boldsymbol{a_0} &= R_{110}R_{120}x_0x_5x_6
\end{aligned}
\tag{18}
$$

7. CONCLUSIONS

We have presented a method for going from a schematic of a state variable filter to a numerical model that well replicates the behavior of the SPICE model. Though much of the analysis was specific to the state variable filter, these methods could easily be adapted to other linear circuits. Additionally, we tackled the common problem of having very large expressions for the filter coefficients by doing a factoring algorithm followed by common subexpression extraction. The results of this factoring were quite promising as they can reduce the operation count for the filter coefficients from thousands down to tens. It is now much more computationally efficient to repeatedly compute coefficients for component values that may change as a function of time.

8. SOURCE CODE

Full schematics, SPICE models, source code, and illustrated documentation can be found on the web:
`http://www.chetgnegy.com/projects/weepingdemon.html`.

9. ACKNOWLEDGMENTS

Thanks to Jonathan Abel for advice, for sharing his extensive experience in virtual analog, and for just being a fun guy to talk to.

10. REFERENCES

[1] Julius O. Smith, *Physical Audio Signal Processing for Virtual Musical Instruments and Audio Effects*, 2010, online book, 2010 edition.

[2] Lawrence P Huelsman, Ed., *Active filters: lumped, distributed, integrated, digital, and parametric*, vol. 11 of *Inter-University Electronics Series*, McGraw–Hill, 1970.

[3] Martin Holters and Udo Zölzer, "Physical modelling of a wah-wah effect pedal as a case study for application of the nodal DK method to circuits with variable parts," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-11)*, Paris, France, September 19–23 2011, pp. 31–35.

[4] Kristjan Dempwolf, Martin Holters, and Udo Zölzer, "Discretization of parametric analog circuits for real-time simulations," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, September 6–10 2010, vol. 13.

[5] Antoine Falaize-Skrzek and Thomas Hélie, "Simulation of an analog circuit of a wah pedal: a port-Hamiltonian approach," in *Proceedings of the International Audio Engineering Society Convention*, New York, NY, October 17–20 2013, vol. 135.

[6] Samuel J. Mason, "Feedback theory—further properties of signal flow graphs," in *Proceedings of the IRE*, 1956.

[7] Samuel J. Mason, "Topological analysis of linear nonreciprocal networks," *Proceedings of the IRE*, vol. 45, no. 6, pp. 829–838, 1957.

[8] Kurt James Werner, Jonathan S. Abel, and Julius O. Smith, "More cowbell: a physically-informed, circuit-bendable, digital model of the TR-808 cowbell," in *Proceedings of the International Audio Engineering Society Convention*, Los Angeles, CA, October 9–12 2014, vol. 137.

[9] Martin Krämer, "Design of a CMOS sample-and-hold amplifier for a high precision front-end circuit using and extended $^{g_m}/_{I_{ds}}$ method," M.S. thesis, Technische Universität, Kaiserlautern, 2009.

[10] David Te-Mao Yeh and Julius O. Smith, "Discretization of the '59 Fender Bassman tone stack," in *Proceedings of the International Conference on Digital Audio Effects (DAFx-06)*, Montréal, Canada, September 18–20 2006.

[11] Peter J. Pupalaikis, "Bilinear transformation made easy," ICSPAT, 2000.

[12] "Sympy," `http://www.sympy.org/en/index.html`, Accessed: 2015-02-6.

CASCADED PREDICTION IN ADPCM CODEC STRUCTURES

Marco Fink, Udo Zölzer

Department of Signal Processing and Communications,
Helmut-Schmidt University Hamburg
Hamburg, Germany
marco.fink@hsu-hh.de

ABSTRACT

The aim of this study is to demonstrate how ADPCM-based codec structures can be improved using cascaded prediction. The advantage of predictor cascades is to allow the adaption to several signal conditions, as it is done in block-based perceptual codecs like MP3, AAC, etc. In other words, additional predictors with a small order are supposed to enhance the prediction of non-stationary signals. The predictor cascade is complemented with a simple adaptive quantizer to yield a simple exemplary codec which is used to demonstrate the influence of the predictor cascade. Several cascade configurations are considered and optimized using a genetic algorithm. A measurement of the prediction gain and the ODG score utilizing the PEAQ algorithm applied to the SQAM dataset shall reveal the potential improvements.

1. INTRODUCTION

Many multimedia applications require high-quality streaming of audio content but only allow a restricted data rate for the transmission. This requirement led to the development of audio codecs which are successfully applied in manifold applications. However, there are some interactive musical applications, like wireless digital microphones or Networked Music Performances [1, 2], which additionally feature very strict latency requirements[3]. Popular audio codecs, like MP3, AAC, or HE-AAC, were designed to deliver lowest bit rates but feature algorithmic delays of up to hundreds of milliseconds. To decrease the delay contribution of audio codecs in interactive internet applications, the ultra-low delay codec [4, 5] and subsequently, OPUS [6, 7] were presented among others. Both codec approaches decrease the coding delay to a few milliseconds.

Codecs based on adaptive differential pulse code modulation (ADPCM), as analyzed in [8, 9, 10], cause a single sample delay which would be optimal for delay-sensitive applications but are ranked behind block-based methods in the quality-bitrate tradeoff. Robustness against transmission errors can be achieved by modifying the predictors as shown in [11]. To potentially improve the quality of ADPCM-like codecs, this study shall investigate how the application of cascaded predictors, as done for lossless coding in [12] and lossy coding in [4, 5], can yield increased prediction gains and therefore, higher perceptual quality. Note that the codec structures of [12, 4, 5] use cascaded prediction but are open loop implementations and the utilized ADPCM structure in this work is a closed loop approach.

Section 2 presents the structure of a typical ADPCM codec. The approach of a cascaded predictor is described in Sec. 3. The exemplary codec and its implementation is denoted in Sec. 4. The optimization of the cascade parameters is illustrated in Sec. 5 whereas the evaluation results are given in Sec. 6. Section 7 concludes this study.

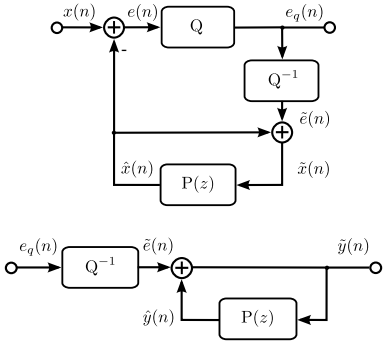


Figure 1: *Typical ADPCM encoder and decoder*

2. ADPCM

The main idea of an ADPCM codec is to solely quantize irredundant signal components. Therefore, a predictor $P(z)$ is applied to estimate a signal $\hat{x}(n)$ using old values of the original input signal $x(n)$. The resulting prediction error $e(n) = x(n) - \hat{x}(n)$ is then quantized, resulting in the quantized prediction error signal $e_q(n)$ which is actually transmitted. At the decoder side, the same predictor as in the encoder is applied to predict the signal $\hat{y}(n)$. The transmitted dequantized prediction error signal $\tilde{e}(n)$ is added to obtain the decoder output signal $\tilde{y}(n)$.

If predictor and quantizer are adaptive, the technique described above is called ADPCM. A block scheme describing ADPCM similar to [8, 9, 10] is depicted in Fig. 1. Since the predictor coefficients are not transmitted, it must be guaranteed that the predictor adaption in encoder and decoder is synchronous. This can be achieved by feeding them with the very same input data. Therefore, the predictor in the encoder is fed with a reconstructed input signal $\tilde{x}(n) = \hat{x}(n) + \tilde{e}(n)$, where $\tilde{e}(n)$ is the dequantized prediction error signal. In other words, the predictor adaption in the encoder is subject to quantization as well. Thus, guaranteeing that $\tilde{x}(n)$ and $\tilde{y}(n)$ are identical. Psychoacoustic knowledge can be involved by applying a set of pre- and post-filters in encoder and decoder to realize noise shaping as shown in [8, 9].

3. CASCADED PREDICTION

Concatenating several predictors and feeding them with the prediction error signal of the corresponding previous predictor is de-

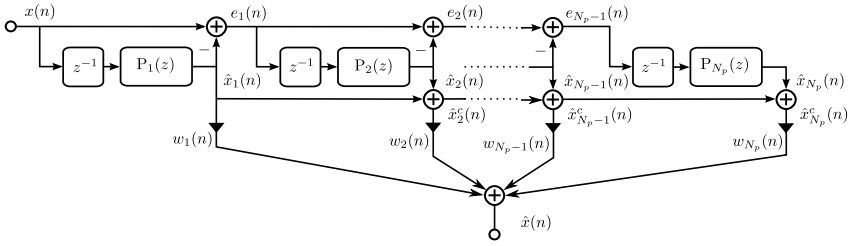


Figure 2: *Cascaded predictors*

noted a predictor cascade structure [13]. Figure 2 illustrates such a structure for N_p predictors. Apparently, the individual prediction signals $\hat{x}_p(n)$ are accumulated $\hat{x}_p^c(n) = \hat{x}_p(n) + \hat{x}_{p-1}^c(n)$ to produce an entire estimated signal per cascade stage.

As it can be seen in Fig. 2, the accumulated predictor outputs $\hat{x}_p^c(n)$ are multiplied with the weights $w_p(n)$ and summed to form the overall predictor output $\hat{x}(n)$. The weight $w_p(n)$ of predictor p is computed using the corresponding predictors prediction error signal

$$w_p(n) \propto e^{(-c(1-\mu)\sum_{i=1}^{n-1}|e_p(n-i)|\cdot\mu^{i-1})}, \tag{1}$$

where $c = 2$ and $\mu = 0.9$ are tuning parameters [12].

This combination of a predictor cascade and *Predictive Minimum Description Length* (PMDL) weighting is called *Weighted Cascaded LMS* (WCLMS) and is used in lossless and lossy [4] codecs.

The advantage of this structure is that it allows to apply differently configured predictors. In other words, predictors of different order M_p and different adaption step sizes λ_p can be used in every cascade stage p, which can be optimized to certain signal characteristics. E.g. higher-order predictors, which adapt slowly, will nicely predict harmonic stationary sounds whereas fast-adapting low-order predictors are superior to follow non-stationary parts. Hence, the overall predictor is expected to adapt to a variety of signals and signal combinations.

An example is given in Fig. 3, where Fig. 3a) shows an excerpt of the trombone sample from the *Sound Quality Assessment Material* (SQAM) [14] dataset. A predictor of order $M_1 = 32$, a faster adapting second predictor of order $M_1 = 4$, and the corresponding cascade of order $M_{1/2} = [32, 4]$ are applied to this signal. The resulting recursively averaged prediction error energy

$$E(n) = (1-\alpha)\, e^2(n) + \alpha\, E(n-1) \tag{2}$$

using $\alpha = 0.999$ is plotted in Fig. 3b). It can be seen that the higher-order predictor produces a clearly smaller prediction error signal than the lower-order predictor except for the first half second where the trombone signal mainly consists of noise until it produces a stable harmonic sound. In this section the smaller-order predictor is superior in terms of error energy. However, the cascade yields an overall result that almost combines the lower bounds of both configurations.

4. EXEMPLARY CODEC

The utilized simple codec for this study is basically structured as shown in Fig. 1. The predictor $P(z)$ is a cascade of lattice filters

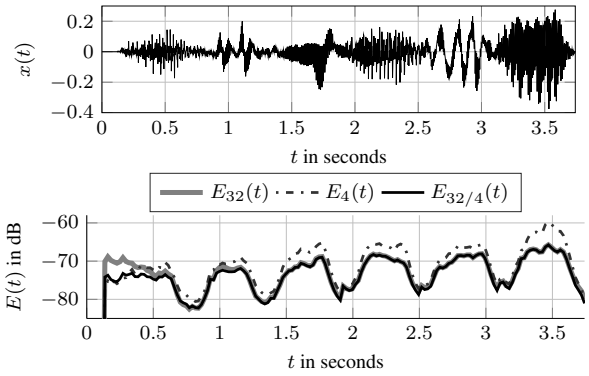


Figure 3: *Input signal and prediction error energy for predictors of order $M \in [32, 4]$ and their cascade. The test signal is a trombone sample from the SQAM dataset.*

in this implementation. These lattice filters are adapted using the *Gradient Adaptive Lattice* (GAL) technique [15] and hence, the lattice predictor cascade is denoted *Weighted Cascaded Gradient Adaptive Lattice* (WCGAL) in the following. The implementation uses power-normalized adaption step sizes

$$\mu_m(n) = \frac{\lambda}{\sigma_m^2(n) + \sigma_{min}}, \tag{3}$$

where m is the lattice stage index, n the sample index, σ_m^2 a recursive error power estimate, and σ_{min} a small offset to avoid a division by zero. λ is the base step size that is used as the main optimization parameter in the following.

The problem of optimally quantizing the prediction error signal $e(n)$ is not considered in this study and hence the quantizer Q is a simple fixed 3 bit quantizer with adaptive scaling. Normalizing the amplitude level is one way to achieve a nearly constant signal variance [16]. The normalization is accomplished by dividing the signal by an estimate of its envelope. The update itself is based on the denormalized quantized signal and hence, is synchronous in encoder and decoder. For more information about the envelope estimation and the calculation of the utilized quantizer levels, the interested reader is referred to [10].

5. OPTIMIZATION

Finding a meaningful combination of prediction order M_p and base step size λ_p for every predictor p of the cascade is a non-trivial task. The authors decided to apply a genetic algorithm to

realize the global optimization of this problem.

Two metrics were considered to create potential cost functions. Initially, a cost function based on the prediction gain

$$G = 10\log_{10}\left(\frac{\sum_{n=0}^{N-1} x(n)^2}{\sum_{n=0}^{N-1} e(n)^2}\right), \tag{4}$$

defined as the logarithmic ratio of an input signal $x(n)$ of length N and the resulting (unquantized) prediction error signal $e(n)$, was used. But it turned out that a pure optimization of the prediction gain yields perceptually unpleasant results.

Therefore, a second cost function based on the so-called *Objective Difference Grade* (ODG) score was applied. The ODG score S is the outcome of the *Perceptual Evaluation of Audio Quality* (PEAQ) [17] method and describes the perceptual quality in terms of coding artifact audibility in a range from -4 (Very annoying) to 0 (Imperceptible). The actual utilized cost function

$$C(\boldsymbol{\chi}_{N_p}) = \sum_{k=1}^{70} S_k^4 | \boldsymbol{\chi}_{N_p} \tag{5}$$

is the sum of the fourth power of all ODG scores $S_k^4|\boldsymbol{\chi}_p$ for all SQAM items k and a certain predictor cascade configuration $\boldsymbol{\chi}_{N_p}$. Raising S_k to the fourth power emphasizes bad results and hence this cost function tends to result in a globally enhanced ODG score instead of predominant excellent and some very poor results as explained in [18].

The optimization routine is repeated for several predictor cascade sizes N_p, a population size of 40, and $15 \cdot N_p$ generations. The range of valid values for the basic step size was restricted to $\lambda \in [1e^{-3}, 0.2]$. The trend of the cost function $C(\boldsymbol{\chi}_{N_p})$ for the predictor configurations over the generations of the genetic algorithm is illustrated in Fig. 4.

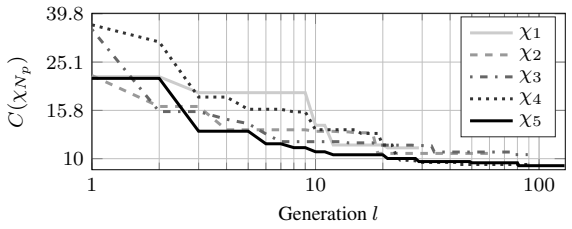


Figure 4: *Cost function $C(\boldsymbol{\chi}_{N_p})$ for cascade parameter vectors $\boldsymbol{\chi}_{N_p}$ over the generations l of the genetic algorithm*

For every generation l the best performing candidate of the population is shown. One can see how $C(\boldsymbol{\chi}_{N_p})$ is decreasing drastically over the generations. Apparantly, the smallest cost function value can be achieved with the highest cascade size and vice-versa. Thereby, the expected gain through the application of a predictor cascade is proven. The cascade configurations and the associated results of the optimization process are denoted in Tab. 1. It denotes for the analyzed configurations c the optimized prediction order M_p, and the optimized adaption base step sizes λ_p.

Note that the genetic algorithm implementation from the MATLAB optimization toolbox with standard settings is used besides the mentioned parameters.

$\boldsymbol{\chi}_{N_p}$	= (Orders M_p	adaption base step size λ_p)
$\boldsymbol{\chi}_1$	(67,	0.0189)
$\boldsymbol{\chi}_2$	(50,4,	0.0020, 0.0423)
$\boldsymbol{\chi}_3$	(58,2,7	0.0016, 0.0098, 0.1036)
$\boldsymbol{\chi}_4$	(76,6,2,2	0.0013, 0.0119, 0.0848, 0.0549)
$\boldsymbol{\chi}_5$	(78,2,2,2,2	0.0010, 0.0199, 0.0779, 0.0052, 0.0024)

Table 1: *Optimized parameter vectors $\boldsymbol{\chi}_{N_p}$ of the prediction cascade for several cascade sizes N_p*

6. EVALUATION

To evaluate the WCGAL-based ADPCM codec structure, the same metrics as in Section 5 applied to the SQAM data set are utilized. The measurements are undertaken using the optimized values from Tab. 1.

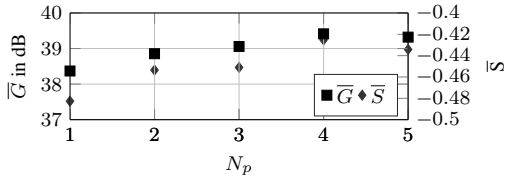


Figure 5: *Prediction gain $\overline{G}$ and ODG scores $\overline{S}$ averaged over all SQAM items for the settings of Tab. 1*

To get an impression of the overall performance, the averaged prediction gain $\overline{G}$ and ODG scores $\overline{S}$ for all configurations from Tab. 1 are illustrated in Fig. 5. Being contrary to the trend of the cost function, where its minimum value was found for the largest cascade size $N_p = 5$, the optimal cascade size in terms of mean ODG scores for the undertaken optimization is $N_p = 4$. In comparison to the single predictor configuration a gain of 0.06 for the mean ODG score and gain of about 1 dB for the mean prediction gain can be achieved.

Analyzing the individual SQAM items by plotting the prediction gain and the ODG score relative to the single predictor configuration $\boldsymbol{\chi}_1$, as done in Fig. 6, reveals the cause for the moderate gain. Despite the cost functions (see Eq. 5) intention of global perceptual enhancement, the individual gains are very signal dependent and occasionally (e.g. $k = [11, 32, 49, 53, 54]$) the predictor cascade even degrades the codecs performance. The mentioned negative examples are the double bass, the triangle, and 3 speech items and hence, a possible explanation could be the noise-like characteristic of those signals.

7. CONCLUSION

The application of cascaded predictors in ADPCM was analyzed in this work. In contrast to its previous application, the WCLMS concept was utilized for gradient-adaptive lattice filters (WCGAL) and in a closed loop codec structure. Order and adaption base step size of predictors of the cascade were optimized using a genetic algorithm by minimizing a cost function, depending on the perceptually motivated ODG score. A simple basic codec was implemented to evaluate this concept. The results for this non-ideal codec already indicate the benefit of cascaded prediction in an ADPCM codec.

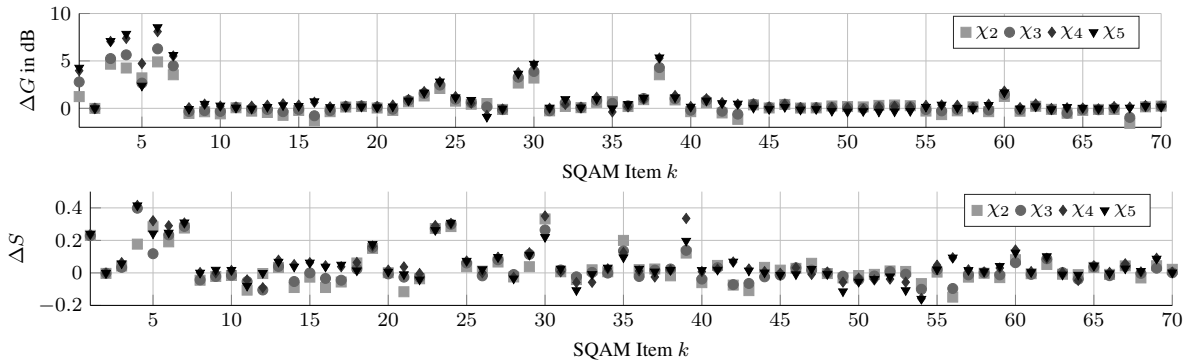


Figure 6: *Prediction gain a) and ODG score gain b) for every SQAM item utilizing the settings of Tab. 1. The results are relative to the ones for* χ_1.

A gain of about 0.06 for the mean ODG score and 1 dB for the mean prediction gain could be achieved without any change of the codecs bitrate. Unfortunately, the presented codec in conjunction with the utilized genetic optimization algorithm could not achieve a global optimization. In other words, the perceptual quality for some items of the used data base degraded. Hence, applying genetic algorithms might not be the optimal solution to find the best predictor cascade configuration.

The application of different optimization approaches, followed by a global optimization of all codec parameters potentially leads to an ADPCM codec offering a very good perceptual quality but featuring algorithmic delay of a single sample. Such a codec can be beneficial in many time-critical applications.

8. REFERENCES

[1] A. Carôt and C. Werner, "Network music performance-problems, approaches and perspectives," in *Proceedings of the Music in the Global Village*, Budapest, Hungary, 2007.

[2] A. Renaud, A. Carôt, and P. Rebelo, "Networked music performance: State of the art," in *Proceedings of the AES 30th International Conference*, Saariselkä, Finland, 2007.

[3] A. Carôt, C. Werner, and T. Fischinger, "Towards a Comprehensive Cognitive Analyisis of Delay-Influenced Rhytmical Interaction," in *Proceedings of the International Computer Music Conference (ICMC 2009)*, Montreal, Canada, 2009.

[4] J. Hirschfeld, J. Klier, U. Kraemer, G. Schuller, and S. Wabnik, "Ultra low delay audio coding with constant bit rate," in *AES Convention 117*, San Francisco, USA, Oct 2004.

[5] J. Hirschfeld, U. Kraemer, G. Schuller, and S. Wabnik, "Reduced bit rate ultra low delay audio coding," in *AES Convention 120*, Paris, France, May 2006.

[6] J.M. Valin, G. Maxwell, T. Terriberry, and K. Vos, "High-Quality, Low-Delay Music Coding in the Opus Codec," in *AES Convention 135*, New York, USA, 2013.

[7] J.M. Valin, T. Terriberry, C. Montgomery, and G. Maxwell, "A High-Quality Speech and Audio Codec With Less Than 10-ms Delay," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 1, Jan. 2010.

[8] M. Holters, O. Pabst, and U. Zölzer, "ADPCM with Adaptive Pre- and Post-Filtering for Delay-Free Audio Coding," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, Honolulu, USA, April 2007.

[9] M. Holters and U. Zölzer, "Delay-free lossy audio coding using shelving pre- and post-filters," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008)*, Las Vegas, USA, March 2008.

[10] M. Holters, C.R. Helmrich, and U. Zölzer, "Delay-Free Audio Coding Based on ADPCM and Error Feedback," in *Proc. of the 11th Conference on Digital Audio Effects (DAFx-08)*, Espoo, Finland, 2008.

[11] G. Simkus, M. Holters, and U. Zölzer, "Error resilience enhancement for a robust adpcm audio coding scheme," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, May 2014.

[12] G. Schuller and A. Hanna, "Low delay audio compression using predictive coding," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2002)*, Orlando, USA, May 2002.

[13] P. Prandoni and M. Vetterli, "An FIR Cascade Structure for Adaptive Linear Prediction," *IEEE Transactions on Signal Processing*, vol. 46, September 1998.

[14] European Broadcast Union, "EBU Tech. 3253-E: Sound quality assessment material," April 1988.

[15] Lloyd J. Griffiths, "A continuously-adaptive filter implemented as a lattice structure," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '77)*, Hartford, USA, May 1977.

[16] D. Cohn and J. Melsa, "The relationship between an adaptive quantizer and a variance estimator (corresp.)," *IEEE Transactions on Information Theory*, vol. 21, Nov 1975.

[17] International Telecommunication Union, "ITU Recommendation ITU-R BS.1387: Method for objective measurements of perceived audio quality," November 2001.

[18] M. Holters and U. Zölzer, "Automatic parameter optimization for a perceptual audio codec," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009)*, Taipei, Taiwan, April 2009.

BEAT HISTOGRAM FEATURES FOR RHYTHM-BASED MUSICAL GENRE CLASSIFICATION USING MULTIPLE NOVELTY FUNCTIONS

Athanasios Lykartsis
Audio Communication Group
Technische Universität Berlin
Berlin, Germany
athanasios.lykartsis@tu-berlin.de

Alexander Lerch
Center for Music Technology
Georgia Institute of Technology
Atlanta, Georgia, US
alexander.lerch@gatech.edu

ABSTRACT

In this paper we present beat histogram features for multiple level rhythm description and evaluate them in a musical genre classification task. Audio features pertaining to various musical content categories and their related novelty functions are extracted as a basis for the creation of beat histograms. The proposed features capture not only amplitude, but also tonal and general spectral changes in the signal, aiming to represent as much rhythmic information as possible. The most and least informative features are identified through feature selection methods and are then tested using Support Vector Machines on five genre datasets concerning classification accuracy against a baseline feature set. Results show that the presented features provide comparable classification accuracy with respect to other genre classification approaches using periodicity histograms and display a performance close to that of much more elaborate up-to-date approaches for rhythm description. The use of bar boundary annotations for the texture frames has provided an improvement for the dance-oriented Ballroom dataset. The comparably small number of descriptors and the possibility of evaluating the influence of specific signal components to the general rhythmic content encourage the further use of the method in rhythm description tasks.

1. INTRODUCTION

The extraction of features describing musical rhythm is an important and a challenging topic in Music Information Retrieval (MIR) with applications in many areas [1, 2, 3, 4]. Rhythm features can be of importance, for example, in musical genre classification, where they have been shown to improve classification accuracy and, in specific cases (such as for special, dance-music oriented datasets) allow very successful rhythm-based classification or similarity computation [2, 5, 6, 7, 4]. Since the concept of 'genre' can be related to various musical parameters [8, 9, 5], it is generally attempted to include features from all possibly relevant musical categories (timbre, pitch, loudness, and rhythm) in order to achieve good classification results. In this paper, we chose to focus on rhythm features for musical genre classification since computational rhythm description remains a challenging subject, even when considerable advances have been made lately in this field [7, 10, 11, 4]. A common approach to the description of the rhythmic content of a musical track focuses on the extraction of features based on changes in the signal's amplitude envelope, possibly applying frequency-band filtering or attempting to track energy changes from the signal's short time spectrum. The basic assumption behind this approach is that rhythm is strongly related to long-term amplitude periodicities and their statistical properties [2, 5, 6]. This assumption is, in our view, well-founded, since many music theoretical works on rhythm have stressed the multidimensional nature of rhythm, establishing the need to consider different musical properties and their temporal evolution when attempting to represent the rhythmic content of music [12, 13, 14, 15].

Several approaches for the automatic extraction of features describing rhythmic content for genre classification through a periodicity distribution representation have been proposed [16, 17, 18]. They are based on earlier studies on beat analysis [19], which aimed at creating a very low frequency periodicity representation (later commonly dubbed the *beat histogram* [16] or *periodicity histogram* [18]), used to extract musical parameters such as tempo as well as low-level features (e.g., statistical properties of the histogram such as mean or standard deviation). Traditionally, a measure of the signal amplitude envelope or its change over time is utilized as the novelty function for the extraction of a beat histogram [18, 16, 17]. Genre classification systems based on such representations have generally shown promising results, although the rhythm features did not perform as well as other subsets of the feature set such as timbre features [5, 16, 17]. There still exist only relatively few studies [20, 21, 18, 22, 3] where rhythm features alone were so efficient as to achieve high accuracy in a genre classification or similarity task. The dataset used in most of those cases (Ballroom [23]) includes only tracks from dance genres with presumably clearly distinguishable rhythmic properties, leading to very high performance but providing little information as to the suitability of the features for music with less distinctive rhythm.

Additionally, there exist many approaches for rhythm description which do not only base themselves on direct feature extraction from a periodicity representation, but rely on direct similarity measurements, computing distances between rhythm representations also being based on periodicities present in the signal [24, 25, 26, 7]. Finally, latest works have attempted to model rhythm using probabilistic models [10, 3, 27] or deep neural networks [11, 28]. Such methods have shown excellent results in rhythmic classification and similarity computation tasks (with accuracies ranging up to 95% [3, 27, 11]), attesting to the plausibility of addressing rhythm-based genre classification tasks with rhythmic similarity methods or elaborate rhythm modeling [4]. However, we identify two main drawbacks with this category of approaches: first, the interpretability of the features describing rhythm is limited, since the method of their generation is either too complex or based on purely technical and not music-theoretical considerations. This is not a problem, of course, if the goal is to develop a method providing high accuracy; it does, however, limit the possible conclusions which can be drawn from the classification task in itself and the

features involved in it. Secondly, the complexity of the methods makes them prone to errors and random influences. Furthermore, the very high results achieved by some studies, although proving the suitability of such classifier systems for e.g. commercial use, should in our view be seen in a critical light for two reasons: on the one hand, it is questionable whether systems based only on rhythmic features could theoretically achieve such performance, since listening experiments show that even human subjects cannot distinguish genres perfectly [29]; on the other hand, it has been demonstrated very often in the intrinsically fuzzily defined genre classification task that complex systems relying on elaborate transformations and trained with small amounts of data might provide very high results, but in effect do not generalize to real-world data (a case of *overfitting*) or do not describe the quantities they are supposed to, their performance being an artifact of an erroneous ground truth or highly dataset-specific features [30, 31, 32]. We chose to focus on the periodicity representation methods and the features which can be extracted on their basis, conducting a detailed examination of the features which can be used in their context and their behavior for many different datasets. This approach allows to investigate the merits of those methods in depth and to identify which signal-based features bear the most importance for rhythmic description.

A common element of previous studies using periodicity representations allowing rhythmic content feature extraction for genre classification is that the features are derived from a beat histogram created on the basis of the signal amplitude and energy changes, as for example in [16, 17, 18]. This approach might seem intuitive at first, since it is based on the assumption that rhythmic properties are conveyed through amplitude or energy changes in the musical signal, which is a common consequence of rhythm perception modalities [33, 34]. However, relevant literature in music theory [12, 15, 14] and cognition [35, 36] indicates that rhythm arises as the combination or interplay of periodicities associated with different sound properties such as *amplitude* (e.g., accents), *spectral* (e.g., instrumentation changes) and *tonal* changes (e.g., chord changes). In the field of onset detection, novelty functions have been proposed which take into account spectral content changes [37, 38, 39]. The goal of this work is to capitalize on this observation by extracting novelty functions from different signal properties (which will allow to take not only amplitude-based but also spectral and tonal changes and their related periodicities into account), using the beat histogram method to create a basis for features describing the rhythmic content of the signal. We investigate the impact of using various musical qualities as novelty functions for beat histogram calculation on classification accuracy, identify the most and least descriptive features and feature groups, and compare the results to those of a traditional timbre-related feature set. Furthermore, the effect of bar boundary annotations from manually annotated data is investigated, since beat histograms based on musically more meaningful data are expected to produce more qualitative features. Finally, we examine the misclassifications in the confusion matrices to draw conclusions on the suitability of the features for rhythm-based classification and possible shortcomings.

The paper is structured as follows. In Section 2, the feature extraction procedure is described. In the third Section, the evaluation of the proposed features is given, along with information about feature selection and the datasets. In the fourth Section, results are presented and discussed in Section 5. Finally, we give conclusions and suggestions for future work (Section 6).

2. METHOD

Novelty functions are generally defined as temporal curves designating prominent changes of a signal [40, 41, 42, 43], resulting from a reduced or filtered version of the original signal from which the first difference is computed, in order to accentuate substantial changes in the monitored quantity. In this paper, we expand this definition somewhat and consider every temporal trajectory of a signal feature to effectively be a novelty function in order to use it as a basis for the beat histogram calculation. This assumption is justified from a practical point of view, since prominent changes in the magnitude of a signal feature (such as, e.g., spectral flux) are still represented (but not as accentuated), their inherent periodicities detectable by methods such as a Discrete Fourier Transform (DFT), an Auto-Correlation Function (ACF) or a resonant filter bank. Furthermore, the avoidance of taking the first difference function has the added advantage of reducing noise in the feature temporal trajectory. For rhythm analysis, signal characteristics beyond amplitude are included here since their periodicities also contribute to the overall rhythm. Examples include changes in instrumentation, which cause a universal, broadband change in the spectral content of a signal, or chord changes, which can be tracked by changes in the instantaneous pitch content.

Fig. 1 shows the novelty function (top) and the resulting beat histogram (bottom) for two features representing spectral change (Spectral Flux) and spectral shape (Spectral Flatness), respectively. The audio track is an excerpt from the *disco* class of the GTZAN dataset [16] and the features are extracted over the whole length of a single texture frame (3 s). In the excerpt, the drum section plays a straight $4/4$ measure with a beat each 0.48 s, clearly visible in the spectral flux temporal curve (tracking general spectral change) and as the prevalent periodicity (126 BPM) in the corresponding beat histogram. The changes tracked by the spectral flatness measure, however, more strongly reflect tonalness changes, taking place every 0.96 s, with a main periodicity of 63 BPM. Both the general form (distribution) of the histogram and the strength and exact BPM value of the most prevalent periodicities can be seen to differ significantly in the two examples. Based on this example, it is obvious that the changes tracked by different novelty functions can lead to different beat histograms for the same audio excerpt, with each histogram potentially providing a rhythmically meaningful description.

The novelty functions used in this paper cover envelope, spectral shape, and tonal content changes. The selected features are the following (for details on their computation, see [44]):

- **Spectral Shape** features include Spectral Flux (SF), Spectral Centroid (SCD), Mel Frequency Cepstral Coefficients (MFCC 1-13) and Spectral Flatness (SFL).
- **Tonal** novelty features comprise Spectral Tonal Power Ratio (STPR) and the Pitch Chroma Coefficients (SPC 1-12).
- **Envelope** novelty is tracked through the Root Mean Square (RMS) measure of the signal amplitude.

The beat histogram computation is similar to the one proposed by Tzanetakis [16] without implementation of the wavelet filtering. All features are calculated through a Short-Time-Fourier-Transform (STFT), except for the RMS which is extracted with the same temporal resolution parameters from the time domain signal. The complete procedure for the generation of a feature vector representing each track includes the following steps:

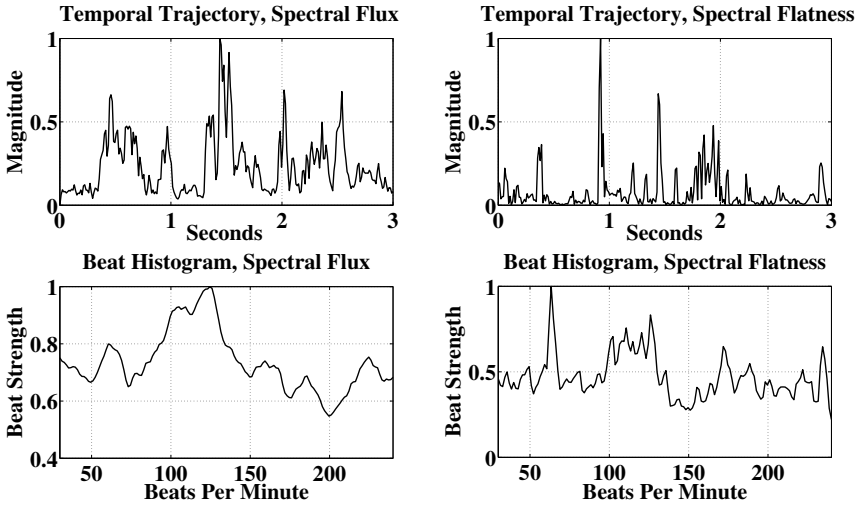


Figure 1: *Temporal Trajectory of Spectral Flux and Spectral Flatness (upper row) and corresponding Beat Histograms from* $30-240$ *BPM (lower row).*

1. **Preprocessing**: the audio signal is down-mixed to mono, resampled to 22.5 kHz, lowpass-filtered to remove DC components and normalized.

2. **STFT**: the transform is computed with a frame-length of approximately 46.4 ms, windowed by a Hann window and with 75% overlap between consecutive frames. The resulting frequency resolution is $\Delta f = 10.77\,\mathrm{Hz}$.

3. **Novelty function**: the features listed earlier are extracted from the STFT frames, and the novelty function is computed through the calculation of the temporal trajectory and half-wave rectification, as is commonly practiced in novelty function extraction [43].

4. **Beat histogram**: the beat histogram is extracted through the calculation of an Autocorrelation Function (ACF) for each texture window of length 3 s. The overlap of the texture windows is 75%.

5. **Subfeature computation**: for each beat histogram, the subfeatures listed in Table 1 are extracted. The concatenation of all subfeature groups for each novelty function produces the final feature vector for an audio excerpt.

Similar subfeatures can be found e.g. in [16] (Peak), and [17, 18] (Distribution). In total, 30 novelty functions are used for the production of as many beat histograms, from each of which 19 subfeatures are extracted, resulting in a total count of 570 features. This effectively means that from the temporal trajectory of every MFCC or chroma coefficient, as well as the other features mentioned in the previous paragraph, a beat histogram is extracted, ensuring that all relevant periodicities and their properties — in different frequency bands and describing various audio aspects — are accounted for.

Table 1: *Subfeatures extracted from Beat Histograms.*

Distribution	Peak
Mean (ME)	Salience of Strongest Peak (A1)
Standard Deviation (SD)	Salience of 2nd Stronger Peak (A0)
Mean of Derivative (MD)	Period of Strongest Peak (P1)
SD of Derivative (SDD)	Period of 2nd Stronger Peak (P2)
Skewness (SK)	Period of Peak Centroid (P3)
Kurtosis (KU)	Ratio of A0 to A1 (RA)
Entropy (EN)	Sum (SU)
Geometrical Mean (GM)	Sum of Power (SP)
Centroid (CD)	
Flatness (FL)	
High Frequency Content (HFC)	

3. EXPERIMENTAL SETUP AND EVALUATION

In order to be able to compare the results for the rhythm content features we extracted a *baseline feature set* by calculating the feature value over all texture windows of an excerpt (keeping the average value inside a window) without extracting periodicities. Those features are considered as a baseline since they represent a standard set of features used in genre classification. They include all features listed in Sect. 2 except the *Pitch Chroma Coefficients* and additionally include the features *Spectral Spread, Peak Amplitude Value*, and *Zero Crossing Rate*. The subfeatures on each of these features' temporal trajectory throughout each track are given in the *Distribution* column of Table 1. In total, the baseline feature set comprises 21 features times 11 subfeatures $=$ 231 features. We chose not to include other, more state-of-the-art non-rhythmic features in the baseline mainly because we aimed at investigating rhythmic features rather than non-rhythmic features. Second, since the main goal of the paper was to evaluate the beat histogram fea-

tures derived from all relevant novelty functions of the signal, it seemed plausible to include as a baseline the same novelty functions. This, however, was performed without applying the beat histogram transformation, so as to assess the added value of this certain processing step and the resulting features.

3.1. Classification

For the classification part, we apply the Support Vector Machines (SVM) [45] algorithm under MATLAB with a Radial Basis Function (RBF) kernel. The two hyperparameters for this kernel C and γ were determined with a grid search procedure. For all the experiments presented here, a 10-fold cross-validation took place, and results are averaged over the folds. The goal of the given classification setup is to compare the performance of the rhythm content feature set to a standard set of features, while also testing the combined set in order to assess the improvement when using the rhythm feature set in addition to the baseline set. All features are subjected to standardization (z-score) prior to classification (train and test set separately). The performance measure used to evaluate the classification is *accuracy*, defined as the proportion of correctly classified samples to all samples classified.

3.2. Datasets

In order to ensure comparability of the results with other publications and to evaluate the rhythm features for different genre hierarchies and tracks, five datasets were evaluated:

- GTZAN [16]
- Ballroom [18, 23]
- ISMIR04 [46]
- Unique [47]
- Homburg [48]

Although none of the datasets raises claims to being either complete or error-free (since the definition of genre itself is a debatable subject), their previous extensive use makes them suitable as benchmarks for the musical genre classification task. In order to avoid having results which could be artifacts of one specific dataset (e.g., for the GTZAN which has been criticized for its content [31], or the Ballroom which has been seen to be easily classifiable by the tempo feature [18]), we chose to include five very diverse datasets here. An interesting task which came under consideration would be training with one dataset and testing with another. However, the different class/genre structure across them did not allow for such a use in this context.

3.3. Feature Selection

The large number of resulting features mentioned in Sect. 2 makes direct feature evaluation a tedious task. In order to identify the best and worst performing descriptors we conducted a two stage feature selection: First, we apply a filter method (Mutual Information with Target Data [49], using the maximum relevance CMIM metric [50] from the MI-Toolbox [51]). Second, we run a sequential forward feature selection using the SVM as a wrapper method [52]. We retain the N best features (the feature number N is dataset-dependent, but ranges between 10 and 20) which gave comparable to or better accuracy than the full feature sets. This procedure is applied to the baseline and rhythm feature sets separately, and the best features from both are then pooled to produce the combined feature set. Finally, we apply feature selection by separating feature subsets in the rhythm set in order to determine the effectiveness of different novelty functions and subfeature groups.

3.4. Bar Annotation

The parameters for the rhythm feature extraction algorithm were given in Sect. 2. It is a common problem of many such algorithms that the overlapping texture windows used in the block-wise processing of the audio file are of pre-defined length that does not necessarily represent "meaningful" parts of the music, such as, e.g., a bar. Placing the frames exactly at the boundaries of a bar or a musical phrase could increase the precision of the beat histogram representation, since the periodicities extracted from the segment would be musically meaningful, without onsets added or being left out because of random framing. In order to adapt the texture window boundaries to the bar boundaries, annotations of the audio files are necessary. In the case of the Ballroom dataset, such a manual annotation is available [53] and will be used here.

4. RESULTS

Results of the classification after feature selection for all datasets are presented in Fig. 2 and Table 2. The *priors* (percentage of a class/genre samples in a dataset) as average (Avg) and of the greatest class (Max P) are also provided. Sample results of the feature selection process concerning feature ranking (the three best and three worst features per dataset) are given in Table 3. Apart from that, in Fig. 3, accuracy results for all novelty function and subfeature groups for each dataset can be seen. Finally, confusion matrices are provided for the GTZAN and Ballroom datasets (Tables 4 and 5) for the rhythmic feature set in order to examine the misclassifications for those very well-known datasets.

Some tendencies can be clearly identified: The baseline feature set performs always better than the rhythm feature set alone except for the Ballroom dataset. The difference is mostly small but also significant at the 0.05 level in all cases except for the ISMIR04 dataset (based on a comparison test of the Cohen's Kappa extracted from the confusion matrices [54]); it ranges from 1.9% for the ISMIR04 dataset to 12.3% for the GTZAN dataset. Only for the Ballroom dataset the accuracy using the rhythm feature set is 6.8% above that observed when using the baseline set. The combined feature set outperforms the individual sets in all cases, the achieved accuracy being very close to that of the baseline feature set. This difference in accuracies is significant at the 0.05 level only in two cases (Ballroom and Unique). With regard to the datasets, results show accuracies in the area of 44.6 (rhythm feature set, Homburg) to 72.8% (combined feature set, GTZAN). The best performance of the rhythm features can be observed for the Ballroom dataset (67.7%), whereas the poorest performance can be found for the Homburg dataset. It should be noted that for the unbalanced datasets (i.e., all except GTZAN) it was observed from inspecting the confusion matrices that the achieved accuracy is mostly influenced by the most prominent class being classified correctly, whereas for the other classes, the performance is inferior but still in most cases above the prior of the respective class. An important result is the performance of the Ballroom dataset when using the annotated bars as texture window length (Table 2): It shows a clear improvement for the rhythm feature set alone and reaches 88.4% for the combined set.

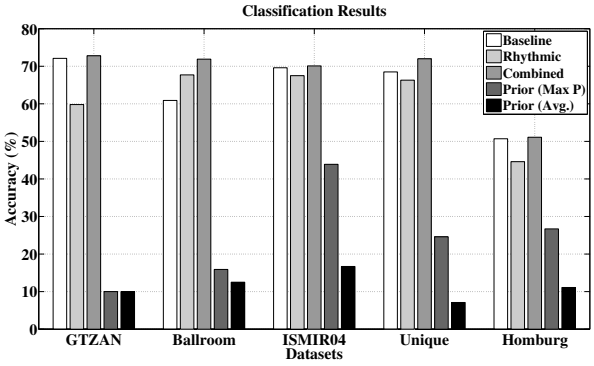


Figure 2: *Classification results after feature selection.*

Table 2: *Classification results (accuracy in %) for various settings and best classification results. For the Ballroom dataset, a second accuracy value is reported for the case of using the manual annotation for frame boundaries.*

Setting	GTZAN	Ballroom	ISMIR04	Unique	Homburg
Baseline	72.1	60.9	69.6	68.5	50.7
Rhythmic	59.3	67.3/76.7	67.5	66.3	44.6
Combined	72.8	71.9/88.4	70.1	72	51.1
P.-Max P	10	15.9	43.9	24.6	26.7
P.-Avg	10	12.5	16.7	7.1	11.1

Concerning feature selection, it can be seen in Table 3 that the best features resulting from the information-theoretical feature selection procedure comprise proportionally more features based on spectral (SF, SFL) or amplitude (RMS) novelty, whereas with respect to the subfeatures the image is not so clear — only the SD feature appears more consistently (4 times) in the first ranks. The worst features can be seen to be based on tonal (STPR, SPC) novelty functions, with subfeatures giving a clearer image, with Peak features such as the A1 being frequent in the ranking.

Those tendencies can also be partially observed in Fig. 3, where the feature groups were tested individually: the only novelty functions showing to provide slightly better results on its own for all datasets are the SFL, MFCC2, STPR and RMS, whereas other MFCC and SPC features show relatively lower performance on their own. However, the performance of all novelty functions appears to be relatively similar, except for the SFL, STPR and RMS novelty functions which seem to provide higher performance for all datasets. In the case of the subfeature groups, no specific feature seems to be standing out.

Table 3: *Best and worst features after feature selection. Abbreviation left of point denotes subfeature, otherwise novelty function.*

Rank	GTZAN	Ballroom	ISMIR04	Unique	Homburg
1	MD.RMS	P1.SF	MD.MFC2	SD.MFC1	SD.RMS
2	FL.RMS	A0.SFL	CD.MFC1	GM.SFL	SD.SPC3
3	GM.SFL	SD.SPC3	A0.SF	MD.MFC2	FL.SFL
568	A0.STPR	A0.STPR	SP.MFC3	A0.STPR	A1.MFC2
569	SP.MFC1	A1.MFC1	A1.RMS	A1.MFC3	A0.MFC2
570	A1.RMS	EN.MFC1	A0.STPR	A0.MFC3	A1.MFC1

Table 4: *Confusion matrix for Ballroom dataset, average accuracy:* 67.3%. *Accuracy and Prior are given in* %.

	Ch.	Ji.	Qu.	Ru.	Sa.	Ta.	Vw.	Wa.
Ch.	87	4	3	4	8	3	2	0
Ji.	9	40	1	1	6	2	1	0
Qu.	2	5	50	5	10	6	3	1
Ru.	10	0	3	62	0	5	2	16
Sa.	9	6	7	3	55	4	2	0
Ta.	3	0	9	2	2	58	7	5
Vw.	0	0	11	9	0	5	25	15
Wa.	0	0	2	12	0	1	2	93
Acc.	78	67	61	63	64	67	38	85
Pr.	15.9	8.6	11.7	14.0	12.3	12.3	9.3	15 .8

Table 5: *Confusion matrix for GTZAN dataset, average accuracy:* 59.3%. *Accuracy and Prior are given in* %.

	Bl.	Cl.	Co.	Di.	Hi.	Ja.	Me.	Po.	Re.	Ro.
Bl.	59	2	4	4	5	5	8	1	7	6
Cl.	2	79	1	1	0	11	2	0	1	5
Co.	11	3	56	4	2	7	3	1	1	13
Di.	3	0	4	56	6	3	7	8	6	10
Hi.	3	0	0	8	64	2	2	5	13	3
Ja.	7	13	4	1	0	64	4	1	3	3
Me.	2	2	1	8	1	1	72	2	0	12
Po.	7	2	3	9	5	4	1	42	10	16
Re.	6	1	3	6	14	4	0	4	59	3
Ro.	7	4	8	4	4	4	15	9	4	42
Acc.	59	79	56	56	64	64	72	42	59	42
Pr.	10	10	10	10	10	10	10	10	10	10

5. DISCUSSION

The results given in Table 2 provide strong support for the view that the presented rhythm features can perform in the same range as non-rhythmic baseline features. The overall better performance of the combined feature set is a consequence of using information related to both timbre and rhythm features, with significant improvement for two out of five datasets.

Concerning the datasets, the poor classification performance observed for the Homburg dataset could be an indication that the latter has a special genre or track selection, which cannot be predicted efficiently using our features. The very similar results observed for all feature sets for the ISMIR04 dataset is most probably due to its largely unbalanced character. The Ballroom dataset stands out as a good example where the rhythm features alone could offer a satisfactory performance. The results reported here are comparable to or better than those reported in studies using similar methods [16, 17, 18], but lie below those of newer studies employing more sophisticated features which depart from the beat histogram [55, 7, 10, 3, 56, 11, 57]. However, the simplicity of the beat histogram calculation and feature extraction and the possibility to assess features and feature groups individually are, in our view, advantages which have to be taken into account.

The use of bar boundaries as texture window boundaries for the Ballroom dataset gave encouraging results: A maximum accuracy of 76.7% was achieved, which is a notable improvement to the 67.7% with fixed-length segmentation. The result is close to the one reported by Gouyon et al. [18] using features extracted

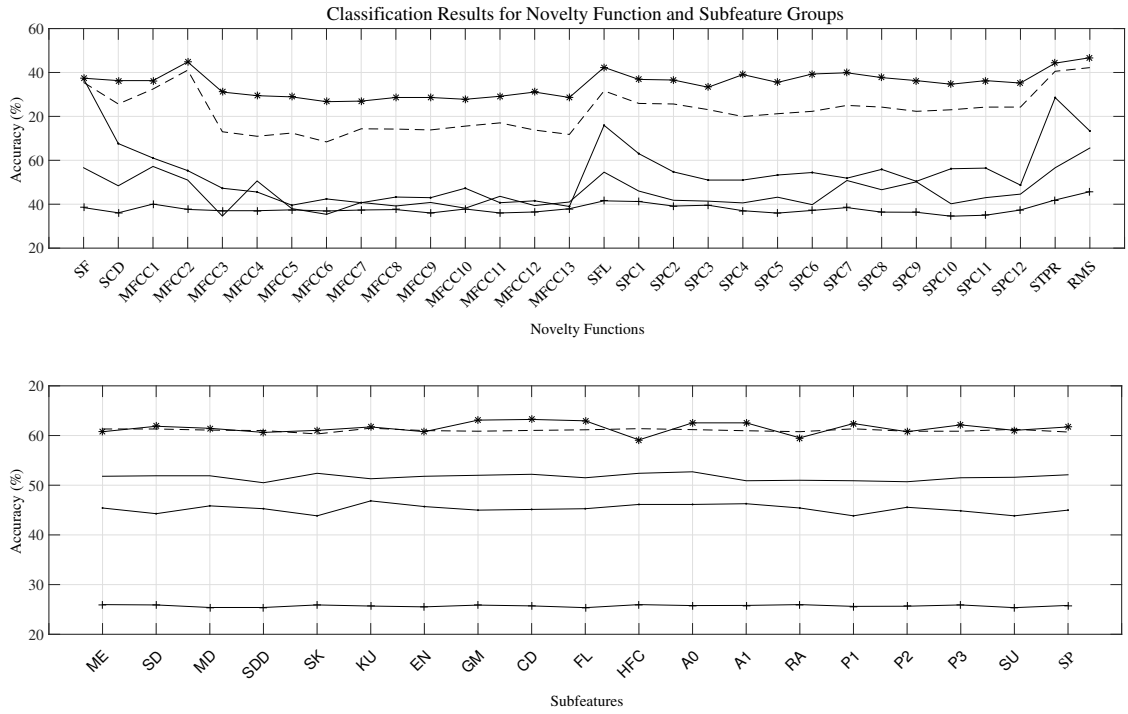


Figure 3: *Classification results for feature groups. Upper graphic shows Novelty Function, lower Subfeature groups. Datasets are denoted by the following line styles: "-" (GTZAN), ".-" (Ballroom), "*-" (ISMIR04), "–" (Unique), "+-" (Homburg).*

from the beat histogram. This suggests that the application of prior knowledge regarding the "real" boundaries of the musical surface can help to considerably improve the accuracy of rhythm-based genre classification.

The misclassifications (reported in Tables 4 and 5) show that genre confusions are those that could be expected when using features capturing the rhythmic character of the pieces, thus making it difficult to distinguish between genres containing similar patterns: for the Ballroom dataset, the most prominent misclassifications take place between Rumba/Waltz, Quickstep/Samba and Waltz/Viennese Waltz. For the GTZAN dataset, Classical is confused with Jazz, Country with Rock, Hiphop with Reggae and Pop with Rock and Metal. Those results indicate that the multiple novelty-based beat histogram features indeed capture the specific rhythmic properties of the genres and the regularities of their periodicities, pointing towards the suitability of those features for the extraction of general rhythmic properties.

With regards to the feature selection, it was shown that mainly novelty functions which are based on amplitude or spectral shape changes in the signal give the best results, a result which could be confirmed both from the standard feature selection procedure, as well as from the feature subset-based selection. Those results can be due to important turning points of a track in most popular western music being mostly mediated through loudness, energy or spectral form (for example, in the case of many instruments playing together in a bar change or a new instrument entering the scene) which leads to those features possessing more salient novelty functions out of which more qualitative beat histogram features can be extracted. Another reason could be the role of those novelty functions in stressing the basic metric positions in most of western popular music. Concerning the subfeatures and the results of the information-theoretical feature selection (Table 3), the higher performance of simple statistics such as the SD of the beat histogram attest to them having more discriminative power, possibly because they express very basic statistical tendencies of the periodicity distribution. Furthermore, higher-level features (such as the P1, which is a good estimate of the excerpts' tempo, already shown to be important in rhythm-based genre classification [18]), also possess discriminative power since their value is an important indicator of a genre character (e.g. dance vs. classical music). However, since no such tendencies are observed in the feature group selection (Fig. 3), it can be deducted that no specific subfeature on the beat histogram bears special discriminative power, but it is rather the combination with a salient novelty function which allows for better results.

6. CONCLUSIONS

The work presented in this paper focuses on the creation of novel features for rhythm-based musical genre classification. The difference in comparison to previous studies in the field [16, 17, 18], using the signal amplitude envelope only, is the use of the tempo-

ral trajectory of other signal quantities such as SF as the novelty function for the calculation of the beat histogram. We showed that performance using these beat histogram features is higher or in a similar range than related work using periodicity histograms. It has also been shown that specific novelty functions relating to amplitude or spectral shape are among the most informative when analyzed with a periodicity representation method. Finally, we showed the positive impact of manual bar-boundary annotation for the extraction of rhythm features on classification performance.

There are many more features which can be considered as novelty functions [38, 43], as well as possible subfeatures on the beat histogram (such as MFCCs, presented in [18]) and other methods for the periodicity representation calculation [19, 17, 18]. Future goals include an even more extensive and detailed feature selection, identifying the features or feature groups which are informative with respect to specific genres (in order to associate specific novelty functions with relevance to specific genres) and a test of the feature robustness against signal degradations. Those subfeatures and novelty functions could be then suitable for future use in more specific rhythm description tasks. Furthermore, the investigation of optimal parameter settings for feature extraction and classification, the utilization of other classification methods and performance evaluation measures, as well as the usage of other methods for feature aggregation are other possible research directions. The high accuracy achieved especially for the Ballroom dataset indicates the suitability of the descriptors for further application and points to the importance of bar-boundary annotation (which can also be performed automatically) for rhythm features.

While the features for beat histogram calculation have been evaluated only in the context of genre classification, we believe that they will prove useful in other tasks as well. Future research will concentrate on adjusting and using rhythm content features for MIR tasks such as audio similarity and mood recognition. Preliminary research has also been undertaken concerning the use of novelty functions of specific instruments (e.g. Drums) extracted through Non-Negative Matrix Factorization (NMF) [58], or the application of the features to other signals, such as speech [59]. Their application in a task of automatic spoken language identification based on the rhythmic elements of speech has shown promising results and points to further research directions.

7. REFERENCES

[1] Enric Guaus i Termens, "New approaches for rhythmic description of audio signals," Tech. Rep., Universitat Pompeu Fabra, Music Technology Group, 2004.

[2] Fabien Gouyon and Simon Dixon, "A review of automatic rhythm description systems," *Computer Music Journal*, vol. 29, no. 1, pp. 34–35, 2005.

[3] Geoffroy Peeters, "Spectral and temporal periodicity representations of rhythm for the automatic classification of music audio signal," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 5, pp. 1242–1252, 2011.

[4] Tlacael Miguel Esparza, Juan Pablo Bello, and Eric J Humphrey, "From genre classification to rhythm similarity: Computational and musicological insights," *Journal of New Music Research*, , no. ahead-of-print, pp. 1–19, 2014.

[5] Nicolas Scaringella, Giorgio Zoia, and Daniel Mlynek, "Automatic genre classification of music content: a survey," *IEEE Signal Processing Magazine*, vol. 23, no. 2, pp. 133–141, March 2006.

[6] Enric Guaus i Termens, *Audio content processing for automatic music genre classification: descriptors, databases, and classifiers.*, Ph.D. thesis, Universitat Pompeu Fabra, Barcelona, 2009.

[7] Tim Pohle, Dominik Schnitzer, Markus Schedl, Peter Knees, and Gerhard Widmer, "On rhythm and general music similarity.," in *ISMIR*, 2009, pp. 525–530.

[8] François Pachet, Daniel Cazaly, et al., "A taxonomy of musical genres," in *Proceedings of the Conference on Content-Based Multimedia Information Access*, 2000.

[9] Jean-Julien Aucouturier and Francois Pachet, "Representing musical genre: A state of the art," *Journal of New Music Research*, vol. 32, no. 1, pp. 83–93, 2003.

[10] Andre Holzapfel, Arthur Flexer, and Gerhard Widmer, "Improving tempo-sensitive and tempo-robust descriptors for rhythmic similarity," in *Proceedings of the 8th Sound and Music Computing Conference*, 2011.

[11] Aggelos Pikrakis, "A deep learning approach to rhythm modelling with applications," in *6th International Workshop on Machine Learning and Music (MML13)*, 2013.

[12] Grosvenor Cooper, *The rhythmic structure of music*, vol. 118, University of Chicago Press, 1963.

[13] Paul Fraisse, "Rhythm and tempo," in *The psychology of music*, Diana Deutsch, Ed., Series in Cognition and Perception, chapter 6. Academic Press, 1982.

[14] Fred Lerdahl and Ray S Jackendoff, *A generative theory of tonal music*, MIT press, 1983.

[15] Joel Lester, *The rhythms of tonal music*, Pendragon Press, 1986.

[16] George Tzanetakis and Perry Cook, "Musical genre classification of audio signals," *IEEE Trans. on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[17] Juan José Burred and Alexander Lerch, "A hierarchical approach to automatic musical genre classification," in *DAFX*, 2003.

[18] Fabien Gouyon, Simon Dixon, Elias Pampalk, and Gerhard Widmer, "Evaluating rhythmic descriptors for musical genre classification," in *AES*, 2004.

[19] Eric D Scheirer, "Tempo and beat analysis of acoustic musical signals," *The Journal of the Acoustical Society of America*, vol. 103, no. 1, pp. 588–601, 1998.

[20] Elias Pampalk, Simon Dixon, and Gerhard Widmer, "Exploring music collections by browsing different views," *Computer Music Journal*, vol. 28, no. 2, pp. 49–62, 2004.

[21] Simon Dixon, Fabien Gouyon, Gerhard Widmer, et al., "Towards characterisation of music via rhythmic patterns," in *ISMIR*, 2004.

[22] Andre Holzapfel and Yannis Stylianou, "Rhythmic similarity of music based on dynamic periodicity warping," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 2217–2220.

[23] Simon Dixon, Elias Pampalk, and Gerhard Widmer, "Classification of dance music by periodicity patterns," in *ISMIR*, 2003.

[24] Jonathan Foote and Shingo Uchihashi, "The beat spectrum: A new approach to rhythm analysis," in *ICME*, 2001.

[25] Kristopher West and Stephen Cox, "Features and classifiers for the automatic classification of musical audio signals," in *ISMIR*, 2004.

[26] Elias Pampalk, Arthur Flexer, Gerhard Widmer, et al., "Improvements of audio-based music similarity and genre classification," in *ISMIR*. London, UK, 2005, vol. 5, pp. 634–637.

[27] Geoffroy Peeters and Helene Papadopoulos, "Simultaneous beat and downbeat-tracking using a probabilistic framework: theory and large-scale evaluation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 6, pp. 1754–1769, 2011.

[28] Bob L Sturm, Corey Kereliuk, and Aggelos Pikrakis, "A closer look at deep learning neural networks with low-level spectral periodicity features," in *Cognitive Information Processing (CIP), 2014 4th International Workshop on*. IEEE, 2014, pp. 1–6.

[29] George Tzanetakis, Georg Essl, and Perry Cook, "Human perception and computer extraction of musical beat strength," in *Proc. DAFx*, 2002, vol. 2.

[30] Bob L Sturm, "An analysis of the gtzan music genre dataset," in *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies*. ACM, 2012, pp. 7–12.

[31] Bob L Sturm, "The gtzan dataset: Its contents, its faults, their affects on evaluation, and its future use," *arXiv preprint arXiv:1306.1461*, 2013.

[32] B Sturm, "A simple method to determine if a music information retrieval system is a "horse"," *IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 16, NO. 6, OCTOBER 2014*, 2014.

[33] H Christopher Longuet-Higgins and Christopher S Lee, "The perception of musical rhythms," *Perception*, vol. 11, no. 2, pp. 115–128, 1982.

[34] Richard Parncutt, "A perceptual model of pulse salience and metrical accent in musical rhythms," *Music Perception*, pp. 409–464, 1994.

[35] David Temperley, *The cognition of basic musical structures*, MIT press, 2004.

[36] Justin London, *Hearing in time*, Oxford University Press, 2012.

[37] Stephen Hainsworth and Malcolm Macleod, "Onset detection in musical audio signals," in *ICMC*, 2003.

[38] Juan P. Bello, Chris Duxbury, Mike Davies, and Mark Sandler, "On the use of phase and energy for musical onset detection in the complex domain," *IEEE Signal Processing Letters*, vol. 11, no. 6, pp. 553–556, 2004.

[39] Axel Roebel, "Onset detection in polyphonic signals by means of transient peak classification," in *ISMIR*, 2005.

[40] Anssi Klapuri, "Sound onset detection by applying psychoacoustic knowledge," in *Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing, 1999.* IEEE, 1999, vol. 6, IEEE.

[41] Chris Duxbury, Mark Sandler, and Mike Davies, "A hybrid approach to musical note onset detection," in *Proc. Digital Audio Effects Conf.(DAFX,i02)*, 2002, pp. 33–38.

[42] Chris Duxbury, Juan Pablo Bello, Mike Davies, Mark Sandler, et al., "Complex domain onset detection for musical signals," in *Proc. Digital Audio Effects Workshop (DAFx)*, 2003, pp. 6–9.

[43] Juan P. Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B Sandler, "A tutorial on onset detection in music signals," *IEEE Trans. on Speech and Audio Processing*, vol. 13, no. 5, pp. 1035–1047, 2005.

[44] Alexander Lerch, *An introduction to audio content analysis: Applications in signal processing and music informatics*, John Wiley & Sons, 2012.

[45] Vladimir Vapnik, *The nature of statistical learning theory*, Springer, 2000.

[46] Adam Berenzweig, Beth Logan, Daniel PW Ellis, and Brian Whitman, "A large-scale evaluation of acoustic and subjective music-similarity measures," *Computer Music Journal*, vol. 28, no. 2, pp. 63–76, 2004.

[47] Klaus Seyerlehner, Gerhard Widmer, and Dominik Schnitzer, "From rhythm patterns to perceived tempo," in *ISMIR*, 2007.

[48] Helge Homburg, Ingo Mierswa, Bülent Möller, Katharina Morik, and Michael Wurst, "A benchmark dataset for audio classification and clustering," in *ISMIR*, 2005.

[49] Isabelle Guyon and André Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[50] Hanchuan Peng, Fulmi Long, and Chris Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2005.

[51] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján, "Conditional likelihood maximisation: a unifying framework for information theoretic feature selection," *The Journal of Machine Learning Research*, vol. 13, pp. 27–66, 2012.

[52] Ron Kohavi and George H John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997.

[53] Florian Krebs, Sebastian Böck, and Gerhard Widmer, "Rhythmic pattern modeling for beat and downbeat tracking in musical audio.," in *ISMIR*, 2013.

[54] Giles M Foody, "Thematic map comparison," *Photogrammetric Engineering & Remote Sensing*, vol. 70, no. 5, pp. 627–633, 2004.

[55] Klaus Seyerlehner, Markus Schedl, Tim Pohle, and Peter Knees, "Using block-level features for genre classification, tag classification and music similarity estimation," *MIREX 2010*, 2010.

[56] Yannis Panagakis, Constantine Kotropoulos, and Gonzalo R Arce, "Music genre classification using locality preserving non-negative tensor factorization and sparse representations.," in *ISMIR*, 2009.

[57] Yannis Panagakis, Constantine L Kotropoulos, and Gonzalo R Arce, "Music genre classification via joint sparse low-rank representation of audio features," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 12, pp. 1905–1917, 2014.

[58] Athanasios Lykartsis, Chih-Wei Wu, and Alexander Lerch, "Beat histogram features from nmf-based novelty functions for music classification," in *ISMIR*, 2015.

[59] Athanasios Lykartsis and Stefan Weinzierl, "Using the beat histogram for speech rhythm description and language identification," in *INTERSPEECH*, 2015.

AN EVALUATION OF AUDIO FEATURE EXTRACTION TOOLBOXES

David Moffat, David Ronan, Joshua D. Reiss

Center for Digital Music
Queen Mary University of London
Mile End Road
London, E1 4NS
{d.j.moffat, d.m.ronan, joshua.reiss}@qmul.ac.uk

ABSTRACT

Audio feature extraction underpins a massive proportion of audio processing, music information retrieval, audio effect design and audio synthesis. Design, analysis, synthesis and evaluation often rely on audio features, but there are a large and diverse range of feature extraction tools presented to the community. An evaluation of existing audio feature extraction libraries was undertaken. Ten libraries and toolboxes were evaluated with the Cranfield Model for evaluation of information retrieval systems, reviewing the coverage, effort, presentation and time lag of a system. Comparisons are undertaken of these tools and example use cases are presented as to when toolboxes are most suitable. This paper allows a software engineer or researcher to quickly and easily select a suitable audio feature extraction toolbox.

1. INTRODUCTION

Audio feature extraction is one of the cornerstones of current audio signal processing research and development.

Audio features are contextual information that can be extracted from an audio signal. Features can be broken down into groups, as presented in the Cuidado Project [1], which includes definitions of a range of features.

Audio features can be applied to a range of research fields including:

- Feature extraction linked to audio effects [2]
- Statistical synthesis [3]
- Feature-based synthesis [4]
- Evaluating synthesis techniques [5]
- Similarity measures [6]
- Data classification [7]
- Data mining [8]

Although these problems are somewhat dissimilar in nature, they lean heavily on a set of related audio features. Low level features are computed directly from the audio signal, often in a frame-by-frame basis' such as zero-crossing rate, spectral centroid or signal energy, and generally have little perceptual relevance in comparison to higher level features, like chord or key of musical piece, which hold greater semantic meaning. In MIR, it is common to refer to audio features or descriptors, whereas, in psychology distinctions are made between dimensions and features, where dimensions are continuous and features are a discrete. Descriptor is often used as a general term since it can refers to either continuous or discrete content.

Seventeen low level descriptors (LLDs) are defined in the MPEG-7 standard, with feature categorisation, for the purpose of performing audio similarity searching with metadata contained within an MPEG file [9, 10], and the Cuidado project takes this work further to define 54 audio features [1]. This project provides definitions of a range of features, grouping them and which are relevant as frame based features. These audio features can then processed with the aim of identifying some particular aspect of an audio signal. A good overview of features for extraction is presented in [11]

Consequently, a range of audio feature extraction libraries and toolboxes have been constructed. Some are built as workflow tools, with pre-processing and batch operations, some are written for algorithmic efficiency or parallelisation, some for specific programming environments or platforms. Despite significant growth and research in the field of audio signal processing and feature extraction, there has been little research on evaluating and identifying suitable feature extraction tools and their appropriate applications.

It has been identified that within music information retrieval (MIR) primarily focuses on precision and recall, which may be considered a limitation [12, 13]. Cleverdon et. al. developed a six point scale for measuring and evaluating information retrieval systems. This model is widely known as the Cranfield model of information retrieval evaluation [14]. The Cranfield model properties are: Coverage; Time Lag; Effort; Presentation; Precision; Recall. This model is an appropriate platform for evaluation and benchmarking of MIR systems.

This paper reviews and evaluates existing feature extraction libraries based on the Cranfield model. The properties of the model can be suitably related to the MIR feature extraction tool evaluation [15] and presents an evaluation based on the following criteria:

Coverage - The range of audio descriptor features presented by a toolkit, along with additional preprocessing or post processing functionality.

Effort - User Interface, how easily one can create a new specific query or modify queries, and appropriate documentation.

Presentation - File Output format options and consistency.

Time Lag - Computational Efficiency of each tool.

Precision and recall are both included as part of the Cranfield Model. However, within the case of evaluating feature extraction toolboxes, precision and recall are not considered applicable to the task, and as such are not used. Existing work discusses the merits of using precision and recall within MIR application [16].

This paper presents ten audio feature extraction toolboxes, evaluated based on the Cranfield model, as proposed in [12]. Section 3 compares the functionality of the tools with respect to the range

of audio features that can be extracted and any further pre or post processing that the tool implements. The interface options of each toolboxes is presented and discussed in Section 4. The output format of data of each toolbox is presented in Section 5 and the computational time is presented in Section 6.

2. EXISTING FEATURE EXTRACTION TOOLBOXES

There are a large number of audio feature extraction toolboxes available, delivered to the community in differing formats, but usually as at least one of the following formats:

- stand alone applications
- plug-ins for a host application
- software function library

To allow for delivery of tools, some APIs have been constructed to allow for feature extraction plug-ins to be developed. Vamp [17] is a C++ API specification which functions with the standalone applications such as Sonic Visualiser, a content and feature visualiser with Graphical User Interface (GUI) and its command line interface (CLI) counterpart Sonic Annotator [18]. The Vamp Plugin API is an independent plugin development framework and as a result plugin libraries have been developed by numerous research labs and academic institutions. However due to the nature of the framework, it is not possible to create plug-ins that depend on pre-existing plug-ins. This results in multiple implementations and instances of certain features being calculated, which causes potential system inefficiencies. Feature Extraction API (FEAPI) is another plugin framework API in C and C++ [19], though it less commonly used than the VAMP plugin format. There are also feature extraction libraries that provide their own plugin API for extending their stand alone system [20], though this is less common. There has been a rise in MIR web services, such as the web based audio feature extraction API produced by Echo Nest[1], where users submit files online and receive XML descriptions. These tools have resulted in large music feature datasets, such as the Million Song Dataset [21].

The feature extraction tools that are evaluated in this paper are:

Aubio A high level feature extraction library that extracts features such as onset detection, beat tracking, tempo, melody [22].

Essentia Full function workflow environment for high and low level features, facilitating audio input, preprocessing and statistical analysis of output. Written in C++, with Python binding and export data in YAML or JSON format. [23].

jAudio Java based stand alone application with Graphic User Interface (GUI) and CLI. Designed for batch processing to output in XML format or ARFF for loading into Weka [20].

Librosa API for feature extraction, for processing data in Python [24]

LibXtract Low level feature extraction tool written with the aim of efficient realtime feature extraction, originally in C but now ported to Max-MSP, Pure Data, Super Collider and Vamp formats [25].

Marsyas Full real time audio processing standalone framework for dataflow audio processing with GUI and CLI. This programme includes a low level feature extraction tool built in C++, with ability to perform machine learning and synthesis within the framework.The feature extraction aspects have also been translated to Vamp plugin format [26].

[1] http://developer.echonest.com/

Meyda Web Audio API based low level feature extraction tool, written in Javascript. Designed for web browser based efficient real time processing [27].

MIR Toolbox Audio processing API for offline extraction of high and low level audio features in Matlab. Includes preprocessing, classification and clustering functionality along with audio similarity and distance metrics as part of the toolbox functionality. Algorithms are fragmented allowing detailed control with simple syntax, but often suffers from standard Matlab memory management limitations [28].

Timbre Toolbox A Matlab toolbox for offline high and low level feature extraction. A toolbox that provides different set of features to the MIR Toolbox, specifically made efficient for identifying timbre and to fulfil the Cuidado standards [29].

YAAFE Low level feature extraction library designed for computational efficiency and batch processing by utilising data flow graphs, written in C++ with a CLI and bindings for Python and Matlab [30].

This list is not exhaustive, as there are many other feature extraction tools out there [31, 32, 33, 34]. However the list of tools was designed with popularity, programming environment range and how recently it has been updated all being taken into consideration.

3. COVERAGE

The coverage of an information retrieval system can be defined as the extent to which all relevant matters are covered by the system. Within the context of audio feature extraction tools, the coverage can be considered as the range of features a tool can extract. This section presents the features provided by each toolbox, relative to the total number of unique features from all presented tool boxes and the features from the MPEG-7 and Cuidado standard sets of audio descriptors. The relative importance of audio features is heavily context based. To provide a meaningful measure of the relative importance of audio features within each toolbox, the toolboxes will be compared to their compliance with the MPEG-7 and Cuidado standards. Additional functionality, including preprocessing and post processing available with each feature extraction tool will also be discussed. The accuracy of audio features or specific implementation detail is beyond the scope of this paper, but is discussed in [35].

The features available within each tool is evaluated, and a list of unique features is created. Each tool is then compared to the total list of unique features. Each tool is also evaluated based on the feature coverage when compared to the MPEG-7 and Cuidado standard feature sets. The results of this can be seen in Figure 1. It can be seen that Essentia provides the largest range of features, and is the only toolbox to produce 100% coverage of the MPEG-7 audio descriptors. Following this the MIR Toolbox and LibXtract both fulfill over 85% of the MPEG-7 and provide 85% and 75% of the features contained within the Cuidado project, respectfully. The Timbre Toolbox provides nearly 75% of the Cuidado feature set, however this may be unsurprising as they were both written by the same principal author. YAAFE, jAudio and Librosa provide fairly similar features sets, presenting between 30% and 38% of the MPEG-7 standard feature set, with YAAFE presenting some more perceptually motivated features than the others. Meyda and Aubio provide a relatively low number of features, however this is

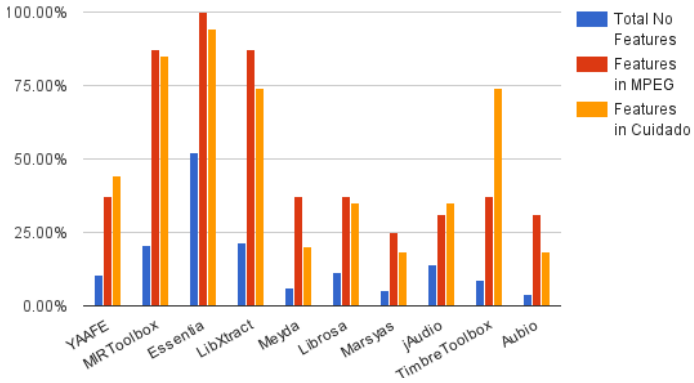


Figure 1: Graph of Percentage Coverage of Multiple Feature Sets

not without justification: Meyda is written for real time processing in the browser, and as such is inherently limited to a set of frame based features; Aubio is designed to focus on more high level feature extraction tools, and though providing a number of low level features, this is only to facilitate the high level feature extraction. Marsyas performs the worse in terms of feature range, complying to just 25% of MPEG-7 standard and 20% of the Cuidado standard. Marsyas is designed as an audio processing workflow, where clustering, classification and synthesis can all be performed within the entire workflow, so the range of available features may be limited, but the tool provides functionality beyond feature extraction. It is worth noting that only three features, spectral centroid, spectral rolloff and signal energy, are present in all the toolboxes and just 30 features are present in more than half of the toolboxes, and so attention must be paid if specific features are required.

Table 1 shows that LibXtract and Meyda do not provide any high level features. jAudio provides a limited range of high level features, such as beat histogram, and strongest beat. Aubio provides a limited range of low level features, such as spectral centroid, spread, skew and kurtosis. Aubio is designed specifically as a high level feature extraction tool, with particular focus on segmentation, whereas LibXtract, Meyda and jAudio are principally designed to extract low level features.

Additional functionality, such as preprocessing or post processing, is provided by a range of tools. Pre processing is an important aspect of evaluating any audio processing system, as it allows the user to be confident of a classification in any low quality environment where the audio may be degraded [36]. The resample function allows a standardisation of sample rates within a toolbox, which can be used to ensure that expected results for spectral comparisons are within the same range. For example, if a sample rate of 96kHz is used, it would be possible to have a spectral centroid of 30kHz, which has no perceptual meaning, compared to a file sampled at 44.1kHz, where a 30kHz spectral centroid would be impossible. As such standardisation of sample rates is an important factor that all feature extraction environments can provide. The quality of the resample method is an important attribute to consider [37], but is beyond the scope of this paper.

It can be seen from Table 1 that Aubio, Essentia, jAudio, Librosa, Marsyas and YAAFE all provide the user with some resample function as part of the toolbox, where as Meyda, MIR Toolbox and Timbre Toolbox all inherit a resample function from their native environments, as Web Audio API and Matlab both have resample functions built in. LibXtract does not provide a resample function, however, if used as a Vamp plugin, many Vamp hosts do contain resample functions.

Clustering, as a post processing tool, is also a useful functionality for many MIR processes. The post processing tools allow the user to directly analyse the output results as part of a single process. Essentia, Marsyas and MIR Toolbox all provide some from of clustering algorithm within them, and jAudio, Marsyas and MIR Toolbox can export files directly to ARFF format for loading directly into Weka, a data mining and clustering tool [38].

Essentia, MIR Toolbox and LibXtract produce a strong range of feature coverage, and the Timbre Toolbox covers the Cuidado feature set well. In terms of feature range these tools seem to perform better than many other existing tools. Essentia and MIR Toolbox both provide a powerful range of additional pre and post processing tools to benefit the user.

4. EFFORT

Effort is used to define how challenging a user finds a system to use, and whether any user experience considerations have been made while developing a system. Within this section, effort is evaluated relative to the user interface that is provided, whether it is a Graphical User Interface (GUI), Command Line Interface (CLI) or an Application Program Interface (API). The existence and quality of documentation and suitable examples is evaluated. The purpose is to identify how intuitively a tool's interface is presented to a user.

Table 1 outlines the user interfaces presented by each of the feature extraction tools. It can be seen that jAudio is the only tool that comes with its own GUI, though Aubio, LibXtract and Marsyas all have GUI capabilities through virtue of being Vamp plug-ins, when paired with a visualisation tool such as Sonic Visualiser. CLI's are more common, as Aubio, Marsyas, jAudio and YAAFE come with complete command line interfaces and Essentia comes with a series of precompiled C++ examples that can be run from the command line. However, this limits control functionality, as all the control is included in the software implementation. LibXtract can also be controlled via command line, through the use of its Vamp plugin format and a Vamp CLI tool such as Sonic Annotator. All tools come with APIs, which means Librosa, Meyda, MIR Toolbox and Timbre Toolbox are all only presented as software APIs, and as such all require software implementation before feature extraction is possible.

There are five different APIs written for both C and Python. Four APIs are available for Matlab, including the Essentia Matlab project[2]. Java has three APIs and only a single API for Javascript, Pure Data, Max-MSP, Supercollider, Lua and R are provided. Although Python and C are common programming languages, it is believed that Matlab is one of the most common frameworks used within MIR [39]. Environments such as web audio, in Javascript, Pure Data and Max-MSP are much less common in the MIR and audio research field, but are advantageous as they are real time audio environments where features are calculated in realtime and as such are excellent for prototyping.

Most toolboxes have clear documentation with examples, but there is limited documentation for LibXtract, Meyda and the Timbre toolbox. Though the documentation is not unclear, all the other tools provide a lot more information regarding basic access and software applications. Similarly, all toolboxes supply basic ex-

Table 1: Overview of Feature Extraction Tools

	Aubio	Essentia	jAudio	Librosa	LibXtract	Marsyas	Meyda	MIR	Timbre	YAAFE
High level Features	Y	Y	N[‡]	Y	N	Y	N	Y	Y	Y
Low Level Features	N*	Y	Y	Y	Y	Y	Y	Y	Y	Y
Resample	Y	Y	Y	Y	N	Y	Y[†]	Y[†]	Y[†]	Y
Filter	N	Y	N	N	N	N	Y[†]	Y[†]	Y[†]	N
Clustering	N	Y	N[§]	N	N	Y[§]	N	Y[§]	N	N
Similarity	N	N	N	N	N	N	N	Y	N	N
Real Time	Y				Y	Y	Y			
Vamp Plugin	Y	N	N	N	Y	Y	N	N	N	N
GUI	Y[+]	N	Y	N	Y[+]	Y[+]	N	N	N	N
CLI	Y	Y[E]	Y	N	Y[+]	Y	N	N	N	Y
APIs	C/C++ Python R PD	C/C++ Python Matlab[O] PD/Max-MSP	Java	Python	C/C++ Supercollider PD/Max-MSP Java	C/C++ Python Java Lua	JS	Matlab	Matlab	Matlab Python C/C++
Output	Vamp	YAML JSON	XML ARFF	CSV	Vamp XML	Vamp CSV ARFF		TSV ARFF	TSV	CSV HDF5

* = Except MFCC and FFT Statistics,
[‡] = Some Mid-high level features but very limited,
[†] = As part of environment, not toolbox,
[+] = As result of being Vamp plugin,
[§] = Can produce ARFF files, designed for being read directly into Weka.
[E] = CLI is produced through C 'Extractor' files, with some examples provided.
[O] = A project for calling Essentia from Matlab has been developed.

amples of implementation, however YAAFE, Essentia, Aubio and MIR Toolbox all have a strong range of examples that run straight away. Marsyas has clear documentation and a range of examples from which to draw inspiration, but required the user to learn a proprietary language for use as part of the system.

In conclusion, when looking for a stand alone tool which covers a user flexibility and usability with a user interface, then the Vamp plugin route is a useful one to take. This provides a simple intuitive interface and any number of specific features can be loaded in as required. If batch processing is required, then either the GUI from jAudio or CLI from YAAFE are intuitive, flexible and simple to use. If a user requires a programming API, then, depending on their environment, there are potentially a range of tools. C, C++ Python and Matlab APIs are provided by a range of tools, with often multiple being offered by each toolkit, as can be seen in Table 1.

5. PRESENTATION

An important aspect of any information retrieval system is how the resulting information is presented back to the user. Within this section, the output format of data is discussed and the relative merits of each approach outlined.

Document output format is one of the most significant barriers in fully integrated workflow solutions within MIR [40]. Output format is important, primarily as it impacts the ease and format of analysis someone can carry out on a dataset. Usually, a user using a software API, stores values in a relevant data structure within the given development language and as such, file output format becomes irrelevant in this case.

XML, YAML and JSON are all standard structured data formats, that allow the presentation of hierarchical structures of data. HDF5 is also a hierarchical data structure specifically designed for efficient storage and accuracy of contents - as such is well suited to big data tasks. CSV and TSV are table structures that allow users to view data in most spreadsheet applications, and ARFF is also a table structure with specific metadata about each column format and available options. ARFF is specifically designed for use with Weka, which is a powerful data mining tool.

CSV and TSV formats are considered to be suitable output formats if the resulting value can be considered as a table, however within the feature extraction, generally there is a much more complex data structure than simply two dimensions. Features carry varying levels of complexity, as some features are global for a signal where as some are based on windowed frames of a signal. Some features, such as MFCC's produce 13 numerical values per frame. As such it seems suitable that the data format used to output these results can represent these hierarchical feature formats. JSON and XML file formats are well supported by almost all programming languages, so there should not be any issues with processing the data results. CSV is also well supported within programming languages, but the lack of complexity or data structure can lead to potential ambiguities or errors with data transfer. The benefits of producing ARFF files, which can be loaded direct into Weka allows the user a great range of data mining opportunities and should not be underestimated.

Although it is clear that the file format is reliant on further applications, any feature extraction library should be able to present its data output in a data structure to suitable represent the hierarchical nature of the data it intends to represent. YAAFE, Essentia, jAudio and LibXtract all provide some from of suitable data structure, however only jAudio can also pass files direct into Weka [38].

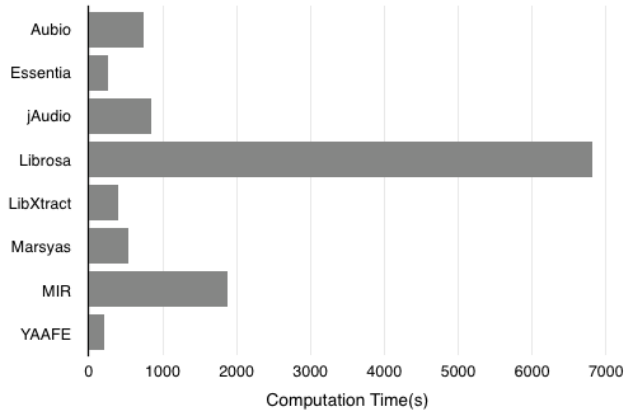


Figure 2: Graph of Computational Time of Feature Extraction Tools

Marsays and MIR Toolbox both allow for unstructured data, but can produce ARFF files for easy data mining, and Librosa and Timbre Toolbox will only allow users an unstructured data in a table format. YAAFE and jAudio are the only two applications that allow users the choice of structured or tabular data.

6. TIME LAG

Time lag is the measure of how long a given task will take to complete. Understanding the time necessary to perform a task, and comparing the relative speed of systems will give users an informed choice as to what system to use, particularly when they want to analyse large data sets. This section will discuss the computational complexity of the ten feature extractions tools and identify whether they are implemented in real time or if they are offline methods. There is existing work on implementing existing feature extraction tools in a distributed manner [41], but this is beyond the scope of this paper.

Meyda and the various LibXtract ports to Pure Data, Supercollider and Max-MSP are all designed to run in realtime. Each of these real time environments are provided with suitable feature extraction, which provides a user with powerful visualisation and real time interactivity but is less useful for users wishing to focus on offline approaches.

The offline approaches were all evaluated for computational efficiency. A dataset for evaluation is a subset of the Cambridge Multitrack Data Set[3]. 32 Different songs were used. The dataset consists of 561 tracks with an average duration of 106s is used, which totalled over 16.5hours of audio and 8.79Gb of data. Each toolbox is used to calculate the MFCC's from this data set, with a 512 sample window size and 256 sample hop size. The input audio is at a variety of different sample rates and bit depths to ensure that variable input file formats is allowable. This test is run on a MacBook Pro 2.9GHz i7 processor and 8Gb of RAM. The results are presented in Figure 2. The MFCCs were used, as they are a computational method, that exists within nine of the ten given tool boxes, and so should provide a good basis for comparison of computational efficiency. MFCCs are not computed by the Timbre Toolbox and Meyda will only run in real-time.

As can be seen from Figure 2, Yaafe is the fastest toolbox, processing over 16.5 hours of audio in just over 3 minutes 30s, with Essentia coming in as a close second place at 4 minutes 12s. LibXtract and Marsyas both completed in under 10 minutes, and both Aubio and jAudio ran in under 15 minutes. The MIR toolbox took over 31 minutes to run and Librosa took 1hour 53 minutes. It is evident that tools written in C or C++ run faster than tools written in Python or java.

[3]http://www.cambridge-mt.com/ms-mtk.htm

7. CONCLUSION

Ten audio feature extraction toolboxes are discussed and evaluated relative to four of the six criteria of the Cranfield Model.

Meyda and LibXtract provide excellent real time feature extraction tools in various programming environments. When high level features and segmentation is required, Aubio provides a simple and intuitive tool. It also provides a Vamp plugin format. When visualisation is required, for annotation or basic exploration of feature, using the Vamp plugin format is very powerful, and the combination of LibXtract and Marsysas as Vamp plug-ins provide excellent coverage of audio features. Research based in MATLAB should use the MIR Toolbox combined with the Timbre Toolbox for maximum feature coverage, or Essentia where computational efficiency is important with little sacrifice of feature range. Essentia performs the best with regards to computation, feature coverage and output presentation, with a range of APIs.

As the suitable feature extraction toolbox is entirely application dependent, there is no single case where a certain tool is better or more powerful than another. However suggestions for suitable toolboxes to use can be identified from Figure 3. When working on real time applications, either Meyda or LibXtract will be most suitable for applications, where as when working in an offline fashion, there is an option for either user interfaces or APIs. If a user interface is required then Vamp plug-ins, of LibXtract and Marsyas, are very powerful and advantageous tools, that can be hosted in either graphic interfaces or command line interfaces. jAudio provides a strong user interface with batch processing tool, but its range of features is limited. Essentia provides a strong CLI with large range of features, but low level of control, so implementation is required for accurate control of the features. If an API is required, then a range of example suggestions are proposed for some commonly used programming languages. A Java API is provided by jAudio, which is powerful and efficient, but performs on a reduced feature set. Strong Matlab APIs are provided by either a combination of MIR Toolbox and Timbre Toolbox or Essentia, with the 'Running essentia in matlab'.

8. ACKNOWLEDGMENTS

Funding for this research was provided by the Engineering and Physical Sciences Research Council (EPSRC).

References

[1] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project," Tech. Rep., IRCAM, 2004.

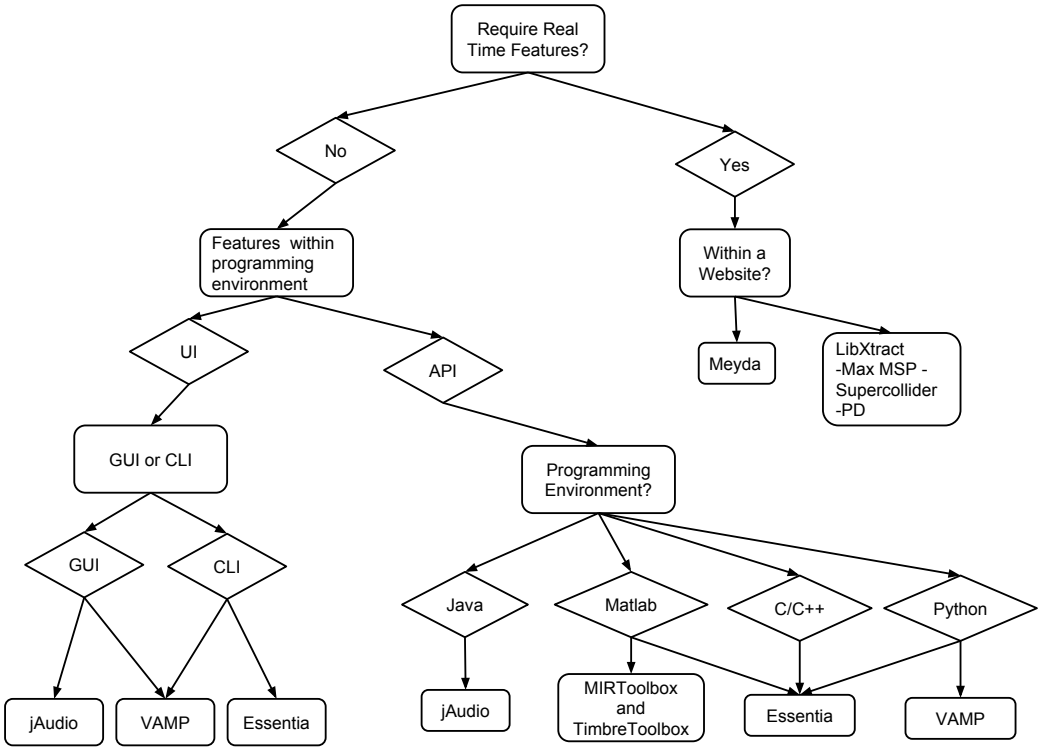


Figure 3: Flowchart to recommend what tool to use
N.B. Vamp = LibXtract and Marsyas Vamp Packages

[2] R. Stables, S. Enderby, B. De Man, G. Fazekas, and J. D. Reiss, "SAFE: A system for the extraction and retrieval of semantic audio descriptors," in *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, October 2014.

[3] J. H. McDermott and E. P. Simoncelli, "Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis," *Neuron*, vol. 71, no. 5, pp. 926–940, 2011.

[4] M. D. Hoffman and P. R. Cook, "Feature-based synthesis: A tool for evaluating, designing, and interacting with music ir systems.," in *ISMIR*, 2006, pp. 361–362.

[5] S. Hendry and J. D. Reiss, "Physical modeling and synthesis of motor noise for replication of a sound effects library," in *Audio Engineering Society Convention 129*. Audio Engineering Society, 2010.

[6] B. Gygi, G. R. Kidd, and C. S. Watson, "Similarity and categorization of environmental sounds," *Perception & psychophysics*, vol. 69, no. 6, pp. 839–855, 2007.

[7] M. F. McKinney and J. Breebaart, "Features for audio and music classification.," in *ISMIR*, 2003, vol. 3, pp. 151–158.

[8] T. Li, M. Ogihara, and G. Tzanetakis, *Music data mining*, CRC Press, 2011.

[9] B. S. Manjunath, P. Salembier, and T. Sikora, *Introduction to MPEG-7: multimedia content description interface*, vol. 1, John Wiley & Sons, 2002.

[10] A. T. Lindsay and J. Herre, "MPEG-7 and MPEG-7 audio – an overview," *Journal of the Audio Engineering Society*, vol. 49, no. 7/8, pp. 589–594, 2001.

[11] D. Mitrović, M. Zeppelzauer, and C. Breiteneder, "Features for content-based audio retrieval," *Advances in computers*, vol. 78, pp. 71–150, 2010.

[12] J. D. Reiss and M. Sandler, "Beyond recall and precision: A full framework for MIR system evaluation," in *3rd Annual International Symposium on Music Information Retrieval, Paris, France*, 2002.

[13] J. D. Reiss and M. Sandler, "MIR benchmarking: Lessons learned from the multimedia community," *The MIR/MDL Evaluation Project White Paper Collection*, vol. 3, pp. 114–120, 2003.

[14] C. W. Cleverdon and M. Keen, "Aslib cranfield research project-factors determining the performance of indexing systems; volume 2, test results," Tech. Rep., Cranfield University, 1966.

[15] J. D. Reiss and M. Sandler, "Benchmarking music information retrieval systems," *JCDL Workshop on the Creation of Standardized Test Collections, Tasks, and Metrics for Music Information Retreival (MIR) and Music Digital Library (MDL) Evaluation*, pp. 37–42, July 2002.

[16] J. S. Downie, "The scientific evaluation of music information retrieval systems: Foundations and future," *Computer Music Journal*, vol. 28, no. 2, pp. 12–23, 2004.

[17] C. Cannam, "The vamp audio analysis plugin api: A programmer's guide," *Availble online: http://vamp-plugins. org/guide. pdf*, 2009.

[18] C. Cannam, C. Landone, and M. Sandler, "Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files," in *Proceedings of the ACM Multimedia 2010 International Conference*, Firenze, Italy, October 2010, pp. 1467–1468.

[19] A. Lerch, G. Eisenberg, and K. Tanghe, "FEAPI: A low level feature extraction plugin api," in *Proceedings of the International Conference on Digital Audio Effects*, 2005.

[20] C. McKay, I. Fujinaga, and P. Depalle, "jAudio: A feature extraction library," in *Proceedings of the International Conference on Music Information Retrieval*, 2005, pp. 600–3.

[21] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida*. University of Miami, 2011, pp. 591–596.

[22] P. M. Brossier, "The aubio library at MIREX 2006," *MIREX 2006*, p. 1, 2006.

[23] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra, "Essentia: An audio analysis library for music information retrieval.," in *ISMIR*, 2013, pp. 493–498.

[24] B. McFee, M. McVicar, C. Raffel, D. Liang, and D. Repetto, "librosa: v0.3.1," Nov. 2014.

[25] J. Bullock and U. Conservatoire, "Libxtract: A lightweight library for audio feature extraction," in *Proceedings of the International Computer Music Conference*, 2007, vol. 43.

[26] G. Tzanetakis and P. Cook, "Marsyas: A framework for audio analysis," *Organised sound*, vol. 4, no. 03, pp. 169–175, 2000.

[27] H. Rawlinson, N. Segal, and J. Fiala, "Meyda: an audio feature extraction library for the web audio api," in *Web Audio Conference*. Web Audio Conference, 2015.

[28] O. Lartillot and P. Toiviainen, "A matlab toolbox for musical feature extraction from audio," in *International Conference on Digital Audio Effects*, 2007, pp. 237–244.

[29] G. Peeters, B. L. Giordano, P. Susini, N. Misdariis, and S. McAdams, "The timbre toolbox: Extracting audio descriptors from musical signals," *The Journal of the Acoustical Society of America*, vol. 130, no. 5, pp. 2902–2916, 2011.

[30] B. Mathieu, S. Essid, T. Fillon, J. Prado, and G. Richard, "YAAFE, an easy to use and efficient audio feature extraction software.," in *ISMIR*, 2010, pp. 441–446.

[31] W. Brent, *A timbre analysis and classification toolkit for pure data*, Ann Arbor, MI: MPublishing, University of Michigan Library, 2010.

[32] F. Eyben, F. Weninger, F. Groß, and B. Schuller, "Recent developments in openSMILE, the munich open-source multimedia feature extractor," in *Proceedings of the 21st ACM international conference on Multimedia*. ACM, 2013, pp. 835–838.

[33] D. L. Bryant, "Scalable audio feature extraction," M.S. thesis, University of Colorado Colorado Springs, 2014.

[34] F. Deliege, B. Y. Chua, and T. B. Pedersen, "High-level audio features: Distributed extraction and similarity search," in *Ninth International Conference on Music Information Retrieval*, 2008, pp. 565–570.

[35] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. Ellis, "mir_eval: A Transparent Implementation of Common MIR Metrics," in *Proc. of the 15th International Society for Music Information Retrieval Conference, Taipei, Taiwan*, 2014.

[36] M. Mauch and S. Ewert, "The audio degradation toolbox and its application to robustness evaluation.," in *ISMIR*, 2013, pp. 83–88.

[37] A. Franck, "Performance evaluation of algorithms for arbitrary sample rate conversion," in *Audio Engineering Society Convention 131*, Oct 2011.

[38] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[39] K. R. Page, B. Fields, D. De Roure, T. Crawford, and J. S. Downie, "Reuse, remix, repeat: the workflows of MIR.," in *ISMIR*, 2012, pp. 409–414.

[40] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[41] S. Bray and G. Tzanetakis, "Distributed audio feature extraction for music.," in *ISMIR*, 2005, pp. 434–437.

DIGITALLY MOVING AN ELECTRIC GUITAR PICKUP

Zulfadhli Mohamad, Simon Dixon,*
Centre for Digital Music,
Queen Mary University of London
London, United Kingdom
z.b.mohamad@qmul.ac.uk
s.e.dixon@qmul.ac.uk

Christopher Harte,
Melodient Limited,
United Kingdom
chris@melodient.com

ABSTRACT

This paper describes a technique to transform the sound of an arbitrarily selected magnetic pickup into another pickup selection on the same electric guitar. This is a first step towards replicating an arbitrary electric guitar timbre in an audio recording using the signal from another guitar as input. We record 1458 individual notes from the pickups of a single guitar, varying the string, fret, plucking position, and dynamics of the tones in order to create a controlled dataset for training and testing our approach. Given an input signal and a target signal, a least squares estimator is used to obtain the coefficients of a finite impulse response (FIR) filter to match the desired magnetic pickup position. We use spectral difference to measure the error of the emulation, and test the effects of independent variables fret, dynamics, plucking position and repetition on the accuracy. A small reduction in accuracy was observed for different repetitions; moderate errors arose when the playing style (plucking position and dynamics) were varied; and there were large differences between output and target when the training and test data comprised different notes (fret positions). We explain results in terms of the acoustics of the vibrating strings.

1. INTRODUCTION

The electric guitar revolutionised Western popular music and was for several decades the most important instrument in most pop, rock and blues music. Although it may no longer hold such unique supremacy, the electric guitar remains an essential element of many of the styles it helped to define. Perhaps more so than with other instruments, many famous guitar players are recognisable by their distinctive electric guitar tone, and guitar enthusiasts are keen to know the "secrets" behind the unique sound of their favourite artist. In order to replicate the sound of their favourite guitar player, they often purchase the same model of guitar and other musical equipment used, and then adjust each of their parameters manually until similar tone is achieved. In recent years, digital replication of electric guitar, guitar amplifiers and guitar effects has grown rapidly in the research community and the music industry.

Several decades of literature exist that deal with synthesising the sound of plucked string instruments. Research can be divided into physical modelling, which involves solving the wave equation describing the vibrating string [1, 2], and more abstract models which attempt only to imitate the resulting sound, such as the Karplus-Strong model [3, 4]. These models have been extended to account for the characteristics of the electric guitar, based on waveguide theory [5, 6, 7].

The magnetic pickup contributes heavily to the timbre of an electric guitar. Thus, accurately modelling the magnetic pickup of an electric guitar is essential. In [5], the Karplus-Strong model is extended by introducing a pickup position model that emulates the comb-filtering effect of the position of the magnetic pickup along the string, whereby harmonics with nodes at or near the pickup position are suppressed. A parametric synthesis model of a Fender Stratocaster electric guitar is described in [6] that alters its level and timbre depending on the distance of the pickup and also includes the inharmonic behaviour of the pickup. A more detailed model for a magnetic pickup in [7] includes the width of the pickup, its nonlinearity and circuit response adding to the tonal colouration of the electric guitar.

As mentioned in [6], the playing technique of a musician also alters the timbre of an electric guitar. Plucking with fingers or a plectrum are the two common styles of exciting an electric guitar. The size, shape and material of the plucking device affect the tone of the guitar [8] and the plucking point along the string also contributes to the timbre. A plucking point close to the bridge will produce a brighter sound, while plucking near the fingerboard will produce a warmer sound [9, 10]. This is caused by the low amplitude of harmonics that have a node at or near the plucking point. Varying how strongly a string is plucked will also affect the level of higher harmonics [6].

Outside academia, commercial products including guitar synthesisers are available that are able to emulate the sound of most popular electric guitars on the market by modifying the sound of a standard guitar [11, 12]. Each string is detected individually by a hexaphonic pickup and processed by the magnetic pickup model of the selected electric guitar sound.

For all physical modelling systems, parameters of the copied electric guitar must be known accurately in order to model its sound. Many influential guitar players used electric guitars and amplifiers that are now considered vintage items; prohibitively expensive and difficult to obtain today. Certain instruments may have been discontinued by the manufacturer and modern examples may not produce a sufficiently similar sound to the older models. The lack of availability of such instruments makes it difficult to measure the physical properties of the guitar, thus making it challenging to model the instrument digitally. In our research, we are exploring the concept of replicating the electric guitar sound from an audio recording without having prior knowledge of the physical parameters of the desired electric guitar sound. In particular, in this study we analyse recordings of an electric guitar made using different pickup positions, and we compute filters to trans-

* Zulfadhli Mohamad is supported by the Malaysian government agency, Majlis Amanah Rakyat (MARA)

form the sound recorded by one pickup into the sound of another pickup. We test the effects of the following variables on the quality of learnt filters: plucking position, dynamics, fret position and random variation in human plucking. Although the problem of inverting a notch filter may appear ill-posed, we show that good results can be obtained in practice.

Section 2 describes the sound samples used in this study and the limitations of replicating a desired sound. Section 3 presents an overview of mathematical foundations for determining the optimal FIR coefficients to estimate the desired sound. Furthermore, Section 4 explains the calculation of the accuracy of the estimated signal. The transformation of the sound of one pickup into that of another pickup at a different position is explained and the optimum number of coefficients is described in Section 5. In Section 6, the robustness of the filters when applied to an input signal with different repetitions, plucking positions, plucking dynamics and fret positions is measured. Lastly, the conclusions are presented in Section 7.

2. AN ELECTRIC GUITAR SOUND ANALYSIS DATA SET

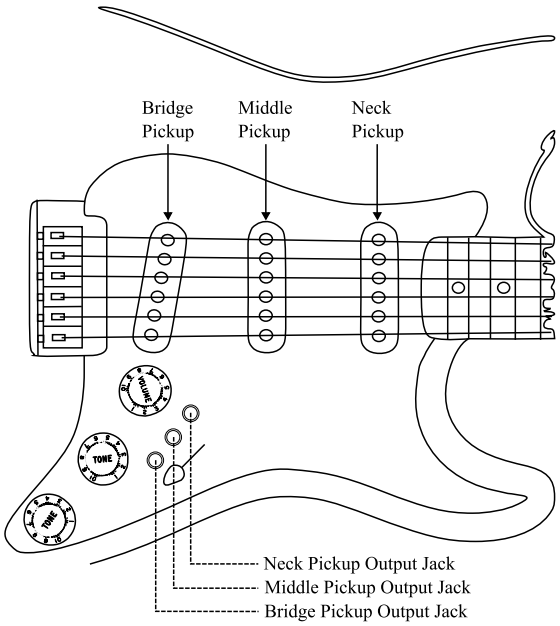


Figure 1: *The modified Squier Stratocaster diagram. Three 1/8" output jacks allow us to tap separate signals from each magnetic pickup simultaneously. The three plucking positions are directly above each pickup.*

The purpose of this research is to investigate whether it is possible to transform the sound produced by an arbitrary electric guitar to a desired guitar sound in an audio recording using optimisation techniques. The initial experiments in this study simplify the purpose by attempting to transform the sound of a pickup position into another on the same electric guitar. This is an important step because the magnetic pickup has a large impact on the timbre of an electric guitar.

Since the playing style of a guitarist affects the colouration of the electric guitar tone, it is preferable that the input signal is played at the same plucking point and plucking dynamic as the target signal. In other words, our aim is to account for differences due to the instrument or its settings (which we assume are fixed) from those due to playing technique. Thus we also assume that the timing and pitch of notes in the input and target signals coincide, and in particular that the input signal is played at the same fret position and on the same string as the target. This work utilises a modified guitar that allows us to tap the signals from each individual pickup simultaneously. This means that each signal will be played at exactly the same plucking point, dynamic, pitch and timing.

The electric guitar that is used in this study is a Squier Stratocaster with three stock magnetic pickups rewired so that each pickup can be simultaneously recorded (see Figure 1), in order to isolate differences due to the pickup from those due to time alignment or playing style. The pickup positions are situated at 158.75 mm (neck pickup), 101.6 mm (middle pickup) and 38.1 mm - 50.8 mm (slanted bridge pickup) from the bridge. The scale length of the guitar is 648 mm. The strings used are nickel wound strings with gauges .010, .013, .017, .026, .036 and .046.

The sound samples used in this paper consist of each string being plucked using a plectrum at three different fret positions, three plucking positions and three plucking dynamics, with each combination being repeated three times. The plucking dynamics are *forte* (loud), *mezzo-forte* (moderately loud) and *piano* (soft); the three different plucking positions are when the electric guitar is plucked near the neck (158.75 mm from the bridge), at a central playing position (101.6 mm from the bridge) and near the bridge (45 mm from the bridge); the fret positions are played at open string, fifth fret and twelfth fret; and lastly, all of the combinations are played on each string and repeated for three times. This leads to a total of $6 \times 3 \times 3 \times 3 \times 3 = 486$ different variations that are recorded from each of the three pickup positions. Thus, 1458 sound samples are available to be analysed. The duration of the audio samples ranges from 3 to 28 seconds depending on the decay rate for each strings. It is planned to make this dataset publicly available for research purposes.

3. OPTIMISATION TECHNIQUE

In this study, we aimed to transform the sound of any pickup selection into the sound of another pickup on the same guitar using an FIR filter. As an example, we are taking the sound of the neck pickup as an input and transforming its sound into the bridge pickup sound. This is achieved by convolving the input signal (neck pickup sound), $x(n)$ with an FIR filter, $h(n)$ to estimate the target signal (bridge pickup), $y(n)$.

$$y(n) = x(n) * h(n) \tag{1}$$

As the filter $h(n)$ is unknown, an optimisation technique is required to estimate the FIR coefficients accurately. In this paper, the coefficients of the FIR filter are obtained by using the least squares method. A least squares estimator has been used to estimate the coefficients of a filter to reverse engineer a target mix [13, 14]. The linear combination to estimate the desired response is given by:

$$\hat{y}(n) = \sum_{k=1}^{M} h(k)\, x(n-k) \tag{2}$$

where $h(k)$ is the filter coefficient vector for the FIR filter, M is the filter order and $\hat{y}(n)$ is the estimated target response. This can also be expressed in matrix notation as:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{h} \tag{3}$$

where the matrix $\mathbf{X}$ is composed of M shifted versions of $x(n)$. The estimation error and its matrix notation are given by:

$$e(n) = y(n) - \hat{y}(n) = \mathbf{y} - \mathbf{X}\mathbf{h} \tag{4}$$

and the set of optimal coefficients are computed by minimising the sum of squared errors:

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\mathbf{h}\| \tag{5}$$

By solving the least-squares normal equations the optimal coefficients are given by:

$$\hat{\mathbf{h}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{6}$$

where $\mathbf{X}^T\mathbf{X}$ is the time-average correlation matrix, $\hat{\mathbf{R}}$, the elements of which can be calculated by:

$$\begin{aligned} \hat{r}_{ij} &= \tilde{\mathbf{x}}_i^T\tilde{\mathbf{x}}_j \\ &= \textstyle\sum_{n=N_i}^{N_f} x(n+1-i)x^*(n+1-j) \quad 1 \le i,j \le M \end{aligned} \tag{7}$$

leading to:

$$\begin{aligned} \hat{r}_{i+1,j+1} &= \hat{r}_{ij} + x(N_i - i)x^*(N_i - j) \\ &\quad -x(N_f + 1 - i)x^*(N_f + 1 - j) \quad 1 \le i,j < M \end{aligned} \tag{8}$$

where N_i and N_f are the range of computing the process of the filtering operation. Here, we set $N_i = 0$ and $N_f = N-1$ which is the pre-windowing method that is extensively used in least squares adaptive filtering [15].

4. TIMBRAL SIMILARITY MEASUREMENT

Once the set of optimal coefficients for the FIR filter and estimated response are obtained, the similarity between the estimated signal and target signal is measured. A more meaningful way of measuring the similarity of both signals is by measuring the distance between two sounds in a time-frequency representation. The short-time Fourier transform (STFT) is commonly used for time-frequency analysis. There are several papers that propose a genetic optimisation approach to find optimal parameters for frequency modulation matching synthesis and a method of measuring the similarity between two sounds [16, 17, 18]. The waveforms that are being measured are divided into short segments and a discrete Fourier transform is calculated for each segment. We set the length of the frame to be 1024 samples with an overlap of 512 samples, where the sampling rate of the audio signals is 44100 Hz. The raw distance (or error), $D_R(\hat{Y}, Y)$ is calculated as follows:

$$D_R(\hat{Y}, Y) = \frac{1}{T}\sum_{t=1}^{T}\sum_{f=1}^{F} \left|\, \hat{Y}_t[f] - Y_t[f] \,\right|^2 \tag{9}$$

where $\hat{Y}_t[f]$ is the magnitude spectrum of the estimated response at time frame t and frequency bin f, $Y_t[f]$ is the magnitude spectrum of the target response at frame t and bin f, T is the number of frames in the STFT and F is the total number of frequency bins in each frame. Ideally, the sound is considered to be more similar to the target response when the distance is closer to zero. Due to the variations in plucking dynamics between different trials, a normalisation is required to compensate for differences in loudness. The raw distance, D_R, is divided by the average energy of the target signal and the estimated signal, giving the normalised distance $D(\hat{Y}, Y)$:

$$D(\hat{Y}, Y) = \frac{2D_R(\hat{Y}, Y)}{\sum_{k=0}^{K}|y(k)|^2 + \sum_{l=0}^{L}|\hat{y}(l)|^2} \tag{10}$$

where K is the sample length of the target signal and L is the sample length of the estimated signal.

5. RESULTS: AN EXAMPLE

5.1. Learning the Filter

Table 1: *Results for transforming one pickup position to another for a tone played on an open G string ($f_0 = 196$ Hz). The normalised distance between estimated signal $\hat{Y}$ and target signal Y is shown in the fourth column. For comparison, the normalised distance between input X and target signal is given in the third column showing a large reduction in distance after transformation.*

Input Signal (X)	Target Signal (Y)	$D(X, Y)$	$D(\hat{Y}, Y)$
neck pickup	bridge pickup	0.802	0.123
bridge pickup	neck pickup	0.802	0.011
neck pickup	mid pickup	0.434	0.085
mid pickup	neck pickup	0.434	0.007
bridge pickup	mid pickup	0.234	0.007
mid pickup	bridge pickup	0.234	0.009

In this section we demonstrate the use of an FIR filter to transform the sound from the neck pickup into the sound of the bridge pickup for the open G string (3rd string, $f_0 = 196$ Hz). The electric guitar is plucked directly above the middle pickup and played *forte*. The filter order is set to 1024, with coefficients obtained using the least squares method described in Section 3; the estimated signal produced by convolving the input signal and the filter impulse response.

Figure 2 shows the magnitude spectra of the neck pickup signal, bridge pickup signal and estimated bridge pickup signal for a tone played on the open G string. The spectral envelope curves drawn on Figures 2(a) and 2(b) illustrate the comb filtering effect due to the different pickup positions. The neck pickup, situated approximately $\frac{1}{4}$ of the way along the string, lowers the amplitude of every 4th harmonic, while the bridge pickup, at about $\frac{1}{14}$ of the string length, lowers the amplitude of every 14th harmonic. As shown in Figure 2(c), the amplitude of every 4th harmonic is increased and every 14th harmonic is decreased relative to the input (Figure 2(a)), which indicates that the magnitude spectrum of the estimated signal matches that of the target signal. The accuracy of the estimated signal is calculated as the normalised distance $D(\hat{Y}, Y)$ between target (Y) and estimated $(\hat{Y})$ signals using Equation 10. For this example, the distance is 0.123, compared

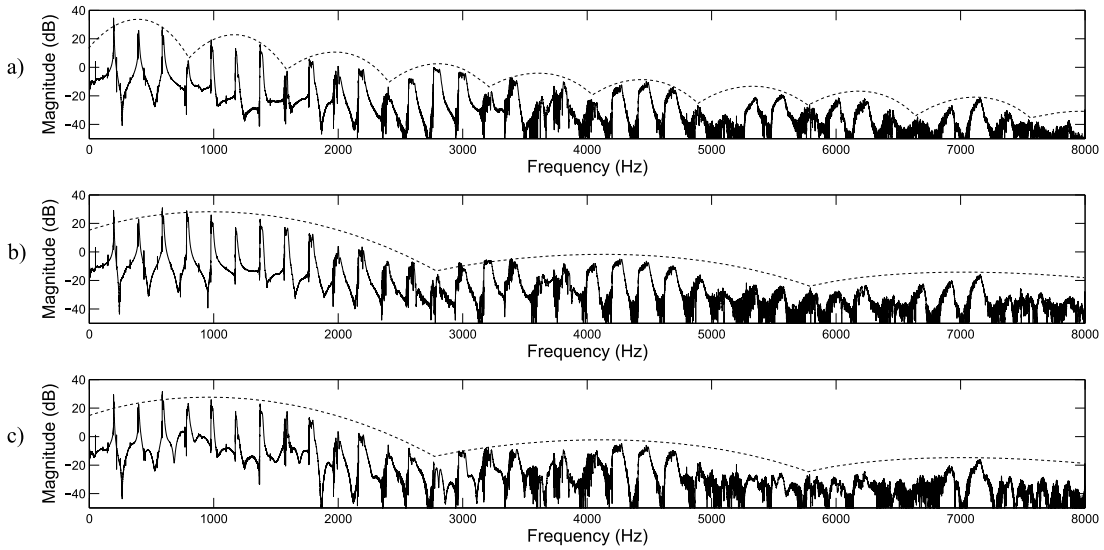


Figure 2: *Magnitude spectra for a guitar tone played on the open 3rd string ($f_0 = 196$ Hz), calculated from (a) the neck pickup signal, (b) the bridge pickup signal and (c) the estimated bridge pickup signal. The spectral envelopes are drawn for illustrative purposes to show the comb filtering effect of the pickup position.*

with the distance $D(X, Y) = 0.802$ between input (X) and target signals, a reduction of 85% in the spectral difference.

The filter coefficients can also be determined for other pairs of input and target signals. Table 1 shows the distances calculated for transforming between pickup positions for one tone. In all cases the filter is able to simulate the effect of moving the pickup position, reducing the spectral difference by 80% to 99% for the various cases. The transformation of the neck pickup sound appears to be more difficult than the other cases; we discuss reasons for this in subsection 5.3. In Section 6 we investigate the factors influencing the generalisation of these results.

5.2. Estimating the Filter Order

The accuracy of the estimated signal was computed for various filter orders, to estimate a suitable number of coefficients for the FIR filter. Figure 3 shows that the error converges for higher order filters. We choose 1024 FIR coefficients as a reasonable compromise of accuracy and efficiency to estimate the target signal.

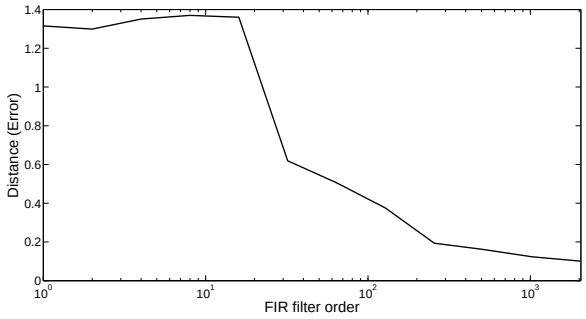


Figure 3: *Normalised spectral distance of the estimated signal from the target signal (Equation 10) as a function of filter order. Error starts to converge above order 1000 so we choose 1024 as our filter order for experiments here.*

5.3. Comparison with Theoretical Model

The filter that was obtained in Section 5.1 is analysed and compared with a theoretical model. Transforming the sound of a neck pickup sound into bridge pickup sound is achieved by cascading an inverse neck pickup model and bridge pickup model. The theoretical model, $H_t(z)$ is computed as follows:

$$H_t(z) = \frac{H_b(z)}{H_n(z)} \tag{11}$$

where $H_n(z)$ is the neck pickup model and $H_b(z)$ is the bridge pickup model. The inverse neck pickup model and bridge pickup model are essentially a feedback comb filter and feedforward comb filter, respectively, which include a fractional delay and a dispersive filter [6, 7, 19]. We excluded some of the details in building the theoretical model such as the nonlinearity of pickup [7, 20] and pickup response [7] which are beyond the scope of this paper. The simplified theoretical model captures the basic behaviour of the system.

Figures 4(a) and 4(b) show the frequency responses of the inverse neck pickup model and bridge pickup model respectively. Also, a comparison of the frequency responses of the theoretical model and the estimated FIR filter is shown in Figure 4(c). We observe that the estimated FIR filter tends to be closer to the the-

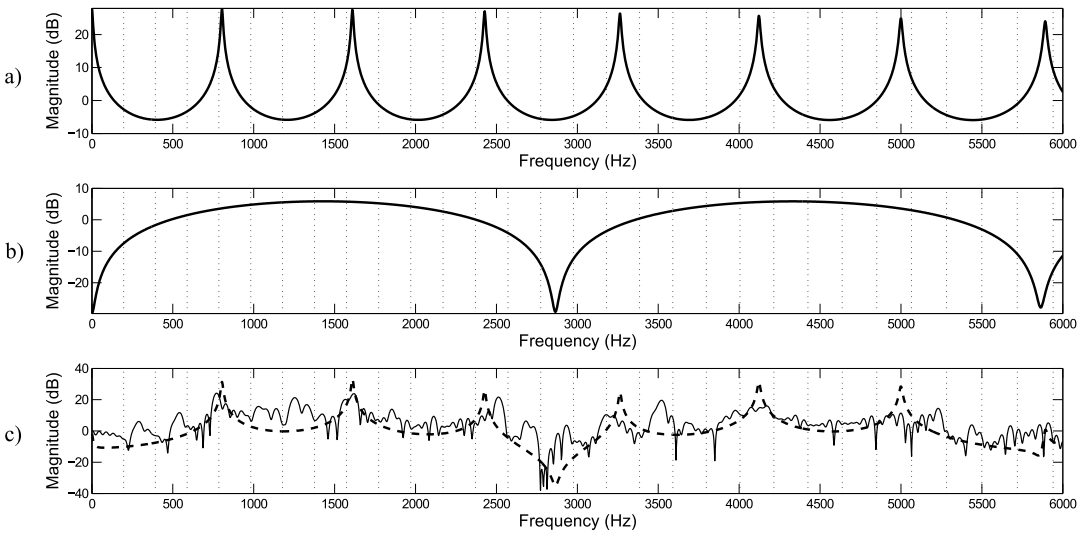


Figure 4: *(a) Frequency response of inverse neck pickup model; (b) frequency response of bridge pickup model; (c) frequency response of the cascaded filter (dashed line) and estimated FIR filter (solid line) for the electric guitar played at open G string. The vertical dotted lines show the frequencies of the partials of the tone.*

oretical model at or near the partial frequencies. This is because most of the energy of the input and target signals is found at the partials, so the filter optimisation is biased to give low errors at these frequencies rather than the frequencies between partials. By the same reasoning, the filter is less accurate at higher frequencies, where the signal has less energy. In addition, the nulls of the pickup model remove energy at specific frequencies. In theory, this could lead to numerical problems when inverting the model but we did not experience this problem with our data. It is either because the notches are not perfectly nulling or because the partial overtones never coincided with notch frequencies. Overall, the frequency response of the estimated FIR filter follows the curves of the theoretical model and captures the most important timbral features.

To explain the results in Table 1, we note that the inverse filter for the neck pickup has the poles most closely spaced, so more poles occur in the region where the signal has the most energy. Since we are using an FIR filter, the approximation of the poles will have some error, which is most noticeable in the case where the neck pickup is the input sound. The error in these cases is approximately one order of magnitude greater.

6. RESULTS: TESTS OF ROBUSTNESS

In guitar synthesisers, each string is processed by an individual filter to emulate the sound of an electric guitar. In Section 5, we extract a filter for a single string played at a particular plucking point and plucking dynamic. In this section we test the generality of learnt filters for each string, to assess the effect of different playing techniques such as variations in plucking dynamics and plucking positions.

In order to measure the robustness of the filter to such differences, we extract a filter for a particular input/target pair (the training pair) and test how well it performs given a different input/target pair (the testing pair). In the simplest case, the training and testing pairs are different instances (*repetitions*) of the same parameters (string, fret, dynamic, plucking position), but we also test for variations in one of the other parameters at a time (except for string). Thus, the variables that we analyse are *repetition*, *plucking position*, *plucking dynamic* and *fret position*. To keep the results manageable, we only report results where the input signal is from the neck pickup and target signal is from the bridge pickup.

6.1. Analysing Filter Generality

We used the guitar recordings described in Section 2. The process of analysing each variable can be explained by an example. In this case, we take the variable *repetition*. The steps for analysing the robustness of the filter to different *repetitions* are as follows:

1. Take three input/target pairs $x_i(n)$ and $y_i(n)$ where $i \in \{1, 2, 3\}$ is the index of the *repetition* and other variables remained constant.
2. Obtain the filters $h_i(n)$ as described in Section 5 for each *repetition*.
3. The input signals $x_i(n)$ are convolved with each filter $h_j(n)$, $j \in \{1, 2, 3\}$ separately to obtain estimated signals $\hat{y}_{i,j}(n)$. The distance $D(\hat{Y}_{i,j}, Y_i)$ between the estimated signal and target signal $y_i(n)$ is calculated.
4. Steps 1, 2 and 3 are repeated for all cases (i.e. for each combination of plucking position, plucking dynamic, fret position and string), leading to a total of 162 cases.

The same process can be used for analysing the robustness of the filters to different *plucking positions*, *plucking dynamics* and *fret*

Table 2: *Errors measured for filters applied to an input/target pair with different (a) repetition, (b) plucking position, (c) plucking dynamic, (d) fret position and (e) fret position (improved filter); where d is the distance of the plucking point from the bridge. All values are averages over 162 different cases, as described in Section 6.1.*

Input signal	$h_1(n)$ Rep 1	$h_2(n)$ Rep 2	$h_3(n)$ Rep 3
Repetition 1	**0.186**	0.240	0.260
Repetition 2	0.216	**0.184**	0.238
Repetition 3	0.217	0.225	**0.187**

(a)

Input signal	$h_1(n)$ d_1	$h_2(n)$ d_2	$h_3(n)$ d_3
$d_1 = 158.75mm$	**0.154**	0.391	0.705
$d_2 = 101.6mm$	0.597	**0.222**	0.366
$d_3 = 45mm$	0.836	0.453	**0.181**

(b)

Input signal	$h_1(n)$ *f*	$h_2(n)$ *mf*	$h_3(n)$ *p*
Forte, *f*	**0.195**	0.364	0.535
Mezzoforte, *mf*	0.358	**0.211**	0.287
Piano, *p*	0.352	0.258	**0.152**

(c)

Input signal	$h_1(n)$ Open string	$h_2(n)$ 5th fret	$h_3(n)$ 12th fret
Open string	**0.135**	0.705	0.591
5th fret	1.080	**0.111**	0.505
12th fret	0.773	1.949	**0.310**

(d)

Input Signal	$h_1(n)$ Open string & 5th fret	$h_2(n)$ Open string & 12th fret	$h_3(n)$ 5th & 12th fret	$h_4(n)$ Open string, 5th & 12th fret
Open string	**0.151**	**0.241**	0.432	**0.242**
5th fret	**0.121**	0.495	**0.152**	**0.152**
12th fret	0.644	**0.311**	**0.333**	**0.325**

(e)

positions where the variable *repetition* is exchanged with the variable to be analysed in the above four steps.

6.2. Results

Following the steps in Section 6.1, we obtain nine distances for each of 162 cases to analyse each variable. Each of the nine distances, or errors, is then averaged over all 162 cases. Tables 2(a), 2(b), 2(c) and 2(d) show the errors for analysing the robustness of the filters when applied to an input/target pair with a different *repetition*, *plucking position*, *plucking dynamic* or *fret position* respectively. For the learnt filters $h_i(n)$, the values of the variable that we are analysing are indexed by i. For instance, in Table 2(b), $h_1(n)$, $h_2(n)$ and $h_3(n)$ are extracted from input/target pairs that were plucked at 158.75 mm, 101.6 mm and 45 mm from the bridge respectively. The filters are then evaluated on input/target pairs from each of the three different plucking positions. The figures in bold emphasise the cases where the training and testing pairs coincide. These can be used a reference values, to obtain the loss in accuracy due to the variable under analysis.

As shown in Table 2(a), the increase in error when the filters are applied to different *repetitions* ranges from 16% to 39%. We would expect the filters to be reasonably robust towards other *repetitions*, because the notes are being plucked at similar positions, dynamics, frets and strings. Note that we did not use a mechanical plucking device, so there are slight random variations in playing technique between the repetitions, but there should be no systematic variation. Hence all non-bold values are quite consistent, because different *repetitions* have similar timbre.

According to Table 2(b), error increases by a factor of 2 to 5 when the input signal is convolved with a filter learnt from a different *plucking position*. In this case, the comb filter effect of plucking position creates nodes which effectively suppress information about the filter to be learnt. The filter $h_2(n)$ has a lower off-diagonal error than filters $h_1(n)$ and $h_3(n)$. Likewise input/target pair 2 has lower off-diagonal errors than the other pairs. It appears that the effect of the middle plucking point, like the position itself, is closer to the other plucking points than they are to each other.

Table 2(c) shows that the error also increases when the filter is applied to a signal with different *plucking dynamics*. Here the effect of changes in *plucking dynamics* is approximately a doubling of error, although for the filter $h_3(n)$ learnt from a quiet pluck, a much larger error is observed. The reason for this could be a lack of information for filter estimation at high frequencies, due to the signal's energy being concentrated towards lower frequencies in the case of p (piano) dynamic.

By far the largest errors are recorded in Table 2(d), when the filters are applied to different *fret positions*. Two reasons can be given for this result: first, the filter is learnt accurately only at the partials of the training tone; for a different testing tone, the frequency response of the filter is inaccurate. The second reason is that each different fret position results in a different comb filtering effect of the pickup. For example, the neck pickup is $\frac{1}{4}$ of the way along an open string, but $\frac{1}{3}$ of the way between the 5th fret and bridge, and $\frac{1}{2}$ way between the 12th fret and the bridge. The comb filtering effect of the pickup is to attenuate every 4th, 3rd or 2nd partial in the respective cases.

In order to improve the results from Table 2(d), we concatenated the input (respectively target) signals played on the open string, at the fifth fret and at the twelfth fret, in order to learn a composite filter. The filter $h_1(n)$ was learnt from the open string and fifth fret signal pairs, $h_2(n)$ from the open string and twelfth fret pairs, $h_3(n)$ from the fifth fret and twelfth fret pairs and filter $h_4(n)$ from all three pairs. The same filter order of 1024 coefficients is used for the composite filters. The filters were then evaluated on input/target pairs for the three fret positions. Table 2(e) shows considerable improvement compared to the errors measured in Table 2(d). The filter with the least error is $h_4(n)$, which uses information from all input/target pairs.

Figure 5 shows the composite filter, $h_4(n)$. We can observe

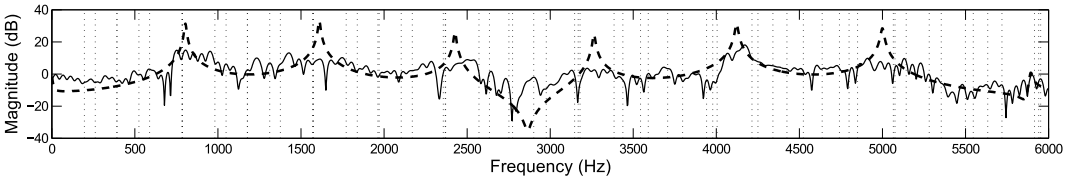


Figure 5: *Frequency response of the cascaded filter (dashed line) and the composite filter $h_4(n)$ (solid line) which is learnt from electric guitar played at open string ($f_0 = 196$ Hz), fifth fret ($f_0 = 262$ Hz) and twelfth fret ($f_0 = 392$ Hz). The electric guitar is played forte and plucked directly above the middle pickup. The vertical dotted lines show the frequencies of the partials of the tones.*

that the frequency response of the composite filter is flatter than the filter in Figure 4(c). This is because the composite filter has more information from which it can learn the frequency response. In particular, the frequency response of the filter is more accurate at the partials of fifth ($f_0 = 262$ Hz) and twelfth fret ($f_0 = 392$ Hz).

6.3. Comparisons Between Variables

Table 3: *Summary table for comparisons between each variables.*

Variables	Mean Diagonal	Mean Off-diagonal	Improved
Repetitions	0.186	0.233	
Plucking positions	0.186	0.558	
Plucking dynamics	0.186	0.359	
Fret positions	0.186	**0.934**	0.240

Table 3 summarises the results of Table 2. The second column shows the averages of the diagonal values, which is the global average error of transforming the neck pickup sound into the bridge pickup sound across all cases where the training and testing pairs coincide (thus it is independent of row). The third column shows the averages of the off-diagonal values for Tables 2(a), 2(b), 2(c) and 2(d) and the averages for the fifth column in Table 2(e). These results give insight into the relative contributions of the variables, and thus the robustness of the filters to changes in *repetition*, *plucking position*, *plucking dynamic* and *fret position*. The filters are most robust to changes in *repetition*, which should have no systematic difference. Changes in *plucking dynamics* double the error on average, and changes in *plucking position* triple the error. The filters are least robust to changes in *fret position*, which result in a 5-fold increase in error. By learning composite filters across multiple notes, a significant reduction in error appears possible.

7. CONCLUSIONS

We described a preliminary step towards altering the sound of an electric guitar in order to replicate a desired electric guitar sound in an audio recording. In this study, a method of transforming the sound of an electric guitar pickup into that of another pickup position has been described as a first step. Although we have not yet performed a formal listening test, informal listening suggests that the technique yields good results. This is supported by a spectral distance measure which shows that around 80% of the difference between input and target signal is reduced by the learnt filter in the case that the target signal is known. It is shown that an FIR filter with 1024 coefficients is sufficient for the current approach.

However, there are limitations to replicating the target signal in the practical use case, such as a guitar synthesiser, where the target signal is not known. As mentioned in Section 2, differences between playing techniques in the input and training signals will affect the accuracy of any emulation. We quantified these effects by testing learnt filters against input/target pairs that failed to match in one of four dimensions. While a small degradation is observed due to random differences between repetitions (which could correspond to the degree of overfitting of the learnt filter), we found that the filter is somewhat less robust when applied to an input with different playing technique (i.e. plucking position or dynamic), and not at all robust when the input changes fret positions to a different pitch.

The results for different fret positions can be improved by training the filter using multiple tones played on different frets along the string. This allows the filter to "fill the gaps" of unknown values in the frequency response between partials of a single training tone. This approach could also be applied to improve performance across different values of the other variables. Future work could compare alternative approaches, such as approximating the target frequency response by smoothing to obtain the spectral envelope, or using a parametric model to explicitly model the physical properties of the instrument and the playing gestures.

8. REFERENCES

[1] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects," *Journal of the Audio Engineering Society*, vol. 19, no. 6 (part I) and 7 (part II), 1971.

[2] J.O. Smith, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.

[3] K. Karplus and A. Strong, "Digital synthesis of plucked-string and drum timbres," *Computer Music Journal*, pp. 43–55, 1983.

[4] D.A. Jaffe and J.O. Smith, "Extensions of the Karplus-Strong plucked-string algorithm," *Computer Music Journal*, pp. 56–69, 1983.

[5] C.R. Sullivan, "Extending the Karplus-Strong algorithm to synthesize electric guitar timbres with distortion and feedback," *Computer Music Journal*, pp. 26–37, 1990.

[6] N. Lindroos, H. Penttinen and V. Välimäki, "Parametric electric guitar synthesis," *Computer Music Journal*, vol. 35, no. 3, pp. 18–27, 2011.

[7] R.C.D. Paiva, J. Pakarinen and V. Välimäki, "Acoustics and modeling of pickups," *Journal of the Audio Engineering Society*, vol. 60, no. 10, pp. 768–782, 2012.

[8] N. Fletcher and T. Rossing, *The Physics of Musical Instruments*, Springer, New York, NY, 1998.

[9] C. Traube and J.O. Smith, "Extracting the fingering and the plucking points on a guitar string from a recording," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA'01)*. Citeseer, 2001.

[10] H. Penttinen and V. Välimäki, "A time-domain approach to estimating the plucking point of guitar tones obtained with an under-saddle pickup," *Applied Acoustics*, vol. 65, no. 12, pp. 1207–1220, 2004.

[11] P.J. Celi, M.A. Doidic, D.W. Fruehling and M. Ryle, "Stringed instrument with embedded dsp modeling," Sept. 7 2004, US Patent 6,787,690.

[12] A. Hoshiai, "Musical tone signal forming device for a stringed musical instrument," Nov. 22 1994, US Patent 5,367,120.

[13] D. Barchiesi and J. Reiss, "Automatic target mixing using least-squares optimization of gains and equalization settings," in *Proceedings of the 12th Conference on Digital Audio Effects (DAFx-09), Como, Italy*, 2009, pp. 7–14.

[14] D. Barchiesi and J. Reiss, "Reverse engineering of a mix," *Journal of the Audio Engineering Society*, vol. 58, no. 7/8, pp. 563–576, 2010.

[15] D.G. Manolakis, V.K. Ingle, and S.M. Kogon, *Statistical and adaptive signal processing: spectral estimation, signal modeling, adaptive filtering, and array processing*, vol. 46, Artech House Norwood, 2005.

[16] A. Horner, J. Beauchamp and L. Haken, "Machine tongues XVI: Genetic algorithms and their application to FM matching synthesis," *Computer Music Journal*, pp. 17–29, 1993.

[17] R.A. Garcia, *Automatic generation of sound synthesis techniques*, Ph.D. thesis, Citeseer, 2001.

[18] Y. Lai, S. Jeng, D. Liu and Y. Liu, "Automated optimization of parameters for FM sound synthesis with genetic algorithms," in *International Workshop on Computer Music and Audio Technology*, 2006.

[19] U. Zölzer and X. Amatriain, *DAFX: digital audio effects*, vol. 1, Wiley Online Library, 2002.

[20] M. Mustonen, D. Kartofelev, A. Stulov and V. Välimäki, "Experimental verification of pickup nonlinearity," in *Proceedings International Symposium on Musical Acoustics (ISMA 2014), Le Mans, France*, 2014, vol. 1.

LARGE STENCIL OPERATIONS FOR GPU-BASED 3-D ACOUSTICS SIMULATIONS

Brian Hamilton[†], [*] *Craig J. Webb*[†], *Alan Gray*[‡], *Stefan Bilbao*[†]

[†]Acoustics and Audio Group,
[‡]EPCC,
University of Edinburgh

ABSTRACT

Stencil operations are often a key component when performing acoustics simulations, for which the specific choice of implementation can have a significant effect on both accuracy and computational performance. This paper presents a detailed investigation of computational performance for GPU-based stencil operations in two-step finite difference schemes, using stencils of varying shape and size (ranging from seven to more than 450 points in size). Using an Nvidia K20 GPU, it is found that as the stencil size increases, compute times increase less than that naively expected by considering only the number of computational operations involved, because performance is instead determined by data transfer times throughout the GPU memory architecture. With regards to the effects of stencil shape, performance obtained with stencils that are compact in space is mainly due to efficient use of the read-only data (texture) cache on the K20, and performance obtained with standard high-order stencils is due to increased memory bandwidth usage, compensating for lower cache hit rates. Also in this study, a brief comparison is made with performance results from a related, recent study that used a shared memory approach on a GTX 670 GPU device. It is found that by making efficient use of a GTX 660Ti GPU—whose computational performance is generally lower than that of a GTX 670—similar or better performance to those results can be achieved without the use of shared memory.

1. INTRODUCTION

At the heart of many grid-based physics simulations is a stencil operation approximating some differential operator of interest. In the context of acoustics modelling based on the 3-D wave-equation [1], the finite difference (FD) method with explicit time integration (also known as finite difference time domain (FDTD)) is a standard approach [2–4]. The stencil operation involved is typically regular over a Cartesian grid and is easily parallelised, making the FD method a suitable candidate for acceleration on graphics processing units (GPUs). General programming on high-performance computing devices has been studied by many in the context of 3-D acoustic wave equations [5–7], and more specifically for GPU-based 3-D room acoustics simulations [8–17].

It is well-known that the simplest FDTD scheme suffers from significant numerical dispersion errors. Mitigating such errors to tolerable levels may require grid refinements that can dramatically increase the computational resources required. As such, there has been much work in designing more accurate FDTD schemes, often by taking into account more than the standard number of points (seven in space and three in time); examples include explicit and implicit schemes using 27-point stencils [15, 18–20], schemes that make use of standard high-order spatial differencing [3, 21], and high-order schemes based on modified equation methods [3, 21].

Of particular interest to this study is a general construction for 3-D discrete Laplacians on the Cartesian grid recently presented [22] that allows for a general family of two-step FDTD schemes utilising stencils of almost arbitrary size and shape. This general construction is beneficial in that it provides a large set of free parameters to be optimised for the purposes of minimising dispersion or isotropy errors [22]. For example, an eight-parameter 125-point ($5\times5\times5$) stencil scheme was shown to have at most two-percent absolute dispersion error over at least 85% of the normalised wavenumbers simulated [22, Fig. 8], which is a significant improvement over conventional 27-point schemes that meet the same criterion over at most 40% of the normalised wavenumbers simulated [20]. As such, by taking on *linear* increases in operations over standard 7-point or 27-point schemes, one can meet any desired accuracy (in terms of numerical dispersion) for a given application without the usual *cubic* or *quartic* increases in computational costs associated with grid refinement, thereby approaching the minimum grid requirements dictated only by sampling considerations.

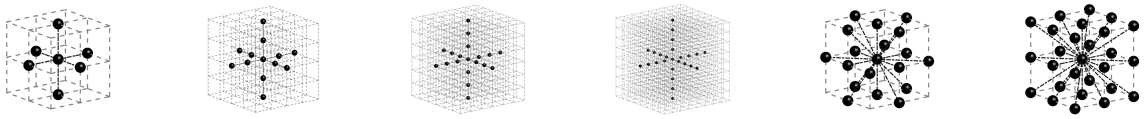
For the purposes of large-scale 3-D acoustics simulations, such large-stencil schemes should also benefit from parallel implementations and memory caches on GPU devices. It is thus of interest, and the purpose of this study, to determine how the size and shape of such stencils affects processing throughput on a GPU device, where performance is not necessarily determined by the number of operations, but rather by data movement. Related work, not specific to acoustics simulations, can be found in [23–26]. We note that boundary conditions become invariably more difficult as the stencil size increases, but as this study is a preliminary one on GPU performance, the boundary problem will not be addressed here. The organisation of this paper is as follows. Section 2 presents the two-step schemes, stencil operations, and compact stencils under study. Section 3 features the GPU kernels and testing procedures used. Results from K20 GPU tests are presented in Section 4, followed by a brief comparison with shared memory approaches using a GTX 660Ti GPU.

2. BACKGROUND

2.1. Two-step finite difference schemes

In order to express the two-step finite difference schemes under consideration, we define a fully-discrete Cartesian grid function

[*] This work was supported by the European Research Council, under grant StG-2011-279068-NESS, and by the Natural Sciences and Engineering Research Council of Canada. Thanks to Shiv Upadhyay for the use of the GTX 660Ti GPU card.
Email of corresponding author: brian.hamilton@ed.ac.uk.



(a) 7-point stencil (b) $M = 2$, leggy (c) $M = 3$, leggy (d) $M = 4$, leggy (e) 19-point, compact (f) 27-point, compact

Figure 1: Various stencils on the cubic lattice. These include $(6M + 1)$-point leggy stencils for $M = 1, 2, 3, 4$ and compact-in-space 19- and 27-point stencils. Bounding boxes are rescaled for clarity.

$u_{\boldsymbol{i}}^n = u(nT, \boldsymbol{i}X)$, where $n \in \mathbb{Z}^+$, T is the time-step and X is the grid spacing, and $\boldsymbol{i} = (i_x, i_y, i_z) \in \mathbb{Z}^3$ are discrete spatial indices. In this case the Cartesian grid is simply $X\mathbb{Z}^3$. An arbitrary 3-D stencil operation will be defined as:

$$\delta_{\mathbb{S},\boldsymbol{\gamma}} u_{\boldsymbol{i}}^n := \sum_{k=0}^{K-1} \gamma_k u_{\boldsymbol{i}+\boldsymbol{l}_k}^n \tag{1}$$

where $\mathbb{S} = \{\boldsymbol{l}_0, \ldots, \boldsymbol{l}_{K-1}\}$ is a 3-D stencil (a set of K points $\boldsymbol{l}_k \in \mathbb{Z}^3$) and $\boldsymbol{\gamma} = (\gamma_0, \ldots, \gamma_{K-1})$ is a vector of non-zero scalar coefficients (stencil weights). The stencil weights are generally chosen to provide an approximation to a particular spatial operator acting on u, such as, e.g., a direction derivative $\mathbf{n} \cdot \nabla u$ for $\mathbf{n} \in \mathbb{R}^3$, the 3-D Laplacian $\Delta := \partial_x^2 + \partial_y^2 + \partial_z^2$, or the biharmonic Δ^2.

This paper is concerned with two-step finite difference schemes of the form

$$u_{\boldsymbol{i}}^{n+1} = \delta_{\mathbb{S},\boldsymbol{\gamma}} u_{\boldsymbol{i}}^n - u_{\boldsymbol{i}}^{n-1} \tag{2}$$

where $\delta_{\mathbb{S},\boldsymbol{\gamma}}$ is chosen such that the above is an approximation to the 3-D wave equation with a formal accuracy that is at least second-order in time and space. Such two-step schemes require two states to be stored in memory since $u_{\boldsymbol{i}}^{n+1}$ can overwrite $u_{\boldsymbol{i}}^{n-1}$ in place. An example is the simplest scheme for the 3-D wave equation, in which $\mathbb{S}$ and $\boldsymbol{\gamma}$ are:

$$\begin{aligned} \mathbb{S} &= \{(0,0,0), (\pm 1,0,0), (0,\pm 1,0), (0,0,\pm 1)\} \\ \gamma_0 &= (2 - 6\lambda^2), \quad \gamma_k = \lambda^2, \quad k = 1, \ldots, 6 \end{aligned} \tag{3}$$

and $\lambda := cT/X$ is the Courant number, which is generally bounded for stability as $\lambda \leq \sqrt{1/3}$, and c represents the wave speed. This scheme employs a seven-point stencil, as illustrated in Fig. 1(a). In this simple scheme, one has that

$$\frac{1}{T^2}(u_{\boldsymbol{i}}^{n+1} - \delta_{\mathbb{S},\boldsymbol{\gamma}} u_{\boldsymbol{i}}^n + u_{\boldsymbol{i}}^{n-1}) = \partial_t^2 u - c^2 \Delta u + O(T^2) + O(X^2) \tag{4}$$

where $t \in \mathbb{R}^+$ is time. Thus, the scheme provides an approximate solution $u(t, \mathbf{x})$ to the wave equation at times $t = nT$ and spatial positions $\mathbf{x} = \boldsymbol{i}X$.

The seven-point stencil in Fig. 1(a) also belongs to the family of $(6M + 1)$-point stencils, henceforth called "leggy stencils", of which examples are shown in Figs. 1(a)-1(c) for $M \leq 4$. The stencil weights for these leggy stencils can be chosen such that the accuracy of the approximate solution is of order two in time and $2M$ in space, i.e.,$(2, 2M)$-accurate. To this end, it is worth expressing these leggy schemes in a more familiar form:

$$u_{\boldsymbol{i}}^{n+1} = 2u_{\boldsymbol{i}}^n - u_{\boldsymbol{i}}^{n-1} + \lambda^2 \textstyle\sum_{d=1}^{3} \sum_{m=-M}^{M} \beta_{M,m}\, u_{\boldsymbol{i}+m\hat{\mathbf{e}}_d}^n \tag{5}$$

where $\hat{\mathbf{e}}_d$ are the standard unit vectors in $\mathbb{R}^3$. The coefficients $\beta_{M,m} = \beta_{M,-m}$ that provide $2M$th-order accuracy in space for the Laplacian in 1-D, and by extension for the 3-D Laplacian, can be found in, e.g., [3, 21, 27] along with stability conditions for the two-step schemes in [21]. Such schemes and stencils have seen use in the room acoustics literature [11] and can be regarded as special cases of the general family of schemes presented in [22]. In a more recent study they are called "large-star stencil" schemes [16]. The simplest seven-point scheme is a special case with $M = 1$ and $(\beta_{1,0}, \beta_{1,1}) = (-2, 1)$. Note that (5) is still of the form (2), since one can set $\gamma_0 = 2 - (\lambda^2/3)\beta_{M,0}$.

Other approaches to improving FDTD schemes have focussed on achieving *isotropic* error, initially with the goal of making use of frequency-warping techniques [18]. This leads to the so-called "interpolated schemes" [18–20], which employ 19- and 27-point stencils that are compact in space [28, 29]. These stencils are illustrated in Figs. 1(e) and 1(f). For this study we will consider two possible generalisations of such compact stencils, to be defined in Section 2.2.

Before proceeding, it is important to note that the purpose of this paper is not to determine optimal stencil weights, or to compare the accuracy of various compact stencils or leggy stencils. The focus here is on computational aspects of stencil operations on GPU devices. To this end, the performance metric used here will be the "compute time per node" (CTPN), where a "node" is simply a grid point. In other words, the CTPN is the time required to process a single grid point, taken as an average over many grid points and time-steps. It has been demonstrated in previous studies [13, 14] that this measure of performance, which is simply the scaled inverse of the commonly used "Megavoxels per second" metric, is more or less constant for different grid sizes within a fixed scheme, provided that a significant proportion of GPU memory is used (i.e., as long as the occupancy rate is sufficiently high). From such performance data, it is straightforward to predict final compute times (neglecting specialised boundary updates) once stencil weights and grid densities are chosen appropriately for a model problem.

2.2. 3-D compact stencils

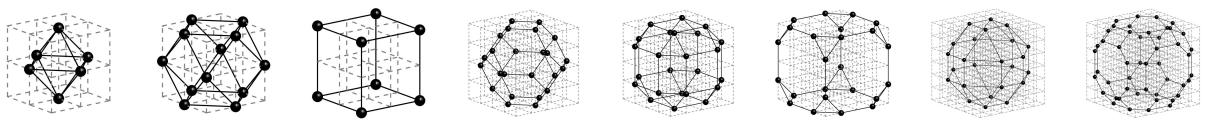
In order to construct 3-D compact stencils—beyond the standard 27-point variants [18–20, 28]—we start with a description of shells of points on the cubic lattice, $\mathbb{Z}^3$. Consider an integer-valued triplet $\boldsymbol{q} = (q_1, q_2, q_3) \in \mathbb{Q}$ with the set $\mathbb{Q}$ defined as

$$\mathbb{Q} := \{\boldsymbol{q} \in \mathbb{Z}^3 : q_1 \geq q_2 \geq q_3 \geq 0, q_1 \geq 1\} \tag{6}$$

and let $\mathcal{P}(\boldsymbol{q})$ be a function that returns the set of *unique* permutations of $(\pm q_1, \pm q_2, \pm q_3)$. Letting $|\mathcal{P}(\boldsymbol{q})|$ denote the cardinality of the set $\mathcal{P}(\boldsymbol{q})$, it is straightforward to work out that $|\mathcal{P}(\boldsymbol{q})| \in \{6, 8, 12, 24, 48\}$. Each set of triplets $\mathcal{P}(\boldsymbol{q})$ represents a shell of points on $\mathbb{Z}^3$, as illustrated in Fig. 2.

Consider now a set of P distinct triplets, $\Omega = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_P\}$, representing a set of shells, and define the function $\mathcal{S}(\Omega)$ as:

$$\mathcal{S}(\Omega) := (0,0,0) \cup \left(\textstyle\bigcup_{p=1}^{P} \mathcal{P}(\boldsymbol{q}_p)\right) \tag{7}$$



(a) $\boldsymbol{q} = (1,0,0)$ (b) $\boldsymbol{q} = (1,1,0)$ (c) $\boldsymbol{q} = (1,1,1)$ (d) $\boldsymbol{q} = (2,1,0)$ (e) $\boldsymbol{q} = (2,1,1)$ (f) $\boldsymbol{q} = (2,2,1)$ (g) $\boldsymbol{q} = (3,1,1)$ (h) $\boldsymbol{q} = (3,2,1)$

Figure 2: Various shells of points, $\mathcal{P}(\boldsymbol{q})$, on the cubic lattice and associated convex hulls. Bounding boxes rescaled for clarity.

Each output set $\mathcal{S}(\Omega)$ represents a 3-D stencil on $\mathbb{Z}^3$. For example, the $(6M+1)$-point leggy stencils could be written as $\mathbb{S}^{(\mathrm{l})}_M := \mathcal{S}(\Omega^{(\mathrm{l})}_M)$, where

$$\Omega^{(\mathrm{l})}_M := \{(1,0,0), \ldots, (M,0,0)\} \tag{8}$$

There are obviously many combinations of triplets $\boldsymbol{q}$ that may be used to build 3-D stencils. For this study, we consider a small subset of the many possibilities, namely two possible generalisations of the aforementioned 27-point compact stencils. The first generalisation will be called the "compact-in-space stencils", which are defined here as $\mathbb{S}^{(\mathrm{c})}_R := \mathcal{S}(\Omega^{(\mathrm{c})}_R)$, where $\Omega^{(\mathrm{c})}_R$ is a set of triplets $\boldsymbol{q}$ defined as:

$$\Omega^{(\mathrm{c})}_R := \{\boldsymbol{q} \in \mathbb{Q} : \|\boldsymbol{q}\|^2 \leq R\} \tag{9}$$

Here, R is a positive integer and $\|\boldsymbol{q}\|$ denotes the Euclidean norm of $\boldsymbol{q}$. Each set $\Omega^{(\mathrm{c})}_R$ leads to a compact-in-space stencil $\mathbb{S}^{(\mathrm{c})}_R$ whose convex hull has a circumradius of $\sqrt{R}$. Some of these compact-in-space stencils are displayed in Fig. 3, along with $K = |\mathbb{S}^{(\mathrm{c})}_R|$, the number of points in each stencil. The sequence of R values that permit valid compact-in-space stencils can be found in [30, Table 4.3]. For example, the 7-, 19-, and 27-point stencils are, respectively, $\mathbb{S}^{(\mathrm{c})}_1$, $\mathbb{S}^{(\mathrm{c})}_2$, and $\mathbb{S}^{(\mathrm{c})}_3$. It is well-known that such stencils (for $K \geq 19$) can provide isotropic error in approximations to the Laplacian [28, 29]. High orders of isotropy become easier to achieve with larger stencils and more degrees of freedom [22], and subsequently can lead to two-step schemes with high orders of accuracy (in both time and space) through the use of modified equation methods [21].

As an aside, in regards to a recent study [16] it is important to point out that, strictly speaking, the two-step leggy-stencil ("large-star") schemes ($(2, 2M)$-accurate) investigated in [16] are *not* higher-order accurate for the wave equation (for all $M \geq 1$); in such schemes, the global accuracy remains second-order since the temporal error (second-order) dominates in the limit of small h provided that λ is fixed (such as, e.g., to the stability limit [16]). This fact does not appear to be taken into account in [16] when making practical comparisons to two-step schemes that are indeed higher-order accurate ($(2M, 2M)$-accurate) derived using modified equation methods [15, 31]. We also note that, contrary to what is suggested in [16], the accuracy of a stencil, or more importantly, of the scheme in which it is used, is not related to its size in points in a simple manner under the modified equation framework; see [31].

Returning to the compact stencils, we note that by the nature of the Euclidean norm, the compact-in-space stencils tend towards a spherical distribution in space, as seen in Fig. 3. One could also consider stencils that are "compact" with respect to a different norm. For example, a stencil made up of a 125-point cube (5×5×5) is not compact in the sense defined above, but can be seen as "compact" in the sense that it compactly fills a cubic volume of space (i.e., compact in a Chebyshev-type norm). As such, this family of stencils, which also comprises the aforementioned 7-, 19-, and 27-point stencils, will be referred to as "box-compact". These stencils can be defined as $\mathbb{S}^{(\mathrm{b})}_{\boldsymbol{q}} := \mathcal{S}(\Omega^{(\mathrm{b})}_{\boldsymbol{q}})$ where $\Omega^{(\mathrm{b})}_{\boldsymbol{q}}$ is defined as:

$$\Omega^{(\mathrm{b})}_{\boldsymbol{q}} := \{\boldsymbol{q}' \in \mathbb{Q} : (q'_1, q'_2, q'_3) \preceq (q_1, q_2, q_3)\} \tag{10}$$

Here, $(q'_1, q'_2, q'_3) \preceq (q_1, q_2, q_3)$ means that (q'_1, q'_2, q'_3) precedes or is the same as (q_1, q_2, q_3) in a lexicographic ordering of the set $\mathbb{Q}$; i.e., $(q'_1, q'_2, q'_3) \preceq (q_1, q_2, q_3)$ iff (a) $q'_1 < q_1$ or (b) $q'_1 = q_1$ and $q'_2 < q_2$ or (c) $q'_1 = q_1$ and $q'_2 = q_2$ and $q'_3 < q_3$.

The set of compact-in-space stencils and the above-defined set of box-compact stencils have some elements in common; e.g., $\mathbb{S}^{(\mathrm{c})}_R$ with $R \in \{1, 2, 3, 4, 5, 6, 8, 12, 13, 14\}$ have equivalent counterparts $\mathbb{S}^{(\mathrm{b})}_{(q_1,q_2,q_3)}$, but otherwise they lead to different stencil shapes. Some box-compact stencils are displayed in Fig. 4. Under this notation, the aforementioned 125-point stencil would be $\mathbb{S}^{(\mathrm{c})}_{(2,2,2)}$, as seen in Fig. 4(a).

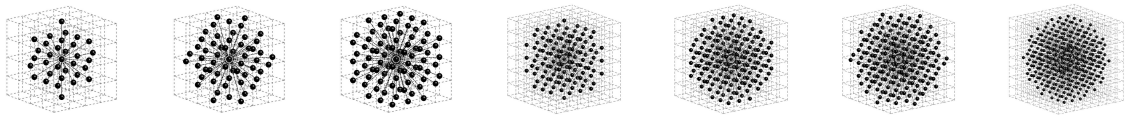
As mentioned previously, a general construction for discrete Laplacians utilising any 3-D such stencil $\mathcal{S}(\Omega)$ was presented in [22], leading to a general family of two-step schemes of which all of the examples mentioned in Section 2.1 are special cases. For brevity, the full derivation is left out, since for GPU performance comparisons (in terms of the number of grid points that may be processed in a certain amount of time) the coefficients (non-zero) are immaterial. We refer the reader to [22, Section VI] for the full derivation of the discrete Laplacian operators.

3. GPU IMPLEMENTATIONS

The GPU device that is used for the majority of the testing is an Nvidia Tesla K20 device (Kepler architecture), with error-correction checking (ECC) turned off. This card has 5119 MiB of device RAM (global memory) with a theoretical maximum memory bandwidth of 208 GB/s, although using the STREAM benchmark [32], we find that the peak copy rate is approximately 172 GB/s. The theoretical compute performance of this card is 3.52×10^{12} floating-point operations per second (FLOPS) (3.52 TFLOPS) and 1.17 TFLOPS in single and double precision, respectively. The GPU device is programmed to execute CUDA kernels.

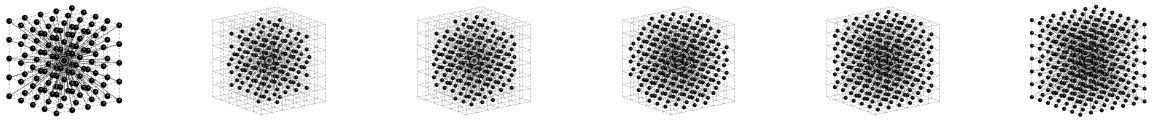
3.1. Algorithm details

Before presenting the testing procedure and the kernels under test, it is worth taking a moment to count the number of floating-point operations (FLOPs) required for the stencil operation $\delta_{\mathbb{S},\boldsymbol{\gamma}} u^n_{\boldsymbol{i}}$ in (1). At first glance it appears that (1) requires K multiplications and $K-1$ additions ($2K-1$ FLOPs) per grid point. However, when fused multiply-add (FMA) instructions are available, as is often the case with GPU devices, the operation can be implemented using $K-1$ FMAs and one multiplication. Since a FMA counts as two FLOPs and only one FLOP instruction, this gives $2K-1$ FLOPs and K floating-point (FP) instructions (executed in K clock cycles) per grid point.



(a) $\mathbb{S}_4^{(c)}$, $K = 33$ (b) $\mathbb{S}_5^{(c)}$, $K = 57$ (c) $\mathbb{S}_6^{(c)}$, $K = 81$ (d) $\mathbb{S}_9^{(c)}$, $K = 123$ (e) $\mathbb{S}_{11}^{(c)}$, $K = 171$ (f) $\mathbb{S}_{13}^{(c)}$, $K = 203$ (g) $\mathbb{S}_{18}^{(c)}$, $K = 341$

Figure 3: Some compact-in-space K-point stencils $\mathbb{S}_R^{(c)}$ on the cubic lattice, reaching out a maximum squared distance R from the origin.



(a) $\mathbb{S}_{(2,2,2)}^{(b)}$, $K = 125$ (b) $\mathbb{S}_{(3,1,0)}^{(b)}$, $K = 155$ (c) $\mathbb{S}_{(3,1,1)}^{(b)}$, $K = 179$ (d) $\mathbb{S}_{(3,2,1)}^{(b)}$, $K = 251$ (e) $\mathbb{S}_{(3,2,2)}^{(b)}$, $K = 275$ (f) $\mathbb{S}_{(3,3,3)}^{(b)}$, $K = 343$

Figure 4: Some box-compact K-point stencils $\mathbb{S}_{\boldsymbol{q}}^{(b)}$ on the cubic lattice, defined by (10).

It is worth pointing out that for the discrete Laplacian operators, there are usually only $P + 1$ distinct stencil weights, where $P = |\Omega|$. In this case, the stencil operation could also be factorised and written in the form:

$$\delta_{\mathbb{S},\boldsymbol{\gamma}} u_{\boldsymbol{i}}^n = \gamma_0 u_{\boldsymbol{i}}^n + \sum_{p=1}^{P} \gamma_p \left(\sum_{\boldsymbol{l} \in \mathcal{P}(\boldsymbol{q}_p)} u_{\boldsymbol{i}+\boldsymbol{l}}^n \right) \qquad (11)$$

When carried out as suggested by the above formulation, this calculation amounts to $K - P - 1$ additions, P FMAs, and one multiply ($K + P$ FLOPs, K FP instructions) per grid point. These stencil operations, (1) and (11), would perform identically on a GPU if the performance was solely determined by the number of instructions required. However, the time taken for data to travel throughout the GPU memory architecture must also be considered. It is worth noting that the former approach leads to a simpler generalised kernel implementation (requiring only one for-loop) and it inherently utilises more of the theoretical peak FLOPS rate from the GPU. In both cases, the extra subtraction operation required for (2) adds a single FP instruction. We also note that the FLOP counts presented here differ from those recently presented [16, 17], since FMA instructions, which the tested GPUs would have invariably used, were not taken into account.

In terms of memory reads and writes, the two-step update (2) requires $K + 1$ read operations and one write operation (not counting reading $\boldsymbol{l}_k$ and $\boldsymbol{\gamma}$). As mentioned previously, the storage requirements of the scheme are $2N$ for a grid of size N, since u^{n-1} can be overwritten by u^{n+1} after being read. The K read operations required for $\delta_{\mathbb{S},\gamma} u_{\boldsymbol{i}}^n$ are not followed by overwrites, which allows for the use of read-only data cache optimisations, as will be explained shortly. There is also a requirement to store the K coefficients in $\boldsymbol{\gamma}$, or only the distinct $P + 1$ if so desired, but this storage is negligible because, generally, $N \ggg K$.

The leggy-stencil family of schemes is also tested alongside the compact stencils in order to determine if the varying shape of the stencil plays a significant role in GPU performance. While these leggy schemes are encapsulated by (2) and (1), a more specialised kernel implementation is suggested by (5) that could lead to improvements in memory coalesced reads on the GPU. However, this comes at the cost of a slight redundancy, since $u_{\boldsymbol{i}}^n$ will be read and operated on three times instead of one. This alternative calculation for the stencil operation requires $K + 2$ FMAs, and one extra addition is required for the two-step update, for a total of $K + 3$ FP instructions ($2K + 5$ FLOPs) per grid point.

3.2. Testing procedure

The first twenty stencils in each family of stencils ($\mathbb{S}_R^{(c)}$, $\mathbb{S}_{\boldsymbol{q}}^{(b)}$, and $\mathbb{S}_M^{(l)}$) are tested for a box-shaped grid; the largest of these stencils is $\mathbb{S}_{22}^{(c)}$ with $K = 461$. Since the K20 card is capable of performing single and double precision FLOPs, two sets of grid sizes are used in order to use up all or half of the available memory. When testing single precision floating point, the dimensions of the grid ($N_x \times N_y \times N_z$) is either 928×800×750 or 720×640×560 and when double precision is being tested, the dimensions of the grid is 672×660×600 or 640×480×420. For each precision level, the former grid size is called "medium" and the latter "large". In each case, the grid is supplemented by a halo of N_h ghost-points (the minimum number required, i.e., the inradius of the bounding box for a given stencil) that are not updated, and thus do not factor into CTPN averages. The command-line profiler `nvprof` is used to obtain accurate estimations of average kernel execution times through the use of its kernel replay mode.

3.3. GPU Kernels

The first GPU kernel under test, `KernelA`, is a straightforward and general implementation of the two-step update (2). It appears in Fig. 5. In this kernel implementation, the grid is of size `Nxh`×`Nyh`×`Nzh`, which includes the `Nh`-thick halo of ghost points. These integer values are defined using C-preprocessor pragma statements (defined with # operator), along with `K`, the number of points in the stencil, and `ReaL`, which can be defined (with #) as either `float` or `double`. The arrays `gamma` (of type `ReaL`) and `offset` (of type `int`) represent stencil weights and linear offsets (linear decompositions of $\boldsymbol{l}_k \in \mathbb{S}$), respectively. These arrays are stored in the 64 KB of read-only constant memory available on the chip and accessible to all threads. Since the stencil weights and initial grid values are not important for performance analysis, the two states (`u0` and `u1`) and the stencil weights are assigned random floating-point values in the setup CPU host-code, which is also where the array `offset` is loaded with the appropriate linear offsets for the stencil under test.

A maximum threading (3-D tiling) approach is employed in this study, as implemented in lines 6-8 in Fig. 5, as opposed to, e.g., a 2-D slicing or 2-D tiling approach (see [33] for more details). In this case, the 3-D grid is decomposed in all three dimensions, with each 3-D subset of the grid assigned to a CUDA 3-D thread

```
1  __global__ void
       KernelA(ReaL *u0, const ReaL * __restrict__ u1){
2     //u1 is read from read-only data (texture) cache
3     //Nh,Nxh,Nyh,Nzh,K are #-defined ints
4     //ReaL gamma[K] is stored in constant memory
5     //int offset[K] is stored in constant memory
6     int ix = blockIdx.x*blockDim.x + threadIdx.x + Nh;
7     int iy = blockIdx.y*blockDim.y + threadIdx.y + Nh;
8     int iz = blockIdx.z*blockDim.z + threadIdx.z + Nh;
9     //condition to prevent illegal memory accesses
10    if (ix<(Nxh-Nh) && iy<(Nyh-Nh) && iz<(Nzh-Nh)){
11       //get linear index of current point
12       int cp = iz*Nxh*Nyh + iy*Nxh + ix;
13       //read previous value from global memory
14       ReaL tmp = -u0[cp];
15       //general stencil operation
16       for(int k=0;k<K;k++){
17          tmp += gamma[k]*u1[cp+offset[k]];
18       }
19       //write final value back to global memory
20       u0[cp] = tmp;
21    }
22 }
```

Figure 5: `KernelA`: CUDA kernel for compact and leggy stencils. See inline comments for implementation details.

```
1  __global__ void
       KernelB(ReaL *u0, const ReaL * __restrict__ u1){
2     //u1 is read from read-only data (texture) cache
3     //Nh,Nxh,Nyh,Nzh,K are #-defined ints
4     //ReaL gamma[K] is stored in constant memory
5     //int offset[K] is stored in constant memory
6     int ix = blockIdx.x*blockDim.x + threadIdx.x + Nh;
7     int iy = blockIdx.y*blockDim.y + threadIdx.y + Nh;
8     int iz = blockIdx.z*blockDim.z + threadIdx.z + Nh;
9     //condition to prevent illegal memory accesses
10    if (ix<(Nxh-Nh) && iy<(Nyh-Nh) && iz<(Nzh-Nh)){
11       //get linear index of current point
12       int cp = iz*Nxh*Nyh + iy*Nxh + ix;
13       //read previous value from global memory
14       ReaL tmp = -u0[cp];
15       //leggy stencil operation
16       for(int m=-Nh;m<=Nh;m++){
17          ReaL gamma_m=gamma[abs(m)];
18          tmp += gamma_m*u1[cp+m];
19          tmp += gamma_m*u1[cp+m*Nxh];
20          tmp += gamma_m*u1[cp+m*Nxh*Nyh];
21       }
22       //write final value back to global memory
23       u0[cp] = tmp;
24    }
25 }
```

Figure 6: `KernelB`: Alternative CUDA kernel for leggy stencils. See inline comments for implementation details.

block, with each thread operating on a single grid point. Enough thread blocks are issued to cover the 3-D grid, not including the halo of ghost points. The thread block size used for this test was 32×4×2, which generally provided high occupancy rates. The conditional statement at line 10 in the kernel ensures that illegal memory accesses are not encountered when the grid size is not an integer multiple of the thread-block size.

It is important to point out the `__const` and `restrict` qualifiers used for the variable `u1`, which represents $u_{\boldsymbol{i}}^n$. These qualifiers signal to the compiler (`nvcc`) to make use of the read-only data cache (texture cache) for the array `u1`, which can be loaded directly from the L2 cache [34]. The L1 cache does not cache global memory reads in the K20 GPU, but the L1 cache can be configured as a shared memory unit. Shared memory implementations, while commonly used for leggy stencil schemes [5, 16], are not directly explored in this study due to space constraints, although more will be said about shared memory approaches in Section 4.3.

The second kernel under test, `KernelB`, appearing in Fig. 6, is specific to the $(6M+1)$-point leggy stencils and is meant to resemble the specific formulation (5). In this kernel, `Nh` is equal to M, and for consistency with the 3-D wave equation `gamma[m]` would be equal to $\beta_{M,m}$ for $m \geq 1$ and `gamma[0]` would be equal to $2/3 - \lambda^2\beta_{M,0}$.

4. RESULTS AND DISCUSSION

4.1. Timings

The timing results from the tests are displayed in Fig. 7. Starting with the compact stencils in Fig. 7(a), we note that, for the most part, the average compute times per node (CTPNs) scale linearly with the stencil size. As expected, single precision results in faster compute times than double precision, aside from the 27-point compact stencil which has similar performance in both precisions. Also as expected, the CTPNs are relatively invariant to the two grid sizes (medium and large), justifying the discussion at the end of Section 2.1. Within each precision, we note that it makes little difference whether the stencil is "compact-in-space" or "box-compact", aside from dips in the CTPNs in double precision for the box-compact $M \times M \times M$ stencils, $\mathbb{S}^{(\mathrm{b})}_{(M,M,M)}$, which is perhaps due to more efficient memory access patterns.

The leggy stencil CTPNs are displayed in Fig. 7(b) for the two different kernels. It can be seen that for leggy stencils, `KernelB` generally provides slightly faster CTPNs than `KernelA`. We also note that there are dips in the CTPNs in double precision when M is a multiple of eight, which may be due to increased memory coalescing.

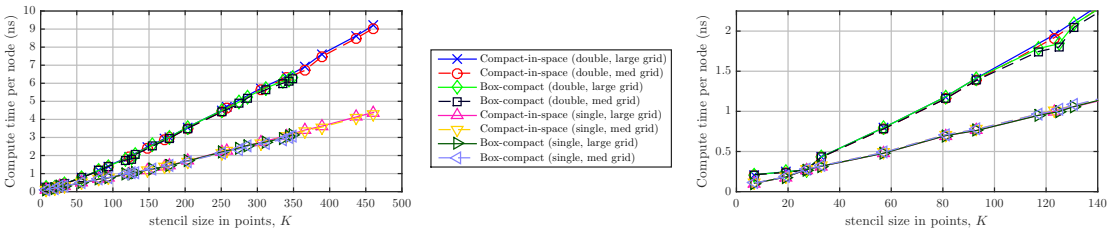
Fig. 7(c) presents a comparison of the compact stencils and the leggy stencils (with `KernelB`) in single and double precision, along with a prediction of CTPNs, stemming from increases over the simplest 7-point scheme in memory reads or in FP instructions. All of the obtained CTPNs are lower than what would be expected from this somewhat naive prediction. Finally, we note that the CTPNs vary slightly between leggy and compact stencils, but overall, the CTPNs are relatively invariant to the shape of the stencil.

In order to illustrate the utility of these timing results, it is worth going back to the example comparison in Section 1 between a 27-point scheme and a 125-point scheme. If a two-percent dispersion error were desired across some fixed bandwidth of interest, it can be calculated that the 125-point scheme with the optimised parameters used in [22] would require approximately 20× fewer pointwise updates than a 27-point scheme.[1] This outweighs the increases in CTPNs that would be seen on the K20 GPU going from a 27-point stencil to a 125-point stencil, which are 3.6× and 6.6×, respectively for single and double precision. One can carry out this type of calculation for other stencil sizes and associated schemes after analysing their dispersion errors, following [22].

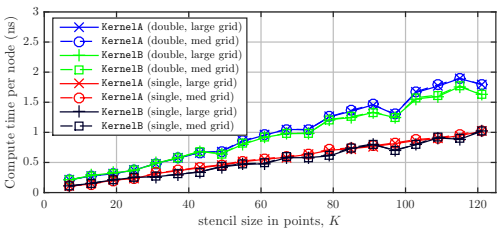
4.2. Memory throughput

In order to explain some of the timings observed, it is worth looking deeper into the many kernel metrics that can be obtained from Nvidia's CUDA profiler, and in particular, a subset relating to

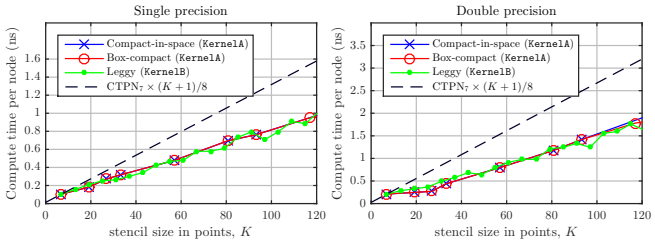
[1] The 27-point scheme would need to be oversampled by $85/40 \approx 2.1$ times that of the 125-point scheme and $2.1^4 \approx 20$.



(a) Compact stencils, double and single precision, large and medium grid sizes, using `KernelA`. Right: Plot on left for $K < 140$ in order to show detail.



(b) Leggy stencils, double and single precision, large and medium grid sizes, and the two kernel implementations.



(c) Compact and leggy stencils in single (left) and double (right) precision.

Figure 7: Average compute times per node for compact-in-space, box-compact, and leggy stencils as a function of stencil size K, on Nvidia Tesla K20 GPU card in double and single precision, and for large and medium grid sizes. In Fig. 7(c) is the line $\text{CTPN}_7 \times (K+1)/8$, where "$\text{CTPN}_7$" is the compute time per node for the 7-point stencil in either double or single precision. This line is what would be expected if compute-time increases were equal to the increases in FP instructions or memory reads required.

memory throughput. As an aside, it is worth mentioning that the following analysis will be more informative and rigorous than an approach to performance analysis recently proposed in [17, Section V], which does not make direct use of profiler metrics.[2]

Before continuing, it is worth explaining the ideal memory-reading scenario (without utilising shared memory) within this Kepler GPU memory architecture. Each stencil operation requires $K+1$ memory reads, but neighbouring points share many of these values. In the ideal scenario, each thread would read `u[cp]` from global memory (in coalesced reads) via the L2->L1 pipeline and `uz[cp]` from global memory via the L2->texture pipeline, and the remaining $K-1$ grid points would be read directly from texture cache. Once the calculation is finished, the final value would be written via L1->L2 back to global memory to be stored at `u[cp]`. So in the ideal case, the global memory read throughput would be equal to two times the global memory write throughput. However, the texture cache is not unlimited in size (48 KB per SMX), so in practice, grid values stored in texture cache that have not been completely utilised may be overwritten and reloaded into the texture cache, fetched either directly from L2 cache or from global memory, at a later stage in the stream. Thus, cache hit ratios will be lower than the ideal 100%.

With that said, Fig. 8 displays the texture cache (read-only) throughput, the L2 cache read throughput, the global memory read and write throughputs, and the global memory bandwidth used (global read + global write), for the compact-in-space (using `KernelA`) and leggy stencils (using `KernelB`) on the large grid sizes. It can be seen in Fig. 8 that the compact and leggy stencils use up nearly all of the available texture cache (read-only data cache) bandwidth (approximately 1.1 TB/s) in single precision, and a large proportion of the available texture cache bandwidth in double precision. However, the global memory read throughput is greater than two times the global memory write throughput, so the ideal scenario is not achieved. For the compact stencils, the global memory read-to-write ratio is approximately three for $K \leq 27$, and is approximately four for $K > 27$. For the leggy stencils, it is much higher, starting around three and going as high as 25, indicating a low efficiency in terms of texture cache usage. The texture throughput in double precision is about 60% that in single precision, which is a consequence of the fact that the texture cache can store more single-precision values than it can double-precision values.

The texture cache hit ratios and L2 hit ratios (from texture reads) are plotted in Fig. 9 (for the large grid sizes). It can be seen from Fig. 9 that the cache hit ratios are higher overall for the compact stencils than the leggy ones, and the L2 hit ratios (from texture reads) increase when the texture cache hit ratios decrease, indicating that many of the data not found in the texture cache were found directly in the L2 cache without having to read from global memory. On the other hand, for the leggy stencils the texture hit ratios are much lower and the L2 hit ratios (from texture reads) are also low, which means that global memory was read more often. Thus, we can conclude that the compact stencils make better use of the texture cache than the leggy stencils, and the leggy stencils are able to achieve similar performance (in terms of CTPNs) due to an increased global memory read throughput.

Aside from the 7-point scheme, the global memory bandwidth usage for compact stencils is a small fraction of the maximum theoretical bandwidth available (208 GB/s). Thus, the implemented compact stencil operations are bound by texture cache bandwidth, rather than by global memory bandwidth (aside from the 7-point scheme). On the other hand, the implemented leggy stencil operations are bound by both texture and global memory bandwidths,

[2] We note the performance metric recently proposed in [17, Section V] is based on an assumption that respective times spent on FLOPs and data transfers are additive, but this assumption does not necessarily hold for a GPU device. More importantly, the proposed metric assumes that all data transfers make use of global memory bandwidth, however, as will be seen here, data transfers in the GPU are more nuanced for such schemes—one must consider the various levels of caches that connect the global memory to SMX registers and the potential for stencil operations to make use of these caches.

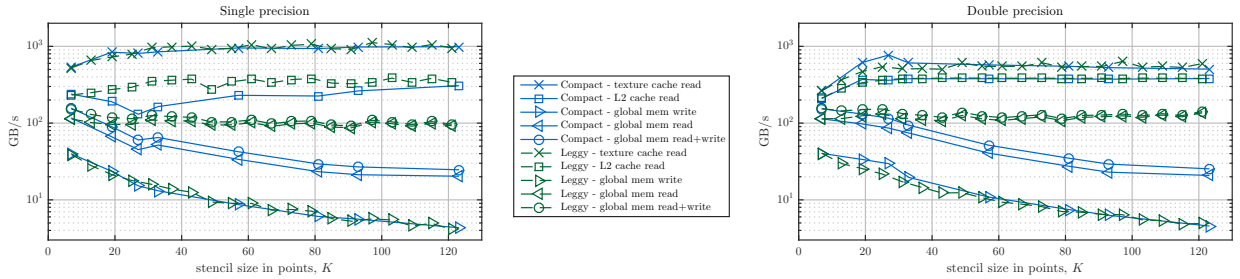


Figure 8: Memory throughput for compact-in-space stencils (using `KernelA`) and leggy stencils (using `KernelB`) as a function of stencil size in points for $K \leq 125$. Left: single precision, right: double precision.

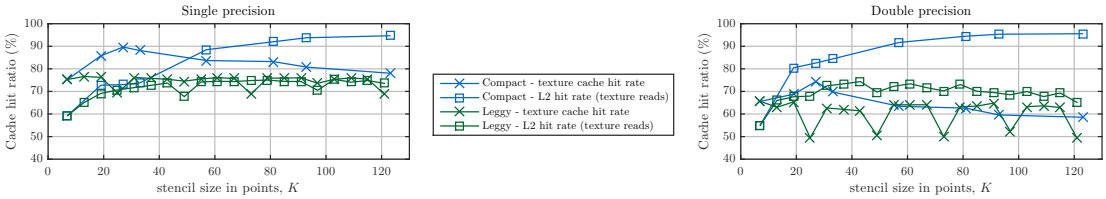


Figure 9: Texture and L2 cache hit-rates for compact-in-space stencils (using `KernelA`) and leggy stencils (using `KernelB`) as a function of stencil size in points for $K \leq 125$.

since they use a large majority of both.

Although not shown in any figures, it is worth pointing out that in all cases, occupancy rates were high (above 85%) and multiprocessor activity was nearly 100%. Also, the number of FLOPs and FLOP instructions obtained from profiler metrics agreed with the FLOP counts provided in Section 3.1. As such the FLOPS usage was at most 8% and 15% of the theoretical peak FLOPS in single and double precision respectively. Clearly then, the bottleneck is data movement rather than compute. It is also worth mentioning that the kernels were also tested without the use of `__cont restrict`, utilising the L2->L1 pipeline, and performance was significantly worse.

4.3. Comparisons with shared memory approaches

While shared memory approaches will not be directly investigated in this study, a brief comparison will be made with recently published results. In [16], leggy stencils up to $M = 8$ and a 27-point scheme were implemented on a Nvidia GTX 670 GPU device[3] following the shared memory approach of Micikevicius [5] for leggy stencils, which was also adapted to the 27-point compact stencil, although kernel codes were not provided. Texture cache was also employed, presumably to load values into shared memory.

The testing procedure in [16] was essentially the same as the one that has been used here, except that the dimensions of the box-domain were specified in metres, so grid sizes varied for each scheme with respect to specific choices of c, T, X appropriate for the schemes. A GTX 670 device was not available for a direct comparison, but a closely-related, and lower performing, GTX 660Ti GPU device was on hand. The GTX 660Ti has the same underlying chip (GK104) and the same compute specifications as the GTX 670 (clock speeds, peak FLOPS, etc.), but the GTX 660Ti is handicapped to 75% of the global memory bandwidth and L2 cache available in the GTX 670, due to one of four memory controllers on the GK104 chip being disabled.

[3] Although the GTX 670 device was not explicitly named in [16], its use was confirmed through correspondence with the authors.

Table 1: Timings for "medium-sized room" tests conducted on GTX 670 with shared memory approach in [16] alongside timings obtained here on GTX 660Ti using a maximum threading approach and pure texture fetching.

stencil	grid dims	GTX 670 [16] CTPN (ns)	GTX 660Ti CTPN (ns)	ratio
leggy, $M = 1$	370×259×148	0.239	0.162	1.47
leggy, $M = 2$	321×224×128	0.270	0.256	1.06
leggy, $M = 3$	301×211×120	0.318	0.271	1.18
leggy, $M = 4$	290×203×116	0.347	0.366	0.95
leggy, $M = 5$	283×198×113	0.424	0.375	1.13
leggy, $M = 6$	278×195×111	0.471	0.441	1.07
leggy, $M = 7$	275×192×110	0.490	0.574	0.85
leggy, $M = 8$	272×190×108	0.430	0.567	0.76
27-point compact	642×449×256	0.725	0.294	2.46

With the GTX 660Ti, the `__const restrict` qualifiers do not enable the read-only data cache loading behaviour as they do with the Tesla K20 device [34], since the compute capability of the GTX 660Ti is only version 3.0. As such, the kernels were modified to make use of texture cache bindings. The block size chosen for the GTX 660Ti was 32×8×4. Using `KernelA`, we repeated the "medium-sized room" and "large room" tests with the grid dimensions listed in [16, Table 3]. The timings obtained on the GTX 660Ti for the medium-sized room are presented in Table 1, alongside the results from [16], where the CTPNs were calculated from the grid sizes and the "frames per second" values in [16, Table 3]. Ratios between timings on the two cards are also provided for comparison purposes. For brevity, the "large room" performance results are left out because they do not vary significantly from Table 1 in terms of compute times per node, both with the GTX 670 timings reported in [16] and with the GTX 660Ti.

From Table 1 it can be seen that in all cases, the performance obtained on the GTX 660Ti is at least 75% of the performance obtained on the GTX 670 in [16], and for the most part, better performance was achieved on the GTX 660Ti without the use of shared memory. Most notably, the last row in Table 1 demonstrates a significant speed-up (2.46×) for the 27-point compact stencil on the GTX 660Ti in comparison to the GTX 670 timings reported in [16]. This suggests either that the shared memory implementation for

the 27-point scheme in [16] was suboptimal, or that the 27-point scheme cannot make efficient use of shared memory on these cards.

5. CONCLUSIONS AND FINAL REMARKS

In this paper, compact stencils on the 3-D cubic lattice and associated two-step finite difference schemes were analysed on GPU devices, as well as standard high-order (leggy) stencil two-step schemes. It was found that GPU performance, measured in terms of compute times per node, scaled linearly with stencil size, but generally the increases were less than what would be expected from increases in operation count over smaller stencils. It was found that data movement, rather than compute, was the bottleneck, and as such, the performance obtained can be attributed to the effects of the L2 and texture caches on the Tesla K20 card. Performance, in terms of compute times per node, varied little with respect to stencil shape for a fixed stencil size.

While overall performance did not vary with shape, the usage of memory bandwidths varied significantly between compact and leggy stencils. Compact stencils were found to make more efficient use of texture cache (higher hit rates) than leggy stencils, thus requiring fewer reads from global memory. The leggy stencil schemes required a significant portion of global memory bandwidth in order to achieve similar performance as compact stencils of similar size in points. Accordingly, it was seen that leggy stencils had relatively low L2 and texture cache hit rates.

Finally, a brief comparison was made with recently reported GPU timing results that used shared memory approaches for leggy stencil schemes ($M \leq 8$) and a 27-point compact stencil scheme. It was found that similar or better performance to the GTX 670 results reported in [16] could be obtained for most of the leggy stencil schemes considered, using a GTX 660Ti (with only 75% of the memory bandwidth and L2 cache of the GTX 670) and without the use of shared memory. For the compact 27-point scheme, a speed-up of 2.46$\times$ was achieved on the GTX 660Ti card over the reported timings from a GTX 670.

In future work, shared memory approaches could be investigated in the context of larger compact stencils, since shared memory presents many opportunities for performance. However, it is clear from these results that good performance can be obtained through pure texture fetching. It is worth recalling that the texture cache is easily accessed in newer generation Nvidia cards through the `__const restrict` qualifiers (or `__ldg()` intrinsics), which enables simpler kernel codes than those that would make use of shared memory (e.g., `KernelA` has, essentially, ten lines of code and `KernelB` has thirteen; compare to, e.g., [5, Appendix A]).

Other avenues for future work, in terms of GPU implementations, include 2-D slicing approaches, varying the order of offsets in the loop, unrolling stencil operations, and L1 caching of global memory on more recent cards than the K20. Finally, we note that maintaining good GPU performance overall with boundary condition for such schemes, which constitutes an important open problem, will present further challenges.

The C/CUDA codes used in this study are available at: `http://www2.ph.ed.ac.uk/~s1164563/dafx15`.

6. REFERENCES

[1] P. M. Morse and K. U. Ingard, *Theoretical acoustics*. Princeton University Press, 1968.

[2] R. D. Richtmyer and K. W. Morton, *Difference methods for initial-value problems*, 1st ed. Interscience Publishers, 1957.

[3] M. A. Dablain, "The application of high-order differencing to the scalar wave equation," *Geophysics*, vol. 51, no. 1, pp. 54–66, 1986.

[4] D. Botteldooren, "Acoustical finite-difference time-domain simulation in a quasi-Cartesian grid," *JASA*, vol. 95, pp. 2313–2319, 1994.

[5] P. Micikevicius, "3D finite difference computation on GPUs using CUDA," in *Proc. of 2nd Workshop on GPGPU*. ACM, 2009, pp. 79–84.

[6] V. Etienne, T. Tonellot, P. Thierry, V. Berthoumieux, and C. Andreolli, "Optimization of the seismic modeling with the time-domain finite-difference method," in *SEG Annual Meeting*, 2014.

[7] J. Shragge, "Solving the 3D acoustic wave equation on generalized structured meshes: A finite-difference time-domain approach," *Geophysics*, vol. 79, no. 6, pp. T363–T378, 2014.

[8] N. Röber, M. Spindler, and M. Masuch, "Waveguide-based room acoustics through graphics hardware," in *Proc. Int. Computer Music Conf.*, 2006.

[9] J. Sheaffer and B. Fazenda, "FDTD/K-DWM simulation of 3D room acoustics on general purpose graphics hardware using compute unified device architecture (CUDA)," in *Proc. Institute of Acoustics*, vol. 32, no. 5, 2010.

[10] L. Savioja, "Real-time 3D finite-difference time-domain simulation of low-and mid-frequency room acoustics," in *Proc. Digital Audio Effects (DAFx)*, vol. 1, Graz, Austria, 2010, p. 75.

[11] R. Mehra, N. Raghuvanshi, L. Savioja, M. C. Lin, and D. Manocha, "An efficient GPU-based time domain solver for the acoustic wave equation," *Applied Acoustics*, vol. 73, no. 2, pp. 83–94, 2012.

[12] J. J. López, D. Carnicero, N. Ferrando, and J. Escolano, "Parallelization of the finite-difference time-domain method for room acoustics modelling based on CUDA," *Mathematical and Computer Modelling*, vol. 57, no. 7, pp. 1822–1831, 2013.

[13] J. Saarelma, "Finite-difference time-domain solver for room acoustics using graphics processing units," Master's thesis, Aalto University, 2013.

[14] B. Hamilton and C. J. Webb, "Room acoustics modelling using GPU-accelerated finite difference and finite volume methods on a face-centered cubic grid," in *Proc. Digital Audio Effects (DAFx)*, Maynooth, Ireland, Sep. 2013, pp. 336–343.

[15] B. Hamilton, S. Bilbao, and C. J. Webb, "Revisiting implicit finite difference schemes for 3-D room acoustics simulations on GPU," in *Proc. Digital Audio Effects (DAFx)*, Erlangen, Germany, 2014.

[16] J. van Mourik and D. Murphy, "Explicit higher-order FDTD schemes for 3D room acoustic simulation," *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 2003–2011, 2014.

[17] C. Spa, A. Rey, and E. Hernandez, "A GPU implementation of an explicit compact FDTD algorithm with a digital impedance filter for room acoustics applications," *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 23, no. 8, pp. 1368–1380, 2015.

[18] L. Savioja and V. Valimaki, "Interpolated 3-D digital waveguide mesh with frequency warping," in *Proc. IEEE ICASSP*, vol. 5. IEEE, 2001, pp. 3345–3348.

[19] S. Bilbao, "Wave and scattering methods for the numerical integration of partial differential equations," Ph.D. thesis, Stanford University, 2001.

[20] K. Kowalczyk, "Boundary and medium modelling using compact finite difference schemes in simulations of room acoustics for audio and architectural design applications," Ph.D. dissertation, Queen's University Belfast, 2008.

[21] G. Cohen, *Higher-order numerical methods for transient wave equations*. Springer-Verlag, 2002.

[22] S. Bilbao, "Optimized FDTD schemes for 3-D acoustic wave propagation," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1658–1663, 2012.

[23] D. Unat, X. Cai, and S. B. Baden, "Mint: realizing CUDA performance in 3D stencil methods with annotated C," in *Proc. Int. Conf. Supercomputing*. ACM, 2011, pp. 214–224.

[24] M. Krotkiewski and M. Dabrowski, "Efficient 3D stencil computations using CUDA," *Parallel Computing*, vol. 39, no. 10, pp. 533–548, 2013.

[25] A. Vizitiu, L. Itu, C. Nita, and C. Suciu, "Optimized three-dimensional stencil computation on Fermi and Kepler GPUs," in *IEEE HPEC*, 2014, pp. 1–6.

[26] S. Leclaire, M. El-Hachem, J.-Y. Trépanier, and M. Reggio, "High order spatial generalization of 2D and 3D isotropic discrete gradient operators with fast evaluation on GPUs," *J. Sci. Comp.*, vol. 59, no. 3, pp. 545–573, 2014.

[27] B. Fornberg, "Generation of finite difference formulas on arbitrarily spaced grids," *Mathematics of Computation*, vol. 51, no. 184, pp. 699–706, 1988.

[28] W. F. Spotz and G. F. Carey, "A high-order compact formulation for the 3D Poisson equation," *Num. Methods for PDEs*, vol. 12, no. 2, pp. 235–243, 1996.

[29] M. Patra and M. Karttunen, "Stencils with isotropic discretization error for differential operators," *Num. Methods for PDEs*, vol. 22, no. 4, pp. 936–953, 2006.

[30] J. Conway and N. J. A. Sloane, *Sphere packings, lattices and groups*, 3rd ed. Springer-Verlag, 1991.

[31] S. Bilbao and B. Hamilton, "Construction and optimization techniques for high order schemes for the two-dimensional wave equation," in *Proc. Int. Cong. Acoustics (ICA)*, Montréal, Canada, 2013.

[32] J. D. McCalpin, "STREAM: Sustainable memory bandwidth in high performance computers," University of Virginia, Tech. Rep., 2007. [Online]. Available: http://www.cs.virginia.edu/stream/

[33] C. J. Webb, "Parallel computation techniques for virtual acoustics and physical modelling synthesis," Ph.D. thesis, University of Edinburgh, 2014.

[34] NVIDIA Corporation, *CUDA C Programming Guide*. PG-02829-001_v7.0, Mar. 2015.

VOWEL CONVERSION BY PHONETIC SEGMENTATION

Carlos de Obaldía, Udo Zölzer
Helmut Schmidt University
Hamburg, Germany
deobaldia@hsu-hh.de

ABSTRACT

In this paper a system for vowel conversion between different speakers using short-time speech segments is presented. The input speech signal is segmented into period-length speech segments whose fundamental frequency and first two formants are used to find the perceivable vowel-quality. These segments are used to represent a voiced phoneme, i.e. a vowel. The approach relies on pitch-synchronous analysis and uses a modified PSOLA technique for concatenation of the vowel segments. Vowel conversion between speakers is achieved by exchanging the phonetic constituents of a source speaker's speech waveform in voiced regions of speech whilst preserving prosodic features of the source speaker, thus introducing a method for phonetic segmentation, mapping, and reconstruction of vowels.

1. INTRODUCTION

Applications for segment-based speech analysis and synthesis in engineering and scientific perspectives have been applied for speech morphing algorithms, speech synthesis and coding, and text-to-speech (TTS) systems just to mention a few. In voice conversion (VC) systems, a *source* speaker's waveform is converted so that it perceptually resembles the voice of a *target* speaker whilst retaining linguistic information [1]. Techniques which have a broad success in speech and voice conversion, usually use signal segments as the unit to concatenate during re-synthesis [2].

In linguistics, speech can be modeled as a series of phonemes. Phonolgy is a branch of linguistics which provides information on the nature of these abstract units. Phonemes follow a particular order to describe words and utterances. Psycholinguistic models of speech production can be abstracted to identify two levels or stages of speech processing: the word (lemma) and the phonological level [3]. In the lemma representation, abstract word properties such as grammatical features are coded. The phonological information, i.e. the form of a word, is coded at the next level of processing [3].

Most of current speech synthesis systems use diphones as a basic selection unit for speech concatenation at the back-end [4, 5]. Diphones are generally constituted by parts of two or more phonetic elements, and contain transitions between phones [6]. However, segments of the speech waveform can also be abstracted to a phonetic level and concatenated attaining to a particular prosody to reproduce a series of phones which form a phoneme [5]. Although using diphones during synthesis may lead to a smooth speech waveform, the use of phone-based analysis and synthesis methods for speech corpora can introduce advantages in context, speed, and consistency [7].

In this paper a phoneme-based vowel conversion system is proposed to convert voiced regions of speech from one speaker (a source speaker) so that it sounds like it was pronounced by another speaker (the target speaker) by using phonetic trajectories instead of supra-segmental models and using signal snippets instead of spectral morphing. Speaker conversion is achieved by synthesis-by-rule of the target speaker's vowel segments using the source speaker's tonality and intensity.

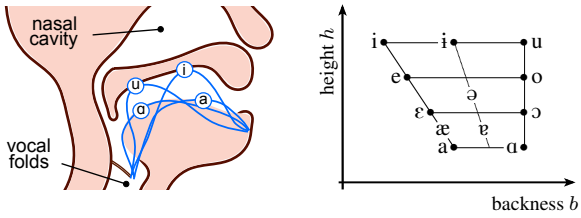


Figure 1: *Position of the tongue at the time of utterance and the cardinal vowel diagram. The x-axis corresponds to backness b and the y-axis to height h of the tongue at the time of utterance.*

Perception in humans of a different speaker quality, or timbre, between different speakers is based on differences on the laryngeal and vocal tract resonances [8]. This work focuses only on voiced parts-of-speech, considering that the vocal-tract filter configuration for particular speakers is enough, at least at the phonological level, for a perceptually viable voice conversion.

Since the work is based on the phonetic level of speech production [3], and speech rate is known to contain linguistic information [9], a language independent synthesizer for different speech qualities, for example, could be constructed using the presented method.

2. PROPOSED SYSTEM

A speech segment is any discrete unit that can be identified in a stream of speech and is composed of a series of phonemes. A phoneme is the smallest phonetic contrastive linguistic unit in a language which may bring about a change of meaning. A logical distinction would be to categorize phonemes according to its acoustic nature, for example into voiced and non-voiced sounds.

Voiced phonemes such as vowels are of a periodic nature. Thus a *short-time speech segment* (STSS) can be extracted from a voiced speech region and characterized as an atomic unit of speech (i.e, a phone) based solely on the information of a particular fundamental period.

Voiced speech is also modeled as the response of the vocal tract to a source excitation, which is generally produced by the opening and closing of the glottal vents in the trachea [10]. The characteristics of such a response are achieved by the amplification of the frequency components of the excitation in the vocal

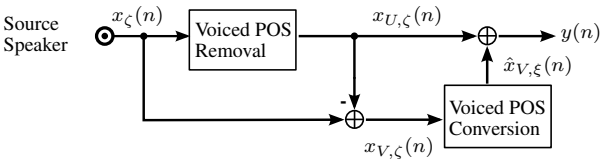


Figure 2: *Block diagram for the applied voice conversion. The voiced parts-of-speech of an input signal from a source speaker ζ are exchanged with a synthesized waveform using a series of acoustic units from a target speaker ξ.*

tract. The main resonance frequencies of the vocal tract are called *formant frequencies* and mostly vary according to the perceived vowel quality.

The International Phonetic Association (IPA) issues a phonetic characterization of vowels according to the position of the tongue in the oral cavity at the time of utterance. Pfitzinger [11] introduced a method which maps the acoustic characteristics of vowels to the cardinal vowel diagram of Fig. 1. The method uses the fundamental frequency f_0 and the first two formants F_1, F_2 as parameters such that

$$h = 3.122\log(f_0) - 8.841\log(F_1) + 44.16 \quad (1)$$

$$b = 1.782\log(F_1) - 8.617\log(F_2) + 58.29, \quad (2)$$

where b represents the perceived *backness* and h the perceived *height* of the tongue at the time of utterance according to the phone position on the IPA vowel diagram of Fig. 1. Backness is restricted between 0 and 12 and height between 0 and 10 to overcome the end-of-scale effect [11].

These coordinates will correspond to the Cartesian location of the STSS in the vowel-space of Fig. 1, and are used as keys for selecting the units to concatenate out of a database.

Synthesis of speech based on the concatenation of short-time signals in the time domain can be achieved with an acceptable naturalness using PSOLA [12], where the signal to concatenate is overlapped at a specific position at each pitch mark k and added to the output signal [13]. PSOLA is used in several speech synthesizers in conjunction with a sinusoidal speech production model which conserves filter-source model theory [14, 15], and in sample-based granular synthesis, uses prerecorded segments with different voice qualities [12].

To exchange voiced parts-of-speech (POS) in the waveform, the signal is decomposed as in Fig. 2. An input speech signal from a *source* speaker $x_\zeta(n)$ is divided into a signal containing voiced POS $x_{V,\zeta}(n)$ and a signal with other non-voiced (or non-tonal) POS $x_{U,\zeta}(n)$, such that

$$x(n)_\zeta = x_{V,\zeta}(n) + x_{U,\zeta}(n). \quad (3)$$

The signal is analyzed to determine its vowel-quality every period. Sections of $x_\zeta(n)$ are then re-synthesized using previously recorded grains (units) from a target speaker. The voiced region, $x_{V,\zeta}(n)$, is then exchanged with a synthesized voiced region of a *target* speaker $\hat{x}_{V,\xi}(n)$, such to obtain the converted speech signal

$$y(n) = \hat{x}_{V,\xi}(n) + x_{U,\zeta}(n). \quad (4)$$

The block diagram of Fig. 3 represents the proposed voiced POS conversion system. During analysis, a target speaker's voiced POS are each segmented into J segments $s_j(n)$. The acoustic features of each STSS $s_j(n)$, the fundamental frequency f_{0_j}, and the first and second formants (F_{1_j}, F_{2_j}) of $s_j(n)$ are extracted and mapped to the perceived backness and height coordinates of the vowel-space of Fig. 1. The same procedure is used to extract the perceived backness and height measures at each pitch mark of the source speaker's signal.

Synthesis is performed using PSOLA with the time envelope and the fundamental frequency vector of the source speaker, selecting a STSS, or a series of STSS from the database for concatenation. The converted voiced POS is reconstructed using an adapted PSOLA approach at pitch marks of the source speaker in voiced regions and modulated by the amplitude of the incoming source speech waveform in order to maintain signal intensity.

3. SEGMENTATION AND MAPPING

To generate the set of units of the target speaker to concatenate, an incoming source signal is analyzed to detect and save the STSS for vowel reconstruction. Acoustic characteristics for each extracted segment of a vowel are thus analyzed, indexed and mapped to the vowel-space.

According to the source filter model, a speech signal can be represented by

$$s(n) = h_{VT}(n) * x_{EX}(n), \quad (5)$$

where $h_{VT}(n)$ describes the impulse response of the vocal tract, $x_{EX}(n)$ the excitation signal coming from the lungs and $s(n)$ is the speech waveform radiated at the lips [5]. If the excitation $x_{EX}(n)$ is an impulse train, the spectral contour of a speech segment will approximate the frequency response of the excitation filter $h_{VT}(n)$.

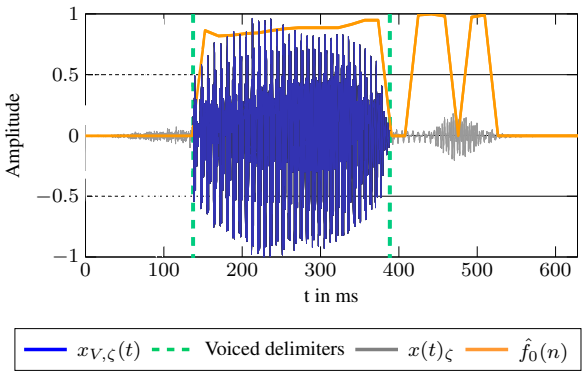


Figure 4: *Voiced - Unvoiced segmentation.*

3.1. Voiced and unvoiced detection

Voiced regions are analyzed using the fundamental frequency contour on the output of a YIN pitch tracker implemented as in [16] using a minimum pitch $f_{0_{min}} = 50$ Hz and a hop size of 256 samples.

Delimiters for voiced regions are set when the fundamental frequency surpasses 10% of the mean maximum frequency of a signal snippet. This is done to take in consideration the moment when a voiced tone starts to overcome the unvoiced POS [17]. For

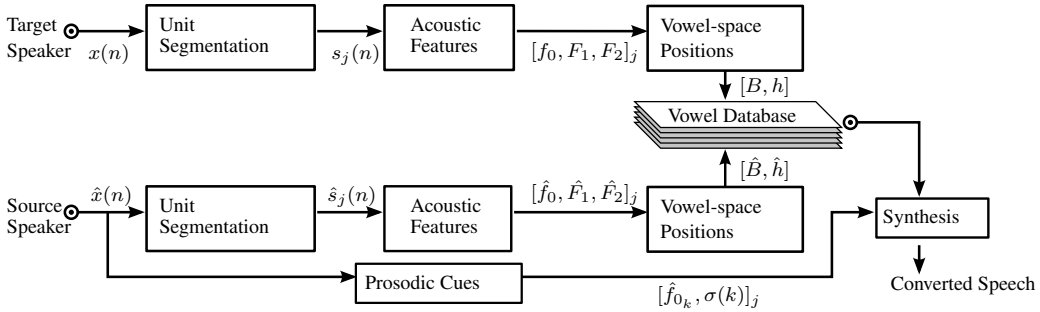


Figure 3: *Block diagram representation of the proposed vowel conversion system. Voiced speech segments of one period length are analyzed and exchanged with those of a target speaker.*

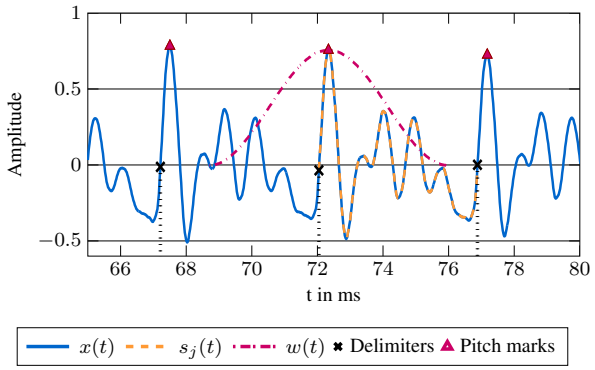


Figure 5: *Excerpt of a voiced segment (/a/). The dotted line shows the synthesis window $w(t)$ for samples in the database. Crosses represent the delimiters of the short-time speech segment $s_j(t)$. The window is centered at the local peak of each segment, i.e. the pitch marks.*

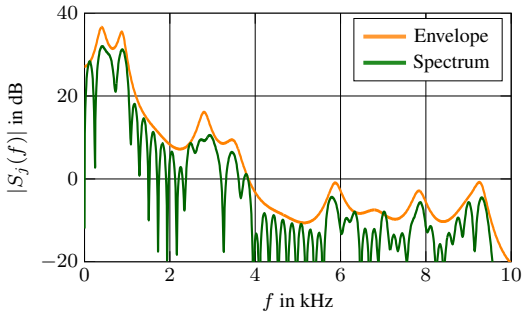


Figure 6: *LP spectral envelope and Fourier spectrum for a short-time speech segment of an /e/. The frequency positions of the envelope correspond with the formants.*

our voice conversion application, regions shorter than 60 ms are ignored since the timbre of shorter sounds is not perceptually recognizable [18]. Fig. 4 shows the normalized $\hat{f}_0(n)$ and the voiced POS $x_{V,\zeta}(n)$ to be exchanged on the utterance /head/ of a male speaker from the Hillenbrand vowel database [19].

3.2. Segmentation and unit extraction

Phonetic segmentation is performed pitch-synchronously on the signal by fragmenting each voiced region $x_V(n)$ of the target speakers into a STSS $s_j(n)$ set for $j \in [1, J]$.

The fundamental frequency vector $f_0(n)$ at the output of the pitch tracker gives the cues for segmentation. Local maxima every pitch period length $T_0(n)$ are identified as pitch marks on the voiced speech region. Maxima are searched for every $\frac{3}{2}T_0(n)$ samples, so that no short-time speech segments overlap.

The *first zero-crossing* before such maxima is regarded as a delimiter of the STSS. Segments are indexed at consecutive delimiters, and form the units for concatenation during synthesis. Fig. 5 shows the segmentation.

3.3. Acoustic features

The acoustic features for each short-time speech segment $s_j(n)$ are the first two formants ($F1, F2$) and the fundamental frequency f_0, which are used to calculate the perceived vowel-quality based on the backness and height coordinates on the vowel-space of Fig. 1.

After extracting the fundamental frequency and finding the delimiters of each STSS $s_j(n)$, Linear Predictive Coding (LPC) [5, 20] is used to approximate the spectral envelope and the formant composition of the segment. The order of the LPC analysis filter is selected such that there is one pole for each kHz of the sampling frequency, in this case $p = 44$. The resulting filter coefficients are transformed to the frequency domain generating a spectral envelope approximation for the STSS.

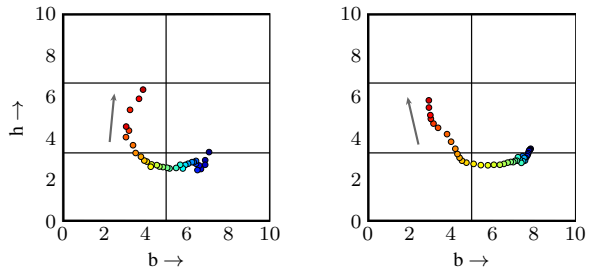
A peak-picking technique is then used to obtain the frequency positions of the first two peaks in the envelope.

In Fig. 6 the spectral envelope of an /e/ show the formant composition of the signal. The peaks indicate the position of the formants in the frequency domain (the first three formants appear below 3 kHz).

3.4. Vowel mapping

A mapping technique is then used to transform the acoustical characteristics of the STSS to a position in the vowel-space.

The amount of correlation which is introduced by the non-uniform Cartesian vowel coordinate system is settled by trans-



(a) m09aw.wav (b) m01aw.wav

Figure 7: *Vowelworm [22] of the utterance /hawed/ from two different male speakers taken from the Hillenbrand database [19]. Each dot represents a short-time speech segment under analysis. The arrows indicate the approximate direction of the trajectories of the STSS in the vowel-space.*

forming the vowel diagram's trapezoidal form to a rectangular space using

$$B = 10\frac{b + 0.5h - 5}{0.5h + 7.25} \tag{6}$$

as in [21]. The position of $s_j(n)$ on the vowel-space represents its perceived vowel quality, so that each extracted STSS represents a particular phone.

In order to reproduce non-isolated vowels, such as diphthongs and consonant - vowel transitions, the trajectories of the segments in the vowel-space should be considered [17]. Fig. 7 shows the perceived vowel height and backness for the STSS of two utterances of the English word /hawed/ from the Hillenbrand database [19]. The figure shows the transition between a starting STSS and an ending STSS in the (squared) vowel diagram from [21]. The perceived backness and height position of the segments also appear to change their velocity, transitioning slowly at the beginning and more abruptly towards the end of the voiced part.

3.5. Database construction

The database is based on the units required to synthesize the converted vowel waveform. In this sense, the window $w(n)$ in Fig. 5 is used for saving units in the database. Each stored unit is situated between three consecutive delimiters for each extracted $s_j(n)$. The *Hann* window $w(n)$ is centered at the maxima of $s_j(n)$ and expands over 1.5 times of the length of $s_j(n)$. The window is then multiplied with a segment $s_j^D(n)$, which is also centered at the peak of $s_j(n)$ and is of the same length as $w(n)$. Each segment $w(n)\cdot s_j^D(n)$ is saved in a dictionary along with its index j, and backness B and height h positions from (2) and (1) respectively. The length of the segments is proportional to the period of the fundamental frequency as they where extracted.

The phonetic dictionary is then conformed of a series of segments $s_j(n)$ extracted during analysis. Each series of grains, as in the vowelworm from Fig.8, are saved. The keys for each extracted unit, correspond to the perceived backness from Eq.(6) and perceived height coordinates from Eq.(**??**). The trajectories of the diphones are also saved and a time stamp for the STSS is also added as a key. The units are normalized to its peak value and saved.

The target speaker's database comprises the whole extracted segment grains. From Fig. 7, several interpretations can be extracted. The signal position on the vowel-space may describe the composition of a phoneme based solely on the use of steady-state vowel grains. The number of grains on each time position, the velocity and the start and ending positions should be considered to model the corresponding vowel. However, the pulses which correspond to steady state vowels could be modeled with a single grain as in [2].

The bi-dimensional matrix with the coordinates of the extracted segments for a speaker in the vowel dictionary is then

$$\boldsymbol{\gamma}_v = [\mathbf{B},\ \mathbf{h}]\,, \tag{7}$$

where $[\mathbf{B}, \mathbf{h}]$ is a list of backness and height coordinates for every indexed segment in each vowel group, and v is the index for a particular vowel group. This will make up the keys for identification of a particular vowel segment $s_j(n)$ in the dictionary of speaker-dependent vowel segments.

A *vowel centroid* is defined for each extracted voiced POS of the target speaker in the vowel-space such that

$$\boldsymbol{\Gamma}_v = \left[\frac{1}{J}\sum_{j=1}^{J} B_j, \frac{1}{J}\sum_{j=1}^{J} h_j\right], \tag{8}$$

which will then form the bi-dimensional matrix of vowel centroids $\boldsymbol{\Gamma}$ for each speaker in the database. These centroids are used as keys for STSS retrieval.

4. SYNTHESIS AND VOWEL RECONSTRUCTION

Each STSS $s_j(n)$ represents a particular phone whose length is only one period long and is intended to be used as a concatenation unit. However, PSOLA requires a greater synthesis window to re-generate speech without excessive artifacts [7]. The PSOLA synthesis window is thus set to 1.5 of the period length from the source speaker at each pitch mark.

Synthesis is performed by multiplying the speech waveform with a sequence of time-translated windows whose length is proportional to the local pitch period of the source speaker. The generated signal is analog to convolve an impulse train, whose impulses are located around glottal closure instants, with the steady-time response of the vocal tract at a particular utterance [13, 12].

4.1. Unit selection

At each pitch mark k, a STSS from the source speaker's waveform is retrieved. One period of the signal is extracted as described in section 3 and the main two formants of extracted segment are calculated using the LPC approximation as in section 3.3. Backness and height measures for each segment are then calculated using Eqs. (6) and (1) respectively.

The resulting $[B, h]_k$ coordinates at the time instant k of the source speaker is used to map the corresponding vowel centroid in the vowel-space of the target speaker. The distance measure

$$v = \min_{v}\ D\left([B, h]_k, \boldsymbol{\Gamma}\right), \tag{9}$$

is used, where $\boldsymbol{\Gamma}$ is the matrix of vowel centroids from section 3.5 of the target speaker and v is the resulting vowel group of the target

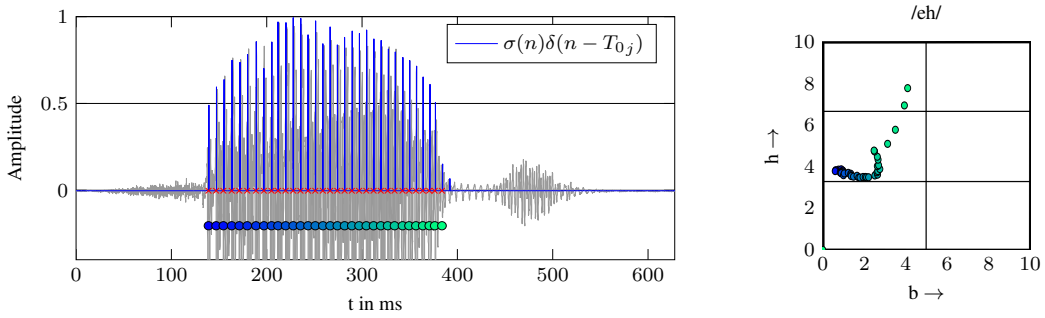


Figure 8: *Vowel segmentation and corresponding vowel-space trajectory. Each short-time speech segment (STSS) is pitch-synchronously extracted between the zero-crossings before each peak. Perceived tongue height and backness is approximated for each STSS, thus giving the positions of the segments in the vowel-space of the right. The generated impulse train (in blue) is used to give the pitch marks for re-synthesis. The impulse train is weighted with the source speaker's envelope for clarity reasons.*

speaker. The retrieved index corresponds to the vowel group of the target speaker closest to the uttered phone of the source speaker. The euclidean distance function

$$D(v)|_{[B,h]_k} = \sqrt{(B - \mathbf{\Gamma}_{(v,1)})^2 + (h - \mathbf{\Gamma}_{(v,2)})^2} \;\; \forall v, \qquad (10)$$

is applied to find the closest position in the $[\mathbf{B}, \mathbf{h}]$ vowel-space between two speakers.

A second search on the list of indexed STSS of the vowel group, $\mathbf{\Gamma}_v$ using (10) is also performed to find the index j of the STSS to concatenate. For consonant-vowel transitions the trajectories depicted in Fig. 7 are also considered in order to select the series of STSS to concatenate.

4.2. Vowel reconstruction

During voiced speech production, the excitation function $x(n)$ of (5) is represented as a sequence of impulses at the local pitch period $T_0(n)$ of the source speaker ζ. The fundamental frequency $f_0(n)$ is tracked at the source speaker as described in section 3.

Prosodic cues of the source speaker, such as intonation, intensity and vowel length are transfered to the converted signal by using the source speaker's fundamental frequency f_{0_k} and highest peak $\sigma(k)$ at each STSS to be replaced.

The synthesized waveform $\hat{x}_{V,\xi}(n)$ is the converted voiced POS and can be approximated to the synthesis model

$$\hat{x}_{V,\xi}(n) = \sigma(k) \sum_k w\left(n - kN_k\right) s_k^D(n), \qquad (11)$$

where $\sigma(k)$ is the time envelope of the source signal at the period maximum, $w(n-kN_k)$ is a variable Hann window of length N_k at each pitch event k of the source signal, and $s_k^D(n)$ is the unit segment from section 4.1 to concatenate. Discontinuities are avoided with the use of a hop size of $\frac{1}{4}N_k$.

The center of the window function is situated at the highest peak of the source's STSS, which can be approximated to its center of gravity [15].

4.3. Speech signal reconstruction

Following Eq.4 the signal is reconstructed by fading the voiced parts $\hat{x}_{V,\xi}(n)$ in the $\hat{x}_{U,\zeta}(n)$ signal after the voiced POS has been re-synthesized. This is done in an offline way. A Tukey window of length L is constructed, where L is the length of the voiced segment to fade. Followingly, the slopes of the window are mirrored to reconstruct the speech waveform. Fig. 9 shows the procedure.

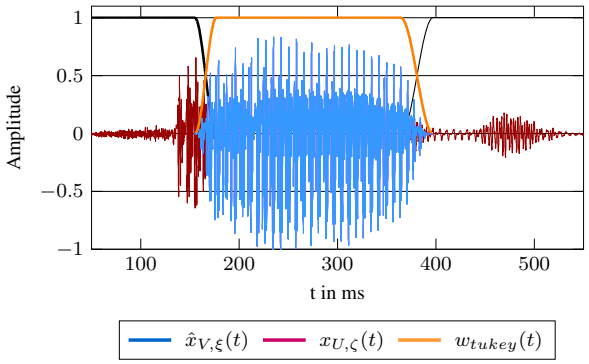


Figure 9: *Overlap add of the converted voiced part $\hat{x}_{V,\xi}(n)$ into the source signal $\hat{x}_{U,\zeta}(n)$.*

5. RESULTS AND EVALUATION

In this work, two sets of signals are considered for evaluation: Steady-state vowels and simple utterances from the Hillenbrand database [19].

To evaluate the algorithm on isolated vowels, several subjects where asked in a laboratory setting to record the same vowel order and retain intonation and intensity by following a prosodic stencil to avoid signal mismatches. The pitch lag of the source speaker is thus approximated to the length of each synthesis window, dropping the need for time stretching at the moment of concatenation. The recordings where composed of a succession of five primary vowels; /a/, /e/, /i/, /o/, /u/. The signals where thus solely composed of voiced regions of speech.

The converted speech waveform spectrum in Fig. 11c depicts that the resulting signal retains the pitch contour of the source

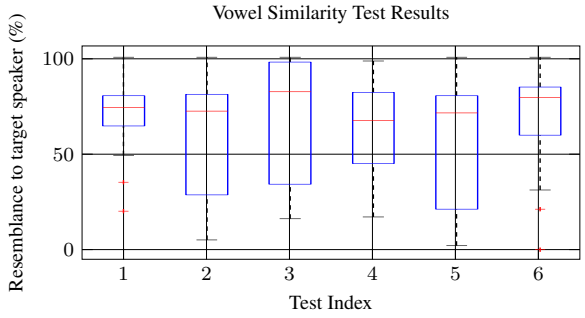


Figure 10: *Box-plot of the results for the evaluation test. The line inside each box indicates the mean of the decisions. Boxes closer to the top indicate that the identity of the speaker in the converted signal is subjectively closer to the target speaker.*

speaker of Fig. 11a while achieving the same energy for the harmonic composition of the target speaker of Fig. 11b. Vowel-quality and prosodic characteristics from the source speaker are thus transfered to a synthesized waveform with speaker-dependent characteristics of a target speaker.

These signals where also used to perform a subjective evaluation test on the capacity of the system to convert the identity of the source speaker to the target speaker. The evaluation is based on the ABX test [23, 24]. The test measures the degree of match between a source (A), a target (B), and a converted (X) signal. The tester should decide whether the converted sample (X) is perceptually closer to the identity of the source (A) or the target (B) speaker. A group of twenty five independent testers with basic to professional musical knowledge and familiar with signal processing where instructed to take the test. The test consisted on 6 different sets of a source, a target and a converted signal of the vowels between two speakers. The spectrum of the signals in the first set is shown in Fig. 11. The box-plot of Fig. 10 presents the outcome of the results. It can be noticed that the mean of the decisions is closer to the target speaker for all the signal sets.

For evaluation on consonant-vowel transitions, several samples of male and female speakers from the Hillenbrand database where taken in consideration. Figure 12 shows the spectrum of a source and a target speaker, as well as the converted speech sample. The converted signal contains re-synthesized voiced POS with the vocal tract information (timbre) of a target speaker and the unvoiced POS of a source speaker (as well as voiced regions shorter than 60 ms). The spectrum shows the same case as with isolated vowels, where the harmonic composition of the voiced regions of converted speech resemble those of the target speaker, while preserving prosodic features like phoneme duration and tonality of the source speaker.

6. CONCLUSIONS

An automatic vowel conversion system is presented which uses a short analysis and synthesis window for speech morphing and a vowel quality mapping method for period-length segments, or STSS.

A low dimensional acoustic parameter for voiced phonemes (formants, vowel-quality) is used to recognize the five vowels /a/, /e/, /i/, /o/ and /u/ according to the position in the vowel-space. The acoustic units used for analysis are the STSS of a voiced phoneme. These units contain intrinsic vowel-quality and speaker specific features, and can be characterized as a phone.

The presented algorithm takes the prosodic properties (tonality, intensity, speech rate) of the speech signal from the *source* speaker and the filter characteristics of a *target* speaker are transfered to a *source* speaker. Subjective evaluation results for isolated vowels demonstrate that the speaker's timbre is successfully transfered. Results of the synthesis of consonant-vowel transitions (Fig. 12) demonstrate that diphones (and other supra-segmental units) can be constructed using the phonetic transition visualized in the vowel-space.

The study of the transitions and characteristics of these short-time speech segments could give a better understanding of the phonological characteristics of speech and speakers, and generate a broader discussion in unit segmentation for speech analysis and synthesis. Study of the trajectories between phones in the vowel-space could be employed in real-time phonetic mapping of incoming speech, or to reduce unit inventories for speech conversion, for example.

In future work, speakers models for VC and TTS synthesis can be constructed based on the STSS positions in the vowel-space. Objective and perceptual measures for the presented speech conversion method should also be developed.

7. REFERENCES

[1] T. Nakashika, T. Takiguchi, and Y. Ariki, "Voice conversion using speaker-dependent conditional restricted boltzmann machine," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2015, no. 1, pp. 1–12, 2015.

[2] J. Bonada and X. Serra, "Synthesis of the singing voice by performance sampling and spectral models," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 67–79, 2007.

[3] G. Hickok, "The architecture of speech production and the role of the phoneme in speech processing," *Language, Cognition and Neuroscience*, vol. 29, no. 1, pp. 2–20, 2014.

[4] H. Kawahara, H. Banno, T. Irino, and P. Zolfaghari, "Algorithm amalgam: morphing waveform based methods, sinusoidal models and STRAIGHT," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), 2004*, vol. 1, pp. I–13.

[5] K. Tokuda, Y. Nankaku, T. Toda, H. Zen, J. Yamagishi, and K. Oura, "Speech Synthesis Based on Hidden Markov Models," *Proc. of the IEEE*, vol. 101, no. 5, pp. 1234–1252, May 2013.

[6] H Hon, A. Acero, X. Huang, J. Liu, and M. Plumpe, "Automatic generation of synthesis units for trainable text-to-speech systems," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), 1998*, vol. 1, pp. 293–296.

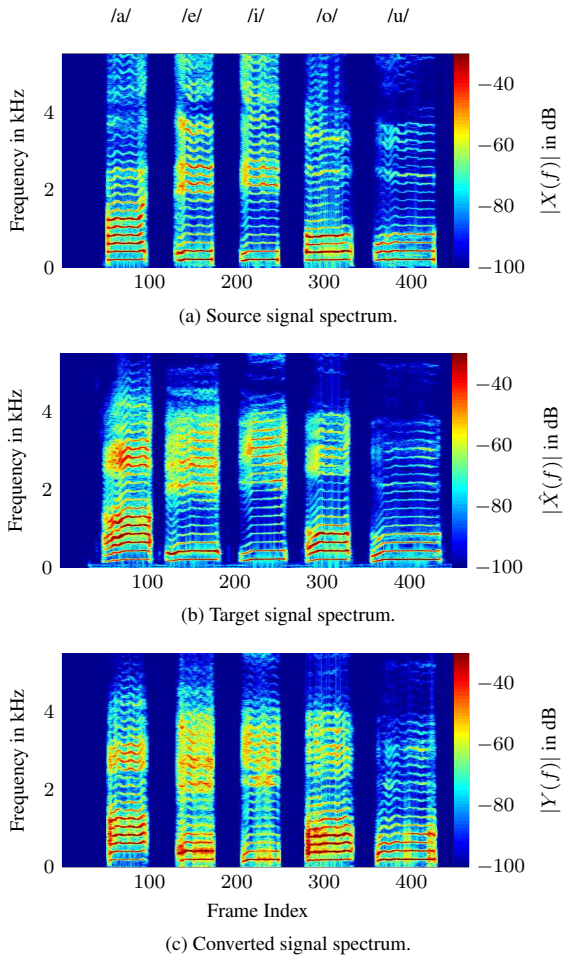
[7] R. Donovan and P. Woodland, "A hidden markov-model-based trainable speech synthesizer," *Computer speech & language*, vol. 13, no. 3, pp. 223–241, 1999.

[8] T.F. Cleveland, "Acoustic properties of voice timbre types and their influence on voice classification," *The Journal of the Acoustical Society of America*, vol. 61, no. 6, pp. 1622–1629, 1977.

[9] F. Ramus, M. Nespor, and J. Mehler, "Correlates of linguistic rhythm in the speech signal," *Cognition*, vol. 73, no. 3, pp. 265–292, 1999.

[10] F. Charpentier and M. Stella, "Diphone synthesis using an overlap-add technique for speech waveforms concatenation," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), 1986*, vol. 11, pp. 2015–2018.

[11] H.R. Pfitzinger, "Acoustic correlates of the IPA vowel diagram," *Proc. of the XVth Int. Congress of Phonetic Sciences*. Citeseer, 2003, vol. 2, pp. 1441–1444.

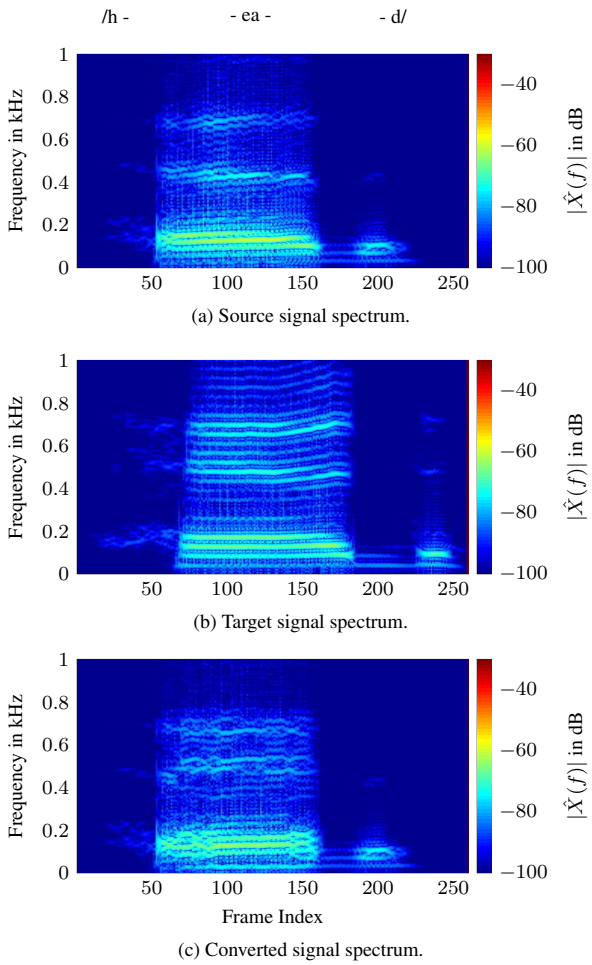


(a) Source signal spectrum.

(b) Target signal spectrum.

(c) Converted signal spectrum.

Figure 11: *Results for voice conversion for isolated vowels on continuous speech. The figure shows the Spectrum of the signals for a source speaker (a), a target speaker (b) and the converted signal (c).*



(a) Source signal spectrum.

(b) Target signal spectrum.

(c) Converted signal spectrum.

Figure 12: *Results for voice conversion on consonant-vowel trajectories. The harmonic composition of the generated waveform corresponds to those of the target speaker.*

[12] X. Sun, "Voice quality conversion in TD-PSOLA speech synthesis," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), 2000*, vol. 2, pp. II953–II956.

[13] C. Hamon, E. Mouline, and F. Charpentier, "A diphone synthesis system based on time-domain prosodic modifications of speech," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), 1989*, pp. 238–241.

[14] A. Syrdal, Y. Stylianou, L. Garrison, A. Conkie, and J. Schroeter, "TD-PSOLA versus harmonic plus noise model in diphone based speech synthesis," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), 1998.*, vol. 1, pp. 273–276.

[15] Y. Stylianou, "Applying the harmonic plus noise model in concatenative speech synthesis," *IEEE Tran. on Speech and Audio Processing,* , vol. 9, no. 1, pp. 21–29, 2001.

[16] U. Zölzer, *DAFX digital audio effects*, John Wiley & Sons, 2011.

[17] P. Birkholz, "Modeling consonant-vowel coarticulation for articulatory speech synthesis," *PloS one*, vol. 8, no. 4, 2013.

[18] S.Z.K. Khine, T.L. Nwe, and H. Li, "Exploring perceptual based timbre feature for singer identification," *Computer Music Modeling and Retrieval*, Springer, pp. 159–171, 2008.

[19] J. Hillenbrand, L.A. Getty, M.J. Clark, and K. Wheeler, "Acoustic characteristics of american english vowels," *The Journal of the Acoustical society of America*, vol. 97, no. 5, pp. 3099–3111, 1995.

[20] S. McCandless, "An algorithm for automatic formant extraction using linear prediction spectra," *IEEE Tran. on Acoustics, Speech and Signal Processing*, vol. 22, no. 2, pp. 135–141, 1974.

[21] H.R. Pfitzinger, "Towards functional modelling of relationships between the acoustics and perception of vowels," *ZAS papers in Linguistics*, vol. 40, pp. 133–144, 2005.

[22] H. Frostel, A. Arzt, and G. Widmer, "The vowel worm: Real-time mapping and visualisation of sung vowels in music," *Proc. of the 8th Sound and Music Computing Conference*, pp. 214–219, 2011.

[23] J. Kreiman and G. Papcun, "Comparing discrimination and recognition of unfamiliar voices," *Speech Communication*, vol. 10, no. 3, pp. 265–275, 1991.

[24] T. Ganchev, A. Lazaridis, I. Mporas, and N. Fakotakis, "Performance evaluation for voice conversion systems," *Text, Speech and Dialogue*. Springer, pp. 317–324, 2008.

ARTICULATORY VOCAL TRACT SYNTHESIS IN SUPERCOLLIDER

Damian T. Murphy,
AudioLab - Department of Electronics
University of York
York, UK
damian.murphy@york.ac.uk

Mátyás Jani,
Faculty of Information Technology and Bionics
Pázmány Péter Catholic University
Budapest, Hungary
jani.matyas@itk.ppke.hu

Sten Ternström,
Dept. of Speech, Music and Hearing,
KTH
Stockholm, Sweden
stern@kth.se

ABSTRACT

The APEX system [1] enables vocal tract articulation using a reduced set of user controllable parameters by means of Principal Component Analysis of X-ray tract data. From these articulatory profiles it is then possible to calculate cross-sectional area function data that can be used as input to a number of articulatory based speech synthesis algorithms. In this paper the Kelly-Lochbaum 1-D digital waveguide vocal tract is used, and both APEX control and synthesis engine have been implemented and tested in SuperCollider. Accurate formant synthesis and real-time control are demonstrated, although for multi-parameter speech-like articulation a more direct mapping from tract-to-synthesizer tube sections is needed. SuperCollider provides an excellent framework for the further exploration of this work.

1. INTRODUCTION

In recent years it has been possible to model highly detailed 3-D models of the vocal tract based on the capture of Magnetic Resonance Imaging data from human subjects (e.g. [2] [3] [4] [5]). This has further enabled the simulation of acoustic wave propagation within these models and the synthesis of speech, typically limited to sets of vowel sounds. The level of physiological and geometric detail in these 3-D models, and the requirements in terms of accurate spatial sampling for the resulting simulations, implies that real-time synthesis is not usually possible. However, although the vocal sounds produced can sound very natural, the requirement for high-level user control of the 3-D tract shape also causes problems, meaning that articulatory voice synthesis based on these methods is still a non-trivial problem. Hence there is still interest in real-time vocal tract models based on 1-D and 2-D simulation methods using more established techniques such as the Kelly-Lochbaum transmission line [6], or digital waveguide [7], approach, particularly as their reduced complexity offers additional scope for high-level control. In particular, recent work has explored how 2-D models of the vocal tract might be optimised to give results closer to that of more complex, and computationally expensive 3-D simulations [8].

In [9] real-time dynamic articulation of a 2-D profile of the vocal tract was enabled through the use of a dynamically varying digital waveguide mesh based on impedance contour maps. Articulatory control of this model was later offered via the APEX system, a tool that can be used to synthesize sound and generate articulatory voice related parameters, based on the positioning of lips, tongue tip, tongue body, jaw opening and larynx height, all mapped from X-ray data [1]. In this case, APEX generates vocal tract cross-sectional area function information that provides control input to the 2-D dynamic waveguide mesh model [10].

This paper revisits the potential offered by the APEX system for enabling real-time articulatory control of a physical model of the vocal tract for sound synthesis. Whereas APEX was originally written for the legacy Windows operating system, it has now been realised using the SuperCollider environment, a flexible dynamic programming language for real-time audio synthesis [11]. Section 2 of this paper outlines the APEX system and how it can be used to generate vocal tract cross-sectional area information suitable for supplying input parameters to a physical model. Section 3 revisits the 1-D Kelly-Lochbaum digital waveguide model of the vocal tract. Section 4 introduces the new APEX SuperCollider implementation with Section 5 verifying the results produced by the vocal tract model and demonstrating the potential offered by the wider system. Section 6 concludes the paper looking at future directions for this work.

2. THE APEX SYSTEM

The original APEX project was designed to map from tract articulation to formant values with audio synthesis of the results obtained. The APEX model takes sampled fixed position data for certain vocal tract articulations and then derives intermediate values from physiologically motivated interpolation rules [1]. The vocal tract geometry is derived from a large number of X-ray images from a single subject, resulting in sampled two-dimensional contour points for the shape of the mandible (or lower jaw), the hard palate, the posterior pharyngeal (back) wall and the larynx, combined with the articulators - the tongue body, tongue tip, and lips. From this an articulatory profile is constructed and a semi-polar coordinate system defined with respect to specific points along the vocal tract - this enables an artificial mid-line along the length of the tract to be defined, equidistant between the front and

back walls. The mid-sagittal distances along this vocal tract mid-line are then calculated between glottis and lips and from these values cross-sectional area functions can be calculated. It is these cross-sectional area values that then enable sound synthesis, and in this work the synthesis engine is based on a 1-D digital waveguide model.

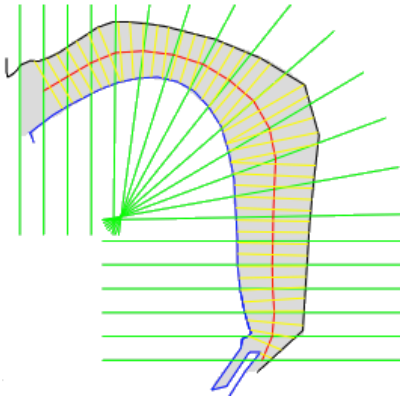


Figure 1: *The APEX vocal tract profile. The back wall of the tract is fixed with the front line depending on the configuration of the vocal tract articulators. The mid-line is first constructed using a semi-polar coordinate system and from this the cross-sectional area functions can be calculated.*

2.1. Voval Tract Cross-Sectional Area Definiton

Figure 1 illustrates this process of defining the cross-sectional area functions. The semi-polar coordinate system is represented by the long lines intersecting the vocal tract profile from an origin external to it. This is used to define the representative mid-line of the tract. New lines are constructed perpendicular to this mid-line so that the distance between the upper (fixed back wall) and lower contours of the vocal tract can be measured. These distances are mapped to equivalent cross-sectional area functions as follows:

$$A(x) = a \cdot d^b \tag{1}$$

Where $A(x)$ is the cross-sectional area function associated with a point along the mid-line x, d is the measured distance between the upper and lower contours of the vocal tract, and a and b are constants varying with vocal tract location and individual speaker characteristics.

2.2. Vocal Tract Articulation

In terms of controlling the vocal tract, some parts can be considered fixed: the posterior pharyngeal wall, the hard palate and upper jaw, as represented by the dark, black line in Figure 1. This data has been extracted from APEX X-ray data from a single individual. The moveable parts of the vocal tract considered in this model are detailed as follows.

The *larynx* shape has been extracted from APEX data and is fixed, although the vertical position can be changed. The larynx height (vertical position) varies only a little during speech, with females typically having a higher larynx position, and children even higher. The shorter vocal tract that results causes the formant frequencies to be scaled up relative to the male voice.

The *tongue body* is the most deformable part of the vocal tract and control parameters have been determined by principal component analysis (PCA). In [12] 394 tongue contours were obtained from X-ray images and each sampled at 25 approximately equidistant points. PCA results in a linear, weighted combination of a smaller number of basis functions as follows:

$$S_{target}(x) = S_{neutral}(x) + \sum_{i=1}^{N} c_i(v) PC_i(x) \tag{2}$$

Where x is the index of the point on the sampled tongue contour, $S_{target}(x)$ is the calculated tongue shape, $S_{neutral}(x)$ is a 'neutral' tongue shape, being the average of the measured shapes, and $PC_i(x)$ is the i^{th} basis function. The coefficient c_i is the weighting for the i^{th} basis function, being a 2-dimensional vector value which depends on the vowel, used as a parameter for calculating the tongue shape. From [12], setting $N = 1$ in (2) is found to account for 85.7% of the variance, while setting $N = 2$ achieves. 96.3%. In this work $N = 3$, giving three 2-D control parameters for tongue shape.

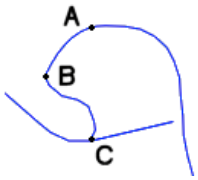


Figure 2: *The tongue tip geometry with the three required points labelled. A is defined as the end of the tongue body, B is the tongue tip and C is the point where the underside of the tongue is fixed to the lower jaw.*

The *tongue tip* is defined according to three points as shown in Figure 2. The tongue body ends at point A on the upper surface of the tongue, point B is the tongue tip and point C is where the underside of the tongue blade is connected to the mouth floor in the lower jaw coordinate system. Hermite interpolation is used between points A and B, with measured data used to form the curve between B and C based on the matched values at these points.

Jaw movement is the translation and then the rotation of the lower jaw coordinate system. The result is that some part of the frontal vocal tract is moved, including both the tongue body and blade, and the mouth floor and teeth (if they were included in this model). The angle for the rotation is determined by:

$$\theta = \frac{J}{2} + 7 \tag{3}$$

Where θ is the angle in degrees, and J is the jaw opening or distance between the upper and lower teeth (in mm).

3. THE 1-D VOCAL TRACT MODEL

The acoustic properties of the vocal tract can be modelled by considering it to be, at its simplest level, a straight tube from glottis to the lips where acoustic wave propagation is predominantly determined by the 1-D lossless wave equation:

$$\frac{\partial^2 p(x,t)}{\partial t^2} = c^2 \frac{\partial^2 p(x,t)}{\partial x^2} \tag{4}$$

where p is acoustic pressure, t is time, c is the speed of sound and x describes the (1-D) coordinate system. The Kelly-Lochbaum 1-D model, equivalent to a 1-D digital waveguide, provides a numerical solution to (4) and is well documented in the literature (e.g. [3] [6] [7] [9]).

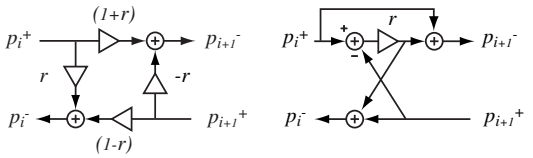


Figure 3: *Kelly-Lochbaum scattering of pressure signals in (a) the standard two-port scattering junction and (b) the one-multiply equivalent*

Summarising, the air column within the tube under consideration is discretized and represented as a series of connected bidirectional 1-D digital waveguide elements. The varied shape of the vocal tract along its length, as quantified by the cross-sectional area function, gives rise to the tract being defined as a series of concatenated acoustic tube elements, varying from glottis to lips. The change in cross-sectional area between tube elements is in turn modelled as a change in acoustic impedance for each waveguide element relative to the corresponding cylindrical tube section. These conditions give rise to the well known Kelly-Lochbaum scattering junction as shown in Figure 3(a) and expressed in the following equations:

$$r = \frac{A_i - A_{i+1}}{A_i + A_{i+1}} \tag{5}$$

$$\begin{aligned} p_i^- &= rp_i^+ + (1-r)p_{i+1}^+ \\ p_{i+1}^- &= (1+r)p_i^+ - rp_{i+1}^+ \end{aligned} \tag{6}$$

Where r is the reflection coefficient between tube sections i and $i+1$ of cross-sectional area A_i and A_{i+1} respectively, p_i^+ is the incoming pressure value to the scattering junction from tube section i, p_{i+1}^+ is the outgoing pressure travelling in the same direction to tube section $i+1$. p_{i+1}^- is the incoming pressure value to the scattering junction from tube section $i+1$, with p_i^- the outgoing pressure value travelling in the same direction to tube section i. As shown in Figure 3(b) this can be expressed more efficiently for computational implementation as:

$$\begin{aligned} p_i^- &= p_{i+1}^+ + w \\ p_{i+1}^- &= p_i^+ + w \end{aligned} \tag{7}$$

Where:

$$w = r(p_i^+ - p_{i+1}^+) \tag{8}$$

The model is completed with terminations accounting for lip and glottal reflection at each end. The lip end may also be terminated with an appropriate filter to approximate lip radiation effects. With the introduction of appropriate glottal excitation at the glottal end, signal output at the lip end results in speech sounds being produced.

4. APEX SUPERCOLLIDER IMPLEMENTATION

APEX vocal tract articulation control and a synthesis engine based on the 1-D Kelly-Lochbaum digital waveguide have been implemented in the real-time SuperCollider audio processing environment (v3.6). The server-client architecture of SuperCollider results in the APEX graphical user interface (GUI) running on the client with the data-processing and sound synthesis being handled separately on the server. The APEX GUI is shown in Figure 4 and is divided into three panels: the left hand side vocal tract articulation controls, the centre panel vocal tract graphical representation, and the right hand panel system parameter and synthesis information.

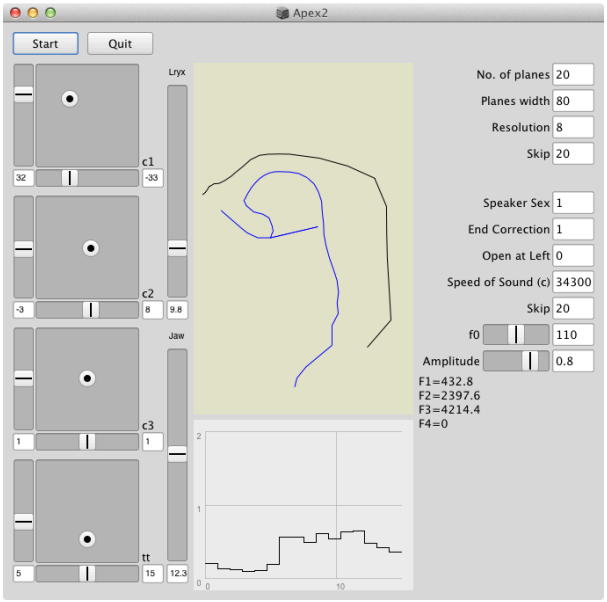


Figure 4: *The APEX SuperCollider implementation graphical user interface. The left hand side are the vocal tract articulation controls, the centre panels give a real-time graphical representation of the articulation and resulting cross-sectional area values. The right hand panel allows the user to set parameters relevant to the spatial sampling of the tract shape and the synthesis engine.*

4.1. APEX User Control

The left hand side of Figure 4 shows six vocal tract articulation controls. The initial state of the tract in terms of the tongue body neutral shape and articulator states are loaded from data files at startup. The top three 2-D panel controllers are mapped to the three Principal Component variables derived from (2). The bottom 2-D panel is mapped to the Tongue Tip control (in mm), with the two vertical sliders used to control larynx height and jaw opening, both in mm. Upon user interaction, the parameters from these six articulation controls are passed via the client to a server based unit generator, or UGen, that is used to calculate the cross-sectional area function data based on the shape of the current articulation. The updated articulation data is then returned to the client to enable the centre panel graphical representation of the vocal tract to be updated, providing valuable feedback in terms of user control. It

is also possible to view the relative changes in cross-sectional area along the vocal tract itself in the lower of the two centre panels.

The top of the right hand panel displays parameters relevant to the system configuration. The 'No. of Planes' parameter is used to set the number of planes in the semi-polar coordinate system that determines the vocal tract mid-line, with 'Planes Width' setting their width (or length) to ensure intersection with the back wall of the tract. The key parameter in terms of the resulting synthesis is 'Resolution', as this sets the spatial sampling interval (in mm) along the mid-line and hence will directly impact the number of cross-sectional area functions calculated.

One of the overheads with this SuperCollider implementation is the communication between server and client and ensuring that synchronisation is maintained between the two. The processing functions of the UGens are called at fixed time intervals (for the control rate UGens used in APEX the default is the audio sample rate divided by 64). The 'Skip' parameter tells the area function calculator UGen to skip a certain number of these calls between calculations. The minimum number for this parameter therefore depends on the speed of the computer being used and any additional load due to the synthesis engine.

The lower parameters relate to a prototype synthesis engine that calculates formant values based on the cross-sectional area values, passing them to a simple oscillator based formant synthesizer. Formants are calculated based on volume velocity transfer through the vocal tract [13], and although this synthesis method is superseded by the 1-D digital waveguide model discussed here, the calculated formant values displayed act as a useful additional check on the synthesized output. Finally the bottom two horizontal sliders allow run-time control of fundamental frequency and amplitude of the output.

4.2. APEX Vocal Tract Synthesis

The 1-D digital waveguide vocal tract model is also implemented as a server-side UGen, based on a specific implementation of the more general SuperCollider *NTube* synthesizer from Nick Collins' SLUGens plugins distribution [14]. The *KLVocalTract* UGen takes three parameters, the input excitation to the system, an array of $N-1$ reflection coefficients based on N cross-sectional area values (5) and the total delay line length in seconds. This is dependent on the sample rate of the system and the spatial sampling 'Resolution' of the vocal tract. Each cross-sectional area tube length, of which there are N, must have a delay of greater than two samples. Losses are set at the lip and glottis end based on reflection values from [7], such that $r_{glottis} = 0.969$ and $r_{lip} = -0.9$. The one-multiply equivalent scattering junction, as given in (7) and (8), is used for efficiency in implementation.

The excitation function, as shown in Figure 3, is implemented as a SuperCollider client-side, time-reversed glottal flow model, in effect the typical Liljencrants-Fant (LF) glottal flow model without a return phase. For output as sound pressure a +6dB/oct radiation characteristic is added. A single cycle of the excitation, with a fundamental frequency of 110Hz is shown in Figure 3(a) with the resulting spectrum in Figure 3(b).

5. RESULTS

The goal at this stage in the development of the SuperCollider APEX tool is to demonstrate the correct implementation of the 1-D digital waveguide model as part of the overal APEX system, and to highlight its capability for real-time dynamic articulation and voice synthesis.

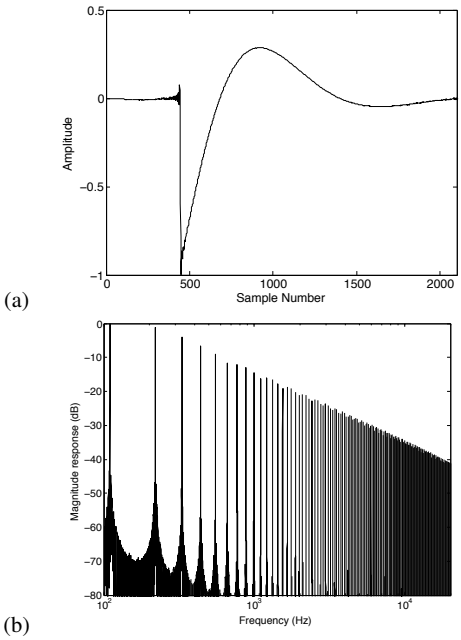


Figure 5: *The 1-D digital waveguide vocal tract excitation model as implemented in SuperCollider, in this case with a fundamental frequency of 110Hz. (a) The result is a time-reversed approximation to the LF glottal flow model; (b) the resulting spectrum.*

5.1. 1-D Vocal Tract Synthesis

Initially the *KLVocalTract* UGen is tested externally to the APEX front end in order to demonstrate its capability for synthesizing voice-like sound. With end termination reflection coefficient values defined as in Section 4.2 and a uniform cross-sectional area function set along the length of the tract, such that there is no internal scattering, should result in quarter-wave resonator like behaviour. In order to test this, *KLVocalTract* is instantiated with 44 equal area function sections, the tract length $L = 0.176$m, speed of sound $c = 343\text{ms}^{-1}$ and the sample rate is set to 192kHz. A white noise source is used as excitation at the glottis end and a 5s output signal is recorded at the lip end. The modal peaks for quarter-wave resonator behaviour can be predicted using:

$$f_{QWR} = \frac{(2m+1)c}{4L} : m \in \mathbb{N}^0 \tag{9}$$

These analytical values are overlaid on the Welch power spectral density of the output in Figure 6 and can be seen to give good agreement to the measurement obtained.

The next stage is to examine the output from *KLVocalTract* for a static cross-sectional area profile that relates more directly to speech-like sound. In this case *KLVocalTract* is instantiated with 44 area function sections under the same conditions as before, but this time based on data from [15] for the /a/ (*Bard*) vowel. As well as white noise excitation, glottal source excitation is also used and 5s output signals are recorded from the lip end. The results

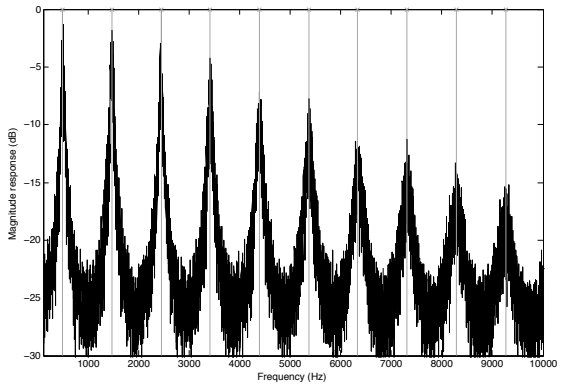


Figure 6: *Welch power spectral density estimate for 5s output from KLVocalTract functioning as a quarter wave resonator with white noise excitation. The grey vertical lines represent the analytical modal peaks for a quarter wave resonator of the same dimensions.*

obtained are compared with average measured formant values for the /a/ vowel based on data from [16] and the results are presented in Figure 7.

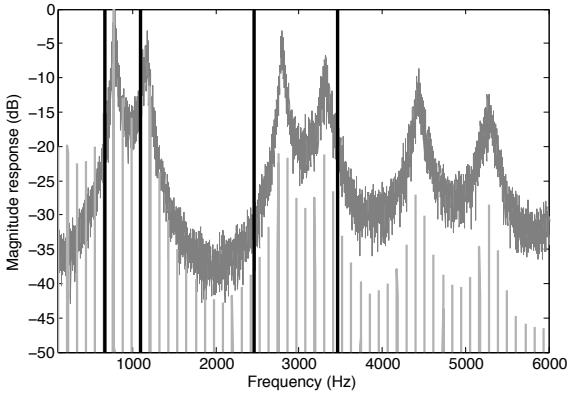


Figure 7: *KLVocalTract synthesis of 5s sound output for the /a/ vowel: Welch power spectral density estimate based on noise (solid dark grey line) and glottal source excitation (light grey lines) compared with average measured values of first four formants (black vertical lines).*

The results give good general agreement, and are comparable with those presented in e.g. [7], [9] which include vowel synthesis/formant values based on a 1-D digital waveguide model (amongst others).

5.2. APEX Real-time Articulatory Synthesis

In this example, the APEX SuperCollider GUI is used as a means to control the *KLVocalTract* UGen, with the dimension and number of the cross-sectional area vocal tract tube sections passed as the input to the synthesizer, together with fundamental frequency values to enable the glottal source excitation to vary pitch accordingly. To test the capabilities of the complete system, pre-calculated APEX data is passed into the GUI, giving the (x, y) values for the three

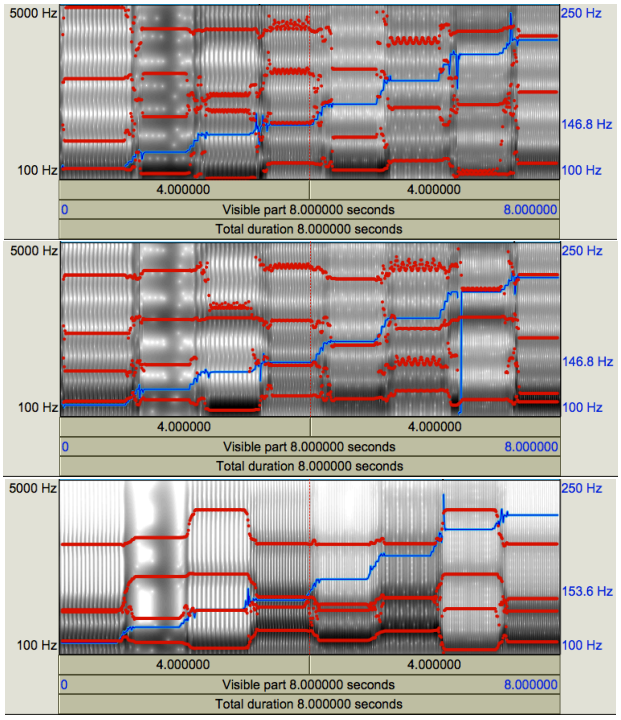


Figure 8: *Praat spectrograms with candidate fundamental frequency and formant values identified and tracked over the 8s of output for the octave scale of 'DoReMe'. The fundamental frequency line corresponds to the right hand scale and is overlaid on the broader formant value calculation spectrogram plot. The top two examples use KLVocalTract with a resolution of 4mm (top) and 5mm (middle). The bottom plot is based on the method presented in [13] and formant synthesis.*

PCA parameters plus the tongue tip, together with fundamental frequency, jaw opening and larynx height positions to enable the scale octave for 'DoReMe' to be articulated. Each note has a start and end set of values over a period of 0.9s, with a transition of 0.1s to the next note. Values are interpolated between states by the APEX system, and the 8s sound output is recorded at the lip end. Two different APEX states are used to obtain the results. In both cases 35 planes are used to calculate the vocal-tract mid-line (the maximum currently allowed by the system), but in the first example a spatial sampling resolution of 4mm is used, and in the second this is set to 5mm, for a sampling rate of 96kHz. For the 4mm case this results in 33-35 vocal tract tube sections, and for the 5mm case 25-27 sections. The number of sections varies slightly as the articulations can actually change the length of the vocal tract model, and in this current implementation, every articulation results in a complete recalculation of the mid-line and cross-sectional area functions, and hence the potential for the total delay-line length to change accordingly in *KLVocalTract*.

In addition to using *KLVocalTract* and to enable some comparison, the same phrase is synthesized based on the method presented in [13] using a band limited saw-tooth oscillator (SuperCollider *Saw* UGen), and up to four resonant low-pass filters centred on the calculated formant frequencies.

The initial fundamental frequency value in each case is 110Hz. These three sets of results are analysed using Praat, and spectrograms produced with automatic tracking of fundamental frequency and formant frequency values enabled (based on default Praat settings). The results are shown in Figure 8 and corresponding audio examples, resampled to 44.1kHz at 16bit resolution are available for audition at `http://www-users.york.ac.uk/~dtm3/vocaltract.html`.

Note that the first and last notes of the scale have the same articulation values, and so should give the same results apart from an octave shift in fundamental frequency. This is clearly the case in the format synthesis example, although it seems that an additional candidate value has also been found. Interestingly, based on the same default analysis values, the 5mm resolution example gives a closer match. Similarly, the lower resolution case in most cases results in a more consistent tracking of identified formats although both demonstrate similarities and differences with the formant synthesis example. Across transitions it is evident that all examples demonstrate discontinuities due to the interpolations applied between articulation states, especially for fundamental frequency. It is notable that the formant synthesis example demonstrates clearer and smoother transitions although this is expected due to the relative simplicity of the synthesis algorithm used and the fact that formants are synthesized directly (from specific filter settings), rather than indirectly as a consequence of the physical modelling algorithm used.

However, the higher resolution *KLVocalTract* does, for some notes, exhibit better, more consistent formant values (Note 3, Note 5, Note 6, Note 7, when compared with the lower plot in Figure 8)) and ultimately results in a better quality sound output, even if formant tracking transition states are a lot less clear than the lower resolution example. This poor formant tracking at high resolution is due to the nature of the cross-sectional area functions calculated by the APEX controller and how they are applied in *KLVocalTract*. Lower resolution implies fewer tube sections and so fewer (or smaller) changes (and resulting interpolations) to be applied to *KLVocalTract* between articulation states. This is particularly the case when the tract length itself changes due to an applied articulation (e.g. larynx height) as the number of tube sections will also change, resulting in discontinuities in the output of *KLVocalTract* as the delay line length will also have to be updated. This is further compounded if, at the lip end, the jaw opening parameter is not mapped directly to the last tube section that also corresponds to where the output is tapped off. This can result in restricted sound output, and mistuning of expected formant positions, as is evidenced in some of the cases in Figure 8. Finally, it should be noted that this example does apply significant, multi-parameter changes in articulation states with the associated required interpolation of these values. Direct user control of the GUI, where fewer parameters are changed in real-time with smaller degrees of variation, results in much smoother and continuous examples of articulatory synthesis.

6. CONCLUSIONS

This paper has presented a SuperCollider implementation of the APEX articulatory vocal tract algorithm, coupled with the *KLVocalTract* 1-D digital waveguide synthesis engine. The latter has demonstrated its potential for accurate, real-time formant synthesis, based as it is on established methods. The APEX GUI reduces the number of variables required for articulatory speech synthesis to a much more manageable number, and SuperCollider provides an excellent real-time framework for linking these two components while providing additional access to standard sound synthesis unit generator processes and user control options. This first investigation into the potential for such articulatory vocal tract synthesis in SuperCollider has revealed that for realistic speech-like sound output the calculation of cross-sectional tract profiles must be done more efficiently, with direct mapping to the tube sections implemented in the synthesis engine, such that the underlying delay-line length does not have to change. This would enable smoother control of *KLVocalTract* while also enabling its eventual replacement with a modelling engine of higher dimension.

In addition, the articulatory controls require some further refinement. The PC variables need to be constrained such that physically impossible tract profiles are similarly not allowed - for instance the tongue tip should not be able to move through the back wall of the tract. The lips also need to be more completely represented, the tongue body should maintain a constant volume, and in this current implementation, branches on the tract, such as the subapical space and velar port/nasal tract are not represented in either the controller or the underlying model.

A version of the source code for this work is also available at `http://www-users.york.ac.uk/~dtm3/vocaltract.html` such that it might be adopted by the wider SuperCollider community.

7. ACKNOWLEDGEMENTS

We are indebted to Björn Lindblom and Christine Ericsdotter, both at Stockholm University, and to Johan Sundberg at KTH, for generously sharing their expertise and data, painstakingly acquired for the original APEX Windows 3.1 implementation, which was written by Johan Stark. The groundwork for the SuperCollider implementation of APEX was done in 2011 in the M.Sc. thesis of author Jani. The extended version presented here was developed in 2014-2015 by authors Murphy and Ternström. We are grateful also to Gerhard Eckel and Ludvig Elblaus for advice and assistance with SuperCollider. The work was supported in part by the Swedish Research Council (VR), contract D-2013-0378.

8. REFERENCES

[1] J. Stark, C. Ericsdotter, P. Branderud, J. Sundberg, H-J. Lundberg, and J. Lander, "The APEX model as a tool in the specification of speaker specific articulatory behaviour," in *Proc. XIVth ICPhS*, San Francisco, USA, 1999.

[2] H. Takemoto, P. Mokhtari, and T. Kitamura, "Acoustic analysis of the vocal tract during vowel production by finite-difference time-domain method," *J. Acoust. Soc. Am*, vol. 128, no. 6, pp. 3724–3738, Dec. 2010.

[3] M. Speed, D. T. Murphy, and D. M. Howard, "Modeling the Vocal Tract Transfer Function Using a 3D Digital Waveguide Mesh," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 22, no. 2, pp. 453–464, Feb. 2014.

[4] B. Delvaux and D. M. Howard, "A New Method to Explore the Spectral Impact of the Piriform Fossae on the Singing Voice: Benchmarking Using MRI-Based 3D-Printed Vocal Tracts," *PLoS ONE*, vol. 9, no. 7, pp. e102680–15, July 2014.

[5] T. Vampola, J. Horáček, and J. G. Švec, "Modeling the Influence of Piriform Sinuses and Valleculae on the Vocal Tract Resonances and Antiresonances," *Acta Acustica united with Acustica*, vol. 101, no. 3, pp. 594–602, May 2015.

[6] J. L. Kelly and C. C. Lochbaum, "Speech synthesis," in *Proc. 4th Int. Congr. Acoustics*, Copenhagen, Denmark, 1962, pp. 1–4.

[7] J. Mullen, D. M. Howard, and D. T. Murphy, "Waveguide physical modeling of vocal tract acoustics: flexible formant bandwidth control from increased model dimensionality," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 3, pp. 964–971, May 2006.

[8] M. Arnela and O. Guasch, "Two-dimensional vocal tracts with three-dimensional behavior in the numerical generation of vowels.," *J. Acoust. Soc. Am*, vol. 135, no. 1, pp. 369–379, Dec. 2013.

[9] J. Mullen, D. M. Howard, and D. T. Murphy, "Real-time dynamic articulations in the 2-D waveguide mesh vocal tract model," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, no. 2, pp. 577–585, 2007.

[10] D. T. Murphy, S. Shelley, and S. Ternström, "The dynamically varying digital waveguide mesh," in *Proc. 19th Int. Congr. Acoustics*, Madrid, Spain, Sept. 2007.

[11] J. McCartney et al., "SuperCollider," Available at http://supercollider.github.io/, accessed May 15, 2015.

[12] B. Lindblom, "A numerical model of coarticulation based on a principal components analysis of tongue shapes," in *Proc. 15th Int. Congr. Phonetic Sciences*, Barlcelona, Spain, 2003.

[13] J. Liljencrants and G. Fant, "Computer program for vt-resonance frequency calculations," in *Dept. for Speech, Music and Hearing, Quarterly Progress and Status Report STL-QPSR 16(4)*, KTH, Stockholm, 1975, pp. 15–02.

[14] N. Collins, "SC3 plugin contributions," Available at http://composerprogrammer.com/code.html, accessed May 15, 2015.

[15] B. H. Story, I. R. Titze, and E. A. Hoffman, "Vocal tract area functions from magnetic resonance imaging," *JASA*, vol. 100, no. 1, pp. 537–554, Feb. 1996.

[16] D. G. Childers, *Speech Processing and Synthesis Toolboxes*, New York: Wiley, 2000.

Keynote 3

Franz Zotter: Ambisonic Audio Effects in Direction and Directivity

Abstract The properties of the spherical harmonics to represent patterns on the sphere, as well as their deep embedding in the acoustical wave equation, enable many nice audio effects in space.

First of all, the inherent smoothness of the finite-order spherical harmonic representations is the basis of Ambisonic amplitude panning on surrounding loudspeakers. The spherical harmonic representation is mighty enough to represent several directional effects, such as mirroring, rotation, directional loudness manipulation, directional warping in terms of a simple matrix multiplication.

What is more, the associated acoustic equations permit design and signal processing of not only microphone arrays for Ambisonic recording, but also spherical loudspeaker arrays for directivity synthesis.

The plenary lecture gives examples and explanations of freely available Plugins for Ambisonics (VST AmbiX plugin suite), and reveals a peek on adjustable directivity as a musical instrument.

A MODEL FOR ADAPTIVE REDUCED-DIMENSIONALITY EQUALISATION

Spyridon Stasis, Ryan Stables, Jason Hockman

DMT Lab,
Birmingham City University,
Birmingham, UK.
spyridon.stasis@bcu.ac.uk
ryan.stables@bcu.ac.uk
jason.hockman@bcu.ac.uk

ABSTRACT

We present a method for mapping between the input space of a parametric equaliser and a lower-dimensional representation, whilst preserving the effect's dependency on the incoming audio signal. The model consists of a parameter weighting stage in which the parameters are scaled to spectral features of the audio signal, followed by a mapping process, in which the equaliser's 13 inputs are converted to (x, y) coordinates. The model is trained with parameter space data representing two timbral adjectives (*warm* and *bright*), measured across a range of musical instrument samples, allowing users to impose a semantically-meaningful timbral modification using the lower-dimensional interface. We test 10 mapping techniques, comprising of dimensionality reduction and reconstruction methods, and show that a stacked autoencoder algorithm exhibits the lowest parameter reconstruction variance, thus providing an accurate map between the input and output space. We demonstrate that the model provides an intuitive method for controlling the audio effect's parameter space, whilst accurately reconstructing the trajectories of each parameter and adapting to the incoming audio spectrum.

1. BACKGROUND

Equalisation is an integral part of the music production workflow, with applications in live sound engineering, recording, music production and mastering, in which multiple frequency dependent gains are applied to the audio signal. Generally the process of equalisation can be categorised under one of the following headings, *corrective equalisation*: in which problematic frequencies are often attenuated in order to prevent issues such as acoustic feedback, and *creative equalisation*: in which the audio spectrum is modified to achieve a desirable timbral transformation. The latter often involves a process of translation between a perceived timbral adjective such as *bright*, *flat* or *sibilant* and an audio effect's input space, by which a music producer must reappropriate a perceptual representation of a timbral transformation as a configuration of multiple parameters in an audio processing module. As music production is an inherently technical process this mapping procedure is not necessarily trivial, and is made more complex by the source-dependent nature of the task.

We propose a system that projects the controls of a parametric equaliser comprising 5 biquad filters arranged in series onto an editable two-dimensional space, allowing the user to manipulate the timbre of an audio signal using an intuitive interface. Whilst the axes of the two-dimensional space are somewhat arbitrary, underlying timbral characteristics are projected onto the space via a training stage using musical semantics data. In addition to this, we propose a signal processing method of adapting the parameter modulation process to the incoming audio data based on feature extraction applied to the long-term average spectrum (LTAS), capable of running in near-real-time. The model is implemented using the SAFE architecture (detailed in [1]), and provided as an extension of the current Semantic Audio Parametric Equaliser,[1] shown in Figure 1.

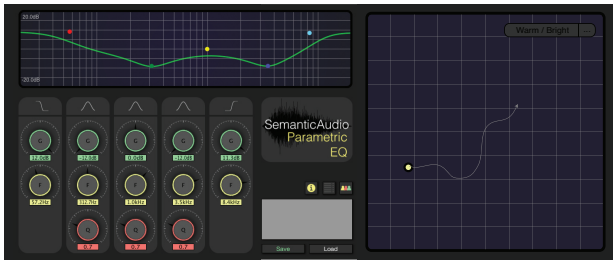


Figure 1: The extended Semantic Audio Equalisation plug-in with the two-dimensional interface. To modify the brightness/warmth of an audio signal, a point is positioned in two-dimensional space.

1.1. Semantic Music Production

Engineers and producers generally use a wide variety of timbral adjectives to describe sound, each with varying levels of agreement. By modelling these adjectives, we are able to provide perceptually meaningful abstractions, which lead to a deeper understanding of musical timbre and system that facilitate the process of audio manipulation. The extent to which timbral adjectives can be accurately modelled is defined by the level of exhibited agreement, a concept investigated in [2], in which terms such as bright, resonant and harsh all exhibit strong agreement scores and terms such as open, hard and heavy all show low subjective agreement scores. It is common for timbral descriptors to be represented in low-dimensional space, brightness for example is shown to exhibit a strong correlation with spectral centroid [3, 4] and has further dependency on the fundamental frequency of the signal [5]. Simi-

[1] Available for download at http://www.semanticaudio.co.uk

larly, studies such as [6] and [7] demonstrate the ability to reduce complex data to lower-dimensional spaces using dimensionality reduction.

Recent studies have also focused on the modification of the audio signal using specific timbral adjectives, where techniques such as spectral morphing [8] and additive synthesis [9] have been applied. For the purposes of equalisation, timbral modification has also been implemented via semantically-meaningful controls and intuitive parameter spaces. SocialEQ [10] for example, collects timbral adjective data via a web interface and approximates the configuration of a graphic equaliser curve using multiple linear regression. Similarly, subjEQt [11], provides a two-dimensional interface, created using a Self Organising Map, in which users can navigate between presets such as *boomy*, *warm* and *edgy* using natural neighbour interpolation. This is a similar model to 2DEQ [12], in which timbral descriptors are projected onto a two-dimensional space using Principal Component Analysis (PCA). The Semantic Audio Feature Extraction (SAFE) project provides a similar non-parametric interface for semantically controlling a suite of audio plug-ins, in which semantics data is collected within the DAW. Adaptive presets can then be selectively derived, based on audio features, parameter data and music production metadata.

2. METHODOLOGY

In order to model the desired relationship between the two parameter spaces, a number of problems must be addressed. Firstly, the data reduction process should account for maximal variance in high-dimensional space, without bias towards a smaller subset of the EQ's parameters. Similarly, we should be able to map to the high-dimensional space with minimal reconstruction error, given a new set of (x, y) coordinates. This process of mapping between spaces is nontrivial, due to loss of information in the reconstruction process. Furthermore, the low-dimensional parameter space should be configured in a way that preserves an underlying timbral characteristic in the data, thus allowing a user to transform the incoming audio signal in a musically meaningful way. Finally, the process of parameter space modification should not be agnostic of the incoming audio signal, meaning any mapping between the two-dimensional plane and the EQ's parameter space should be expressed as a function of the (x, y) coordinates and some representation of the signal's spectral energy. In addition to this, the system should be capable of running in near-real time, enabling its use in a Digital Audio Workstation (DAW) environment.

For addressing these problems, we develop a model that consists of two phases, where the first comprises a training phase, in which a map is derived from a corpus of parameter data, and the second comprises a testing phase in which a user can present (x, y) coordinates and an audio spectrum, resulting in a 13 dimensional vector of parameter state variables. To optimise the mapping process, we experiment with a combination of 3 dimensionality reduction techniques and 3 reconstruction methods, followed by a stacked-autoencoder model that encapsulates both the dimensionality reduction and reconstruction processes. With the purpose of scaling the parameters to the incoming audio signal, we derive a series of weights based on a selection of features, extracted from the signal's LTAS coefficients. To evaluate the model's performance, we train it with binary musical semantics data and measure the parameter-wise reconstruction error, along with inter-class variance in low-dimensional space.

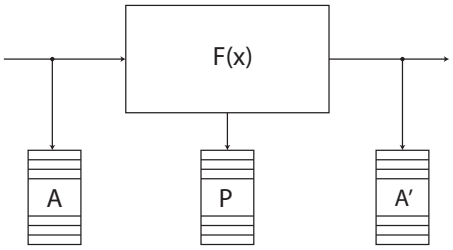


Figure 2: An overview of the SAFE data collection architecture, where A represents the audio features captured before the effect is applied, A' represents the features captured after the effect is applied, and P represents the parameter vector.

2.1. Dataset

For the training of the model, we compile a dataset of 800 semantically annotated EQ parameter space settings, comprising 40 participants equalising 10 musical instrument samples using 2 descriptive terms: *warm* and *bright*. To do this, participants were presented with the musical instrument samples in a DAW and asked to use a parametric equaliser to achieve the two timbral settings. After each setting was recorded, the data were recorded and the equaliser was reset to unity gain. During the test samples were presented to the participants in a random order, across separate DAW channels. Furthermore, the musical instrument samples were all performed unaccompanied, were RMS normalised and ranged from 20 to 30 seconds in length. All of the participants had normal hearing, aged 18-40 and all had at least 3 years' music production experience.

The descriptive terms (*warm* and *bright*) were selected for a number of reasons, firstly the agreement levels exhibited by participants tend to be the high (as suggested by [2]), meaning there should be less intra-class variance when subjectively assigning parameter settings. When measured using an agreement metric, defined by [10] as the log number of terms over the trace of the covariance matrix, *warm* and *bright* were the two highest ranked terms in a dataset of 210 unique adjectives. Secondly, the two terms are deemed to be sufficiently different enough to form an audible timbral variation in low dimensional space. Whilst the two terms do not necessarily exhibit orthogonality (for example brightness can be modified with constant *warmth* [8]), they have relatively dissimilar timbral profiles, with brightness widely accepted to be highly correlated with the signal's spectral centroid, and *warmth* often attributed to the ratio of the first 3 harmonics to the remaining harmonic partials in the magnitude LTAS [13].

The parameter settings were collected using a modified build of the SAFE data collection architecture, in which descriptive terms, audio feature data, parameter data and metadata can be collected remotely, within the DAW environment and uploaded to a server. As illustrated in Figure 2, the SAFE architecture allows for the capture of audio feature data before and after processing has been applied. Similarly, the interface parameters P (see Table 1) are captured and stored in a linked database. For the purpose of this experiment, the architecture was modified by adding the functionality to capture LTAS coefficients, with a window size of 1024 samples and a hop size of 256.

Whilst the SAFE project comprises a number of DAW plug-ins, we focus solely on the parametric equaliser, which utilises 5

biquad filters arranged in series, consisting of a low-shelving filter (LS), 3 peaking filters (Pf_n) and a high-shelving filter (HS), where the LS and HS filters each have two parameters and the (Pf_n) filters each have 3, as described in Table 1.

n	Assignment	Range	n	Assignment	Range
0	LS gain	-12 - 12 dB	7	Pf_1 Q	0.1 - 10 Hz
1	LS Freq	22 - 1,000 Hz	8	Pf_2 Gain	-12 - 12 dB
2	Pf_0 Gain	-12 - 12 dB	9	Pf_2 Freq	220 - 10,000 Hz
3	Pf_0 Freq	82 - 3,900 Hz	10	Pf_2 Q	0.1 - 10 Hz
4	Pf_0 Q	0.1 - 10 Hz	11	HS Gain	-12 - 12 dB
5	Pf_1 Gain	-12 - 12 dB	12	HS Freq	580 - 20,000 Hz
6	Pf_1 Freq	180 - 4,700 Hz			

Table 1: A list of the parameter space variables and their ranges of possible values, taken from the SAFE parametric EQ interface.

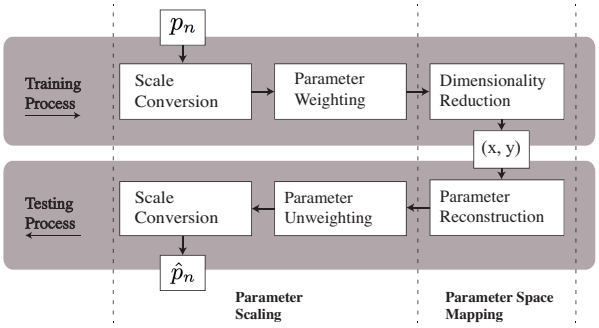


Figure 3: An overview of the proposed model, where the grey horizontal paths represent training and testing phases.

3. MODEL

The proposed system maps between the EQ's parameter space, consisting of 13 filter parameters and a two-dimensional plane, whilst preserving the context-dependent nature of the audio effect. After an initial training phase, the user can then submit (x, y) coordinates to the system using a track-pad interface, resulting in a timbral modification via the corresponding filter parameters. To demonstrate this, we train the model with 2 class (*bright*, *warm*) musical semantics data taken from the SAFE EQ database, thus resulting in an underlying transition between opposing timbral descriptors, in two-dimensional space. By training the model in this manner, we intend to maximise the separability between classes when projected onto the reduced-dimensionality interface.

The model (depicted in Figure 3) has 2 key operations, The first involves weighting the parameters by computing the vector $\alpha_n(A)$ from the input signal's long-term spectral energy (A). We can then modify the parameter vector (P), to obtain a weighted vector (P'). The second component scales the dimensionality of (P'), resulting in a compact, audio-dependent representation. During the model's testing phase, we apply an unweighting procedure, based on the (x, y) coordinates and the signal's modified spectrum. This is done by multiplying the estimated parameters with the inverse weight vector, resulting in an approximation of the original parameters. In addition to the weighting and dimensionality reduction stages, a scale-normalisation procedure is applied aiming to convert the ranges of each parameter (given in Table 1), to ($0 < p_n < 1$). This converts the data into a suitable format for dimensionality reduction.

3.1. Parameter Scaling

As the configuration of the filter parameters assigned to each descriptor by the user during equalisation is likely to vary based on the audio signal being processed, the first requirement of the model is to apply weights to the parameters, based on knowledge of the audio data at the time of processing. To do this, we selectively extract features from the signal's LTAS, before and after the filter is applied. This is possible due to the configuration of the data collection architecture, highlighted in Figure 2. The weights (α_m) can then be expressed as a function of the LTAS, where the function's definition varies based on the parameter's representation (i.e. gain, centre frequency or bandwidth of the corresponding filter). We use the LTAS to prevent the parameters from adapting each time a new frame is read. In practice, we are able to do this by presenting users with means to store the audio data, rather than continually extracting it from the audio stream. Each weighting is defined as the ratio between a spectral feature, taken from the filtered audio signal (A'_k) and the signal filtered by an enclosing rectangular window (R). Here, the rectangular window is bounded by the minimum and maximum frequency values attainable by the observed filter $f_k(A)$

We can define the equaliser as an array of biquad functions arranged in series, as depicted in Eq 1

$$\begin{gathered} f_k = f_{k-1}(A, \vec{P}_{k-1}) \\ k = 1, \ldots, K-1 \end{gathered} \tag{1}$$

Here, $K = 5$ represents the number of filters used by the equaliser and f_k represents the k^{th} biquad function, which we can define by its transfer function, given in Eq 2.

$$H_n(z) = c \cdot \frac{1 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{2}$$

The LTAS is then modified by the filter as in Eq. 3 and the weighted parameter vector can be derived using the function expressed in Eq. 4.

$$A'_k = H_k(e^{j\omega}) A_k \tag{3}$$

$$p'_n = \alpha_m(k) \cdot p_n \tag{4}$$

Where p_n is the n^{th} parameter in the vector P. The weighting function is then defined by the parameter type (m), where $m = 0$ represents gain, $m = 1$ represents centre-frequency and $m = 2$ represents bandwidth. For gain parameters, the weights are expressed as a ratio of the spectral energy in the filtered spectrum (A') to the spectral energy in the enclosing rectangular window (R_n), derived in Eq. 5 and illustrated in Figure 4.

$$\alpha_0(k) = \frac{\sum_i (A'_k)_i}{\sum_i (R_k)_i} \tag{5}$$

For frequency parameters ($m = 1$), the weights are expressed as a ratio of the respective spectral centroids of A' and R_n, as demonstrated in Eq. 6, where b_i are the corresponding frequency bins.

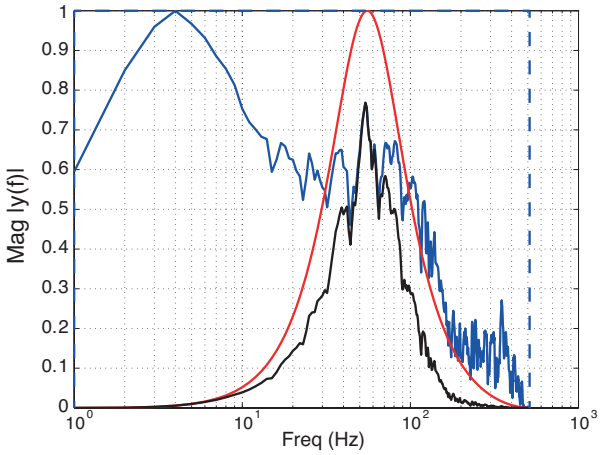


Figure 4: An example spectrum taken from an input example, weighted by the biquad coefficients, where the red line represents a peaking filter, the black line represents the biquad-filtered spectrum and the blue line represents the spectral energy in the rectangular window (R_m).

$$\alpha_1(k) = \left(\frac{\sum_i (A'_k)_i b_i}{\sum_i (A'_k)_i}\right) / \left(\frac{\sum_i (R_k)_i b_i}{\sum_i (R_k)_i}\right) \tag{6}$$

Finally, the weights for bandwidth parameters ($m = 2$) are defined as the ratio of spectral spread exhibited by both A' and R_n. This is demonstrated in Eq. 7, where $(x)_{sc}$ represents the spectral centroid of x.

$$\alpha_2(k) = \left(\frac{\sum_i \left(b_i - (A'_k)_{sc}\right)^2 (A'_k)_i}{\sum_i (A'_k)_i}\right) / \left(\frac{\sum_i \left(b_i - (R_k)_{sc}\right)^2 (R_k)_i}{\sum_i (R_k)_i}\right) \tag{7}$$

During the testing phase, retrieval of the unweighted parameters, given a weighted vector can be achieved by simply multiplying the weighted parameters with the inverse weights vector, as in Eq. 8.

$$\hat{p}_n = \alpha_m^{-1}(k) \cdot \hat{p}'_n \tag{8}$$

Where $\hat{p}$ is a reconstructed version of p, after dimensionality reduction has been applied.

To ensure the parameters are in a consistent format for each of the dimensionality scaling algorithms, a scale normalisation procedure is applied using Eq. 9, where during the training process, the p_{min} and p_{max} represent the minimum and maximum values for each parameter (given in Table 1), and q_{min} and q_{max} represent 0 and 1. During the testing process, these values are exchanged, such that q_{min} and q_{max} represent the minimum and maximum values for each parameter and p_{min} and p_{max} represent 0 and 1.

$$\rho_n = \frac{(p_n - q_{min})(p_{max} - p_{min})}{q_{max} - q_{min}} + p_{min} \tag{9}$$

Additionally, a sorting algorithm was used to place the three mid-band filters in ascending order based on their centre frequency. This prevents normalisation errors due to the frequency ranges allowing the filters to be rearranged by the user.

3.2. Parameter Space Mapping

Once the filters have been weighted by the audio signal, the mapping from 13 EQ variables to a two-dimensional subspace can be accomplished using a range of dimensionality reduction techniques. We start by evaluating the performance of three commonly used algorithms for data reduction by training them with weighted parameter space data and measuring the variance. Conversely, the reconstruction process is less common due to the nature of dimensionality reduction. We evaluate the efficacy of three multivariate regression-based techniques at mapping two-dimensional interface variables to a vector of EQ parameters. This is done by approximating functions using the weighted parameter data and measuring the reconstruction error. Finally, we evaluate a stacked autoencoder model of data reduction, in which the parameter space is both reduced and reconstructed in the same algorithm, we are then able to detach the reconstruction (decoder) stage for the testing process.

Dimensionality reduction is implemented using the following techniques: *Principal Component Analysis* (PCA), a widely used method of embedding data into a linear subspace of reduced dimensionality, by finding the eigenvectors of the covariance matrix, originally proposed by [14]; *Linear Discriminant Analysis* (LDA), a supervised projection technique that maps to a linear subspace whilst maximising the separability between data points that belong to different classes (see [15]); *Kernel PCA* (kPCA), a non-linear manifold mapping technique, in which eigenvectors are computed from a kernel matrix as opposed to the covariance matrix, as defined by [16]. As LDA projects the data-points onto the dimensions that maximise inter-class variance for C classes, the dimensionality of the subspace is set to $C - 1$. This means that in a binary classification problem, such as ours, we need to reconstruct the second dimension arbitrarily. For each of the other algorithms, we select the first 2 variables for mapping, and for the kPCA algorithm, the feature distances are computed using a Gaussian kernel.

The parameter reconstruction process was implemented using the following techniques: *Linear Regression* (LR), a process by which a linear function is used to estimate latent variables; *Natural Neighbour Interpolation* (NNI), a method for interpolating between scattered data points using Voronoi tessellation, as used by [11] for a similar application; *Support Vector Regression* (SVR), a non-linear kernel-based regression technique (see [17]), for which we choose a Gaussian kernel function.

An autoencoder is an Artificial Neural Network with a topology capable of learning a compact representation of a dataset by optimising a matrix of weights, such that a loss function representing the difference between the output and input vectors is minimised. Autoencoders can then be cascaded to form a network and initialised using layer-wise pre-training, commonly using Restricted Boltzmann Machines (RBMs), followed by backpropagation, leading to a complex nonlinear mapping between parameter spaces. This approach has proven to be successful for data compression [18] due to its ability to reconstruct high-dimensional data via a reduced feature subset. In our model, the symmetrical network consists of three hidden layers, with 13 units in the input and output layers, and two units in the central layer. The remaining hidden layers have nine units, and sigmoidal activation functions were used for each node. After the weights were pretrained with the RBMs, further optimisation was performed using back propagation and stochastic mini-batch gradient decent, with a batch size of 10 and a learning rate of 0.1.

P:	0	1	2	3	4	5	6	7	8	9	10	11	12	μ	σ
PCA-LR	0.676	0.180	0.186	0.101	0.036	0.530	0.184	0.024	0.283	0.148	0.023	0.501	0.108	0.229	0.040
LDA-LR	0.229	0.086	0.270	0.061	0.045	0.207	0.117	0.031	0.306	0.097	0.041	0.356	0.124	0.151	0.011
kPCA-LR	0.135	0.069	0.149	0.048	0.041	0.152	0.081	0.028	0.137	0.084	0.030	0.127	0.110	0.091	0.002
PCA-SVR	0.460	0.245	0.262	0.179	0.036	0.433	0.350	0.031	0.403	0.249	0.028	0.365	0.109	0.242	0.024
LDA-SVR	0.219	0.136	0.272	0.102	0.043	0.250	0.174	0.040	0.281	0.102	0.037	0.317	0.126	0.162	0.009
kPCA-SVR	0.126	0.063	0.151	0.047	0.038	0.149	0.075	0.027	0.144	0.088	0.031	0.131	0.104	0.090	0.002
PCA-NNI	0.515	0.354	0.294	0.597	0.028	0.504	0.527	0.052	0.374	0.591	0.024	0.488	0.406	0.366	0.044
LDA-NNI	0.391	0.363	0.333	0.200	0.051	0.314	0.305	0.097	0.294	0.222	0.066	0.396	0.188	0.248	0.015
kPCA-NNI	0.132	0.075	0.168	0.051	0.038	0.170	0.087	0.025	0.154	0.109	0.034	0.132	0.107	0.099	0.002
SAe	0.074	0.077	0.129	0.058	0.045	0.168	0.082	0.022	0.128	0.103	0.027	0.094	0.115	0.086	0.002

Table 2: Mean reconstruction error per parameter using combinations of dimensionality reduction and reconstruction techniques. The final two columns show mean (μ) and variance (σ) per parameter across all techniques. The model with the lowest mean reconstruction error (Stacked Autoencoder) is highlighted in grey.

4. RESULTS AND DISCUSSION

Parameter reconstruction error: We measure the reconstruction error for each combination of dimensionality reduction and reconstruction techniques by computing the mean squared error between predicted and actual parameter states, across all training examples. To do this, we use *K*-fold cross validation with $k = 100$ iterations, and a test partition size of 10% (80 training examples). The mean error for each technique is then calculated and the variance is found per-parameter, as shown in Table 2. As highlighted in the table, the stacked autoencoder model outperforms all combinations of reduction and reconstruction algorithms, with a mean reconstruction error of 0.086. Similarly, the autoencoder equals the lowest parameter variance measurement from all of the evaluated models, showing the consistency in high-dimensional parameter reconstruction. This is a desirable characteristic as it demonstrates that the problem of loading-bias towards a smaller subset of parameters does not exist in the system, which may cause unresponsive filter parameters during the testing phase.

Class Separation: To measure the extent to which the class separation (*warm* and *bright*) has been preserved in low-dimensional space, we measure the 2 sided Kullback-Leibler Divergence (KLD) between classes, after each of the dimensionality reduction techniques have been applied to the dataset. This allows us the measure the relative entropy between the class-distributions, in two-dimensional space. The KLD measurements show the LDA technique exhibited the highest degree of separation with a score of 1.98, whilst the autoencoder performed similarly, with score of 1.63. Conversely, the PCA-based techniques performed less favourably, with kPCA and PCA exhibiting 1.08 and 1.03 respectively. The class-labelled two-dimensional spaces are shown in Figure 5, along with the dimensions of maximal variance and class centroids.

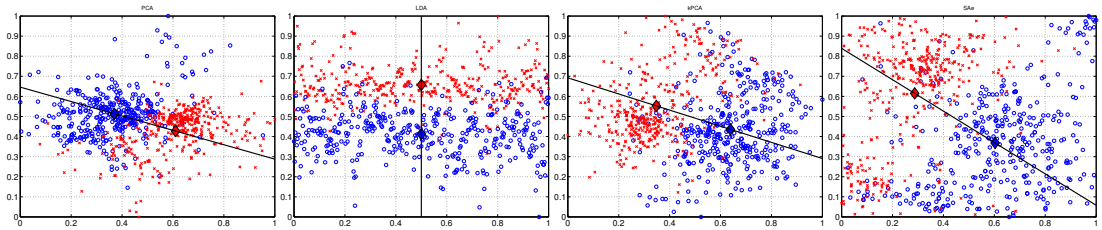
As LDA is a supervised technique, with the sole purpose of preserving discriminatory dimensions, thus maximising the variance between cluster centroids, it is expected that the class discrimination outperforms the other evaluated models. The autoencoder however performs similarly using an unsupervised algorithm, with no prior introduction to class labels. As LDA projects out 2 class data onto a 1 dimensional plane, a 2 class problem such as this (*warm* and *bright*) lacks the additional interface variability. This is shown in Figure 5b, where the spacing of points along the x-axis is arbitrarily based upon their sequential order. The dimension of maximal variance in this case is often aligned vertically, however this can be re-orientated by adding bias values to the data.

Parameter Weighting: To evaluate the effectiveness of the signal-specific weights, we measure the cross entropy of each dimensionality reduction technique before and after the weights have been applied. By doing so, we are able to observe the behaviour of the (*warm* and *bright*) classes in adaptive and non-adaptive reduced dimensionality space. The results show that three of the four techniques exhibit a positive change after signal weighting, with the autoencoder being the highest at +24%, followed by LDA at +19% and PCA at +6%, whereas the cross entropy of the kPCA technique exhibited a small negative change at -4%. Overall the separability of the data were shown to increase with a mean KLD improvement of 11%, suggesting the weighting process is a valuable attribute in the model. This reinforces the views of [4] and [5], who both suggest perceived timbral features such as *brightness* and *warmth* vary with attributes of the input signal such as $f0$ and perceived loudness.

4.1. Discussion

The results suggest that the stacked autoencoder method of reducing and reconstructing the parameter space provides both high reconstruction accuracy and class preservation in low-dimensional space. To implement the equaliser, the auto encoder's weighting matrix is optimised using the musical semantics data, and the decoder weights ($W' \in \mathbb{R}^{<13\times2>}$) are applied to new input data, using $f(W_n^T X + b_n)$, where f is a non-linear activation function and b is an intercept term. This maps a two-dimensional input vector to a 13 dimensional parameter vector, which can then be scaled to the audio spectrum using the inverse weights vector and scale-normalised, as shown in the testing phase of Figure 3.

Whilst the parameter reconstruction of the autoencoder is sufficiently accurate for our application, it is bound by the intrinsic dimensionality of the data, defined as the minimum number of variables required to accurately represent the variance in lower dimensional space. For our *bright*/*warm* parameter-space data, we can show that this intrinsic dimensionality requires three variables, when computed using Maximum Likelihood Estimation as defined by [19]. As our application requires a two-dimensional interface, this means the reconstruction accuracy is limited. To demonstrate this further, the reconstruction accuracy increases to 99.15% when the same autoencoder model is cross-validated with three variables in the hidden layer. This intrinsic dimensionality often dictates the variable reduction process, such as in [18].



(a) PCA (b) LDA (c) kPCA (d) SAe

Figure 5: Two-dimensional parameter-space representations using four data reduction techniques. The red data points are taken from parameter spaces described as bright and the blue points are described as *warm*. The solid black lines through the centroids mark dimensions of maximal variance in the data.

5. CONCLUSION

We have presented a model for the modulation of equalisation parameters using a two-dimensional control interface. The model utilises a stacked autoencoder to modify the dimensionality of the input data, and a weighting process that adapts the parameters to the LTAS of the input audio. We show that the autoencoder outperforms traditional models such as PCA and LDA in terms of reconstruction accuracy, and preserves timbral class discrimination in low-dimensional space. Whilst the model is limited by the inherent dimensionality of the data, we are able to achieve 92% reconstruction accuracy and 1.98 KLD separability on a dataset of 800 samples.

6. ACKNOWLEDGEMENTS

The work of the first author is supported by The Alexander S. Onassis Public Benefit Foundation.

7. REFERENCES

[1] R. Stables, S. Enderby, B. De Man, G. Fazekas, and J. D. Reiss, "SAFE: A system for the extraction and retrieval of semantic audio descriptors," in *ISMIR*. ISMIR, 2014.

[2] M. Sarkar, B. Vercoe, and Y. Yang, "Words that describe timbre: A study of auditory perception through language," in *Proceedings of the 2007 Language and Music as Cognitive Systems Conference*, 2007, pp. 11–13.

[3] J. W. Beauchamp, "Synthesis by spectral amplitude and "brightness" matching of analyzed musical instrument tones," *Journal of the Audio Engineering Society*, vol. 30, no. 6, pp. 396–406, 1982.

[4] E. Schubert and J. Wolfe, "Does timbral brightness scale with frequency and spectral centroid?," *Acta Acustica united with Acustica*, vol. 92, no. 5, pp. 820–825, 2006.

[5] J. Marozeau and A. de Cheveigné, "The effect of fundamental frequency on the brightness dimension of timbre," *Journal of the Acoustical Society of America*, vol. 121, no. 1, pp. 383–387, 2007.

[6] J. M. Grey, "Multidimensional perceptual scaling of musical timbres," *Journal of the Acoustical Society of America*, vol. 61, no. 5, pp. 1270–1277, 1977.

[7] A. Zacharakis, K. Pastiadis, J. D. Reiss, and G. Papadelis, "Analysis of musical timbre semantics through metric and non-metric data reduction techniques," in *Proceedings of the 12th International Conference on Music Perception and Cognition*, 2012, pp. 1177–1182.

[8] T. Brookes and D. Williams, "Perceptually-motivated audio morphing: Brightness," in *Proceedings of the 122nd Convention of the Audio Engineering Society*, 2007.

[9] A. Zacharakis and J. Reiss, "An additive synthesis technique for independent modification of the auditory perceptions of brightness and warmth," in *Proceedings of the 130th Convention of the Audio Engineering Society*, 2011.

[10] M. Cartwright and B. Pardo, "Social-EQ: Crowdsourcing an equalization descriptor map," in *ISMIR*, 2013, pp. 395–400.

[11] S. Mecklenburg and J. Loviscach, "subjEQt: Controlling an equalizer through subjective terms," in *CHI-06*, 2006, pp. 1109–1114.

[12] A. T. Sabin and B. Pardo, "2DEQ: an intuitive audio equalizer," in *Proceedings of the 7th ACM Conference on Creativity and Cognition*, 2009, pp. 435–436.

[13] T. Brookes and D. Williams, "Perceptually-motivated audio morphing: Warmth," in *Proceedings of the 128th Convention of the Audio Engineering Society*, 2010.

[14] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, no. 6, pp. 417, 1933.

[15] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[16] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[17] H. Drucker, C. J. C. Burges, , L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," *Advances in neural information processing systems*, vol. 9, pp. 155–161, 1997.

[18] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[19] E. Levina and P. J. Bickel, "Maximum likelihood estimation of intrinsic dimension," in *Advances in neural information processing systems*, 2004, pp. 777–784.

REAL-TIME EXCITATION BASED BINAURAL LOUDNESS METERS

Dominic Ward, Sean Enderby, Cham Athwal
DMT Lab
Birmingham City University
Birmingham, UK
dominic.ward@bcu.ac.uk

Joshua D. Reiss
C4DM
Queen Mary University of London
London, UK
josh.reiss@eecs.qmul.ac.uk

ABSTRACT

The measurement of perceived loudness is a difficult yet important task with a multitude of applications such as loudness alignment of complex stimuli and loudness restoration for the hearing impaired. Although computational hearing models exist, few are able to accurately predict the binaural loudness of everyday sounds. Such models demand excessive processing power making real-time loudness metering problematic. In this work, the dynamic auditory loudness models of Glasberg and Moore (J. Audio Eng. Soc., 2002) and Chen and Hu (IEEE ICASSP, 2012) are presented, extended and realised as binaural loudness meters. The performance bottlenecks are identified and alleviated by reducing the complexity of the excitation transformation stages. The effects of three parameters (hop size, spectral compression and filter spacing) on model predictions are analysed and discussed within the context of features used by scientists and engineers to quantify and monitor the perceived loudness of music and speech. Parameter values are presented and perceptual implications are described.

1. INTRODUCTION

The need to measure perceived loudness is imperative within fields such as psychoacoustics and audio engineering. In particular, accurate loudness alignment of complex stimuli is crucial when conducting controlled listening experiments and configuring multichannel systems [1]. In broadcasting, the loudness of a wide range of program material must be consistent yet natural to maintain a comfortable listening experience. In recent years, a number of researchers in the automatic mixing community [2, 3] have developed systems to automatically balance the loudness of multi-track content according to perceptual loudness features extracted from the audio.

Although there is a clear need for models of loudness, developers of real-time applications are often forced to sacrifice prediction accuracy. The purpose of this paper is to demonstrate how the dynamic loudness models proposed by Glasberg and Moore [4] and Chen and Hu [5] can be modified to obtain fast and efficient estimates of perceived loudness whilst maintaining agreement with empirical data. Before describing the models in detail, background information on single and multiband approaches to loudness prediction is given.

1.1. Loudness Models

Following the ITU broadcast standard [6] first published in 2006, subsequent EBU recommendation [7] and metering specifications [8] and related standards [9], a number of commercial loudness meters have appeared over the past few years. These meters employ single band loudness models which consist of a frequency weighting function, simulating the transmission response of the outer and middle ear, followed by an energy integrator. Such efficient estimators have proven to work well on broadcast material [10] and have been extended to capture the perception of single instruments [11] making them attractive candidates for real-time meters [12].

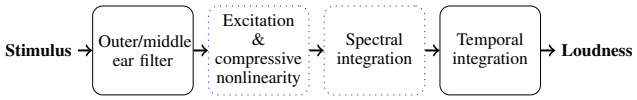


Figure 1: Block diagram of a simplified loudness model. Dotted boxes indicate stages used by multiband models.

In contrast to single band loudness models, sophisticated multiband models founded on empirical measurements of auditory filters and excitation patterns [13, 14, 15] have received little interest outside of psychoacoustics. As shown in Figure 1 multiband models also correct for the response of the outer and middle ear, but go a step further by accounting for the frequency selectivity of the cochlea. The signal is decomposed into frequency bands by means of an auditory filter bank. The output of the filter bank is called an excitation pattern and approximates the distribution of energy on the basilar membrane. When plotted as a function of filter centre frequency, the frequency scale is transformed to a perceptual scale with units Cams [16]. A compressive nonlinearity, modelled either as part of the filter bank or separately, describes the active mechanism of the auditory system. The specific loudness (SL) pattern represents compressed intensity as a function of filter centre frequency with units sones/Cam, where sones is the unit of loudness [17].

The spectral integration stage calculates the area under the SL pattern which estimates the instantaneous loudness (IL). This is the key aspect of multiband models that differentiate them from their single band counterpart. For narrow-band noise of constant intensity, loudness increases once its spectral bandwidth exceeds a critical bandwidth [13]. Single band models cannot account for this phenomenon known as spectral loudness summation. For time-varying sounds, the IL is smoothed by either a low-pass filter or empirically derived sliding window. Dynamic models output a loudness time series, from which various features can be computed, such as average or peak loudness. In short, multiband models have been shown to explain the observed variations in loudness caused by experimental factors such as sound intensity, frequency, spectral bandwidth and masking [18, 19].

The primary drawback of multiband models is their computational demand which consequently limits their application. This work explores this problem by optimising the excitation based loudness models of Glasberg and Moore [4] and Chen and Hu [5],

referred to as the GM02 and the CH12 respectively. Both models are refinements of Zwicker's stationary-sound loudness model [18] and have been extended to deal with time-varying sounds (see [20] for a detailed historical review). The key difference between Zwicker's and Glasberg and Moore's procedure is the way in which excitation patterns are computed and the equations defining the critical bandwidth as a function of frequency. The procedure used by the GM02 is based on more recent measures of auditory filters [14] and excitation patterns are directly derived from the output of the filter bank [21]. The revised model for steady-state sounds accurately predicts absolute thresholds, equal loudness contours and binaural loudness and became the basis for the 2007 ANSI standard [22].

The model of Chen and Hu is important because unlike its predecessors, the active process within the cochlea is modelled as an integral part of the filter bank [23] - there is no transformation from excitation to specific loudness. This is consistent with the idea that frequency selectivity and cochlear compression are the result of a single active process [24]. The authors of this recent loudness model also made improvements to the spectral decomposition procedure specified by Glasberg and Moore to correct for inflated loudness estimates.

1.2. Faster Multiband Models

Efficient implementations based on the core excitation model used by the GM02 have been established. In [25] a real-time loudness meter was developed with optimisations at the excitation transformation stage for increased efficiency. Although this model accounted for the loudness of short duration sounds it was less accurate in predicting the loudness of amplitude-modulated sounds compared to the GM02 as discussed in [26]. A binaural loudness device was established by [27] but the temporal integration and binaural summation stages were simple approximations when compared to more recent techniques [4, 28]. Other authors have concentrated on fast calculation of excitation patterns by using pruning techniques [29, 30] or making use of nonuniform spectral sampling [31, 32]. These proposals were specific to the GM02 and it is not clear how flexible they are, e.g. when a specified error limit is to be achieved. Finally, Burdiel et al. [33] discussed the computation savings of the GM02 obtained through parameterisation. Importantly, they demonstrated that reducing the number of analysis bands gave minimal impact on loudness estimates of musical sounds when compared to the effects of other parameters. Although real-time performance was obtained, that implementation was limited to monaural sounds and again specific to the GM02.

A full binaural dynamic loudness meter incorporating more recent loudness theory has not yet been established. It is the purpose of this study to present efficient parameterised implementations of both the GM02 and CH12 to facilitate real-time binaural loudness metering, with a focus on analysing the errors introduced. Additionally, the models have been realised as software plugins for use in digital audio workstations[1]

An overview of both models and implementational details of the meter is given in Section 2. In Section 3, the procedure used to analyse the performance of the models is provided and results are outlined in Section 4. Section 5 discusses key findings and draws comparisons to previous approaches, before concluding remarks and suggestions for future work are given in Section 6.

[1]Code available at https://github.com/deeuu/LoudnessMeters.

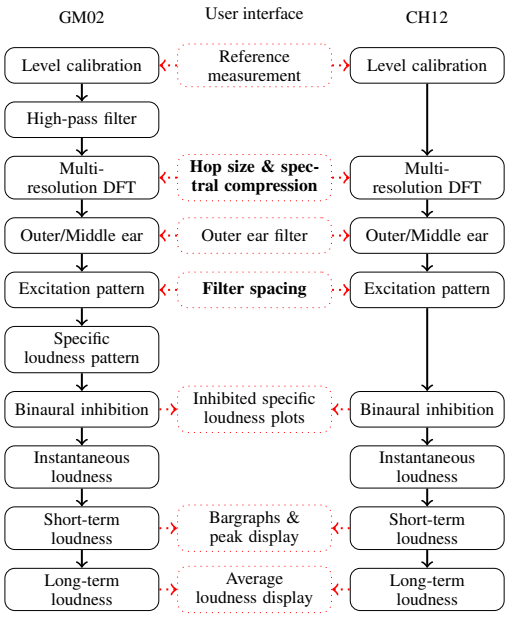


Figure 2: Block diagram of the loudness models and meter interface. Bold text indicates parameters investigated in this study.

2. MODELS AND METER IMPLEMENTATION

In this section the GM02 is described first, followed by the CH12. Developments for the meter are subsequently outlined before performance profile reports are summarised. The structure of both loudness models within the context of the binaural meter is given in Figure 2. The time-domain input signal can be either from a microphone, head and torso simulator (HATS) or digital recording (mono or stereo). A calibration stage is required to anchor the signal to a known reference. In order to reduce both latency and computational load, a third order Butterworth high-pass filter is applied to approximate the low-frequency response of the middle ear and the remaining outer and middle-ear filtering is conducted in the frequency domain [32]. This is in contrast to the 4096 order FIR used in the original model [4].

Short-term multi-resolution power spectra are obtained via six parallel FFTs. The six segments are obtained using Hann windows, with each successive window half the length of the previous. The windows are aligned at their centres via zero-padding. The FFTs are updated every millisecond, but the window hop size (time-step) was exposed as a free parameter. A spectral compression stage was added to reduce the number of components in the composite spectrum. This achieves a compact power spectrum by summing components into composite bins, the width of which increases with frequency. More specifically, the algorithm tries to maintain a constant spacing between components on the Cam scale. This reduces the number of components significantly in order to simplify the computation of excitation patterns as suggested in [25] and is similar to the enhancement used in the GM02 implementation by [34].

The power spectrum is then weighted according to the presentation of the stimulus [22]: free-field, diffuse-field or middle ear only (HATS). An additional BeyerDynamic DT990 option is available for simulating headphone presentation. The transfer function

of the headphone capsules were measured on an artificial ear.

The power spectrum is transformed to an excitation pattern using a bank of rounded exponential (roex) filters equally spaced from 1.8 to 38.9 Cams in steps of 0.1, yielding a total of 372 filters [22]. The shape of the filters are level dependent and have to be re-evaluated on every frame. In our implementation, filter shapes are computed according to [15], but a lookup table is employed to save on computing exponentials. As with hop size and spectral compression criterion the filter spacing was also made variable.

Following [22], a separate stage is required to transform the excitation pattern to an SL pattern, which represents compressed intensity within the cochlea. Unlike the published version of the model which approximates overall binaural loudness by summation of loudness across the ears, the procedure of [28] was incorporated to improve estimates of loudness for stimuli presented binaurally by modelling inhibitory interactions between the ears. The binaural inhibition block produces an inhibited SL pattern for each ear.

The IL in each ear is given by summing the inhibited SL values. The overall IL is the sum of the ILs at each ear, which is then smoothed by two cascaded asymmetrical low-pass filters to give the short-term loudness (STL) and long-term loudness (LTL) respectively. The time constants are given in Table 1 in accordance with [4] and [5].

Table 1: *Time constants (seconds) used by the two models.*

Model	STL		LTL	
	τ_A	τ_R	τ_A	τ_R
GM02	0.022	0.050	0.1	2
CH12	0.016	0.032	0.1	2

The CH12 differs from the GM02 in the following ways:

1. It uses twice the frequency resolution (largest window size of 128 ms compared to 64 ms for the GM02).
2. The pre-cochlear filter is performed entirely by weighting the power spectrum.
3. The middle ear transfer function follows [23].
4. It uses a set of double roex filters [23] equally spaced from 1.5 to 40.2 in steps of 0.1 Cams (388 filters).
5. No SL transformation is required.

Chen and Hu decided to double the frequency resolution because the DFT specification reported by Glasberg and Moore resulted in a spectral bandwidth that exceeded the critical bandwidth at 1 kHz. Consequently, the predicted total loudness of pure tones was larger than expected. Unlike the GM02, the filter bank used by the CH12 incorporates a compressive nonlinearity and thus the area under the *excitation* pattern is proportional to loudness. The output of the filter bank was scaled by the constant of proportionality in order to arrive at the SL pattern, which can then be fed to the binaural inhibition stage. Although the binaural procedure has been integrated into the GM02 [20], our modification to the CH12 to account for binaural loudness requires validation.

2.1. Modifications Specific to the Meter

As stated above, the pre-cochlear filter of the original GM02 has been modified to simplify computation and a binaural inhibition procedure has been incorporated into both models. The loudness meter displays inhibited SL on a logarithmic scale as a function of frequency on the Cam scale. Both left and right patterns are shown to facilitate visual comparison. In addition, bargraphs have been added to show the STL (in sones) for each ear, which are useful for comparing and aligning loudness between the ears. Finally, the overall LTL and peak STL are displayed for purposes of measuring global binaural loudness. A third number box shows the average unweighted sound pressure level (SPL) which is useful for system calibration.

The user interface, shown in Figure 3, also features two additional screens which can be activated using the appropriate buttons. The first is a settings screen for configuring the models in accordance with the exposed parameters highlighted in bold in Figure 2. The second is a calibration window, allowing the user to enter SPL measurements and estimate calibration gains.

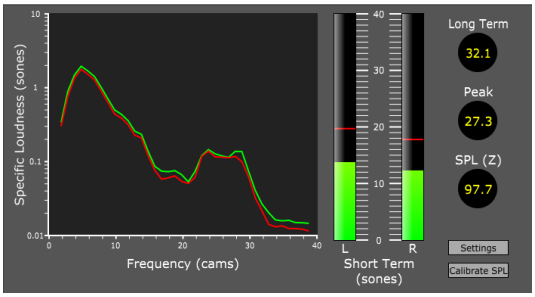


Figure 3: *Loudness meter user interface.*

2.2. Profiling the Models

The three parameters that govern the speed-accuracy trade-off are: hop size (R), spectral compression criterion (α) and filter spacing (β). A program to extract the loudness from 30 stereo sounds was developed for both models, each configured with their respective default parameter set (see Section 3). The sample-based profiler OProfile[2] was used to identify the most computationally intensive parts of each model. The programs were run on an idle Intel Core 2 Quad CPU Q8200 operating at 2.33 GHz with 4 GB RAM running Ubuntu 14.04 over 10 executions to increase the sample size. The profile reports are summarised in Table 2.

Table 2: *Program profile report - absolute sample count (and percentage of total samples collected).*

Process	GM02	CH12
Excitation transformation	2.7×10^8 (87.3)	1.3×10^8 (67.2)
FFTW [35]	1.4×10^7 (4.6)	3.4×10^7 (17.6)
Binaural Inhibition	8.1×10^6 (2.6)	8.8×10^6 (4.6)

For both models, the majority of CPU time was spent at the excitation transformation stage. Compared to the CH12, the GM02 collected a larger number of samples in this module, likely due to the added overhead of calculating variables pertaining to the roex filter shapes. Furthermore, the number of frequency points used by the CH12 is twice that of its predecessor and thus the time to compute the multi-resolution DFT is greater. The time required to compute the inhibited SL values was longer for the CH12, which can be attributed to the use of slightly more auditory filters.

[2] http://oprofile.sourceforge.net/

Importantly, the time complexity of the stages involved in excitation transformation of both models are linear functions of the number of frequency components (N) and auditory filters (M). For both models, the inner product of the auditory filters and power spectrum has the highest complexity of the excitation transformation module being $O(NM)$ (see [30]). Thus, although increasing the hop size reduces overall process time by computing fewer estimates, both spectral compression and filter spacing parameters target the bottleneck directly.

3. EVALUATION PROCEDURE

For each model, the parameter values were varied to reduce the computational workload and the errors between the reference and approximations were examined. The values used for each reference model and the approximations are given in Table 3 using Matlab notation ([start: delta: end]). The models were configured to process stereo recordings using the combined outer ear (free-field) [22] and middle ear [22, 23] transfer functions.

Table 3: *Reference and evaluation parameter values.*

Parameter	Reference	Approximation
R	1 ms	[2:2:16] ms
α	NA	[0.1:0.1:1] Cams
β	0.1 Cams	[0.25:0.25:4] Cams

The parameters were varied both independently and in combination. The effect of filter spacing was studied with and without interpolation applied to the excitation pattern. From experimenting with various input signals and interpolation schemes, cubic spline interpolation applied to the log excitation pattern was found to capture the smooth shape of the reference high-resolution patterns with good accuracy.

3.1. Stimuli

Each of the configurations were evaluated by extracting loudness features from 30 stereo tracks taken from the Sound Quality Assessment Material (SQAM) database [36] and assessing the error introduced. All tracks were sampled at 44.1 kHz. Each recording belonged to one of the following stimulus categories: single instruments, vocal, speech, solo, orchestra and pop music. Representative sound segments of the selected tracks were edited manually (average duration of 4.4 s (standard deviation (s.d.) 0.5 s)) and then peak calibrated to a random level between 84 and 94 dB SPL giving a spread of typical listening levels across the 30 sources (average RMS of 72.2 dB SPL (s.d. 4.4 dB) and peak of 89.1 dB SPL (s.d. 2.7 dB)).

3.2. Errors

The primary interest was to investigate the error between the reference and estimated short-term loudness time series of a given stimulus, denoted STL and $\hat{STL}$ respectively. The normalised root-mean-square error (nRMSE) between the reference and approximation is given by

$$nRMSE_{STL} = \frac{1}{STL_\mu}\sqrt{\frac{1}{F}\sum_{f=1}^{F}(STL_f - \hat{STL}_f)^2}, \quad (1)$$

where F is the number of non-zero frames and STL_μ is the average of the reference loudness. This normalisation was used to obtain a scale-free evaluation metric and corresponds to the coefficient of variation of the RMSE [37].

The following metric was used to measure the error introduced in the specific loudness patterns,

$$nRMSE_{SL} = \frac{1}{SL_\mu}\sqrt{\frac{1}{F \times M}\sum_{f=1}^{F}\sum_{m=1}^{M}(SL_{f,m} - \hat{SL}_{f,m})^2}, \quad (2)$$

where $SL_{f,m}$ is the total specific loudness in auditory filter m obtained by summing the corresponding inhibited specific loudness values in both ears. The RMS error is normalised by the average specific loudness value over time. In order to calculate the error at hop sizes greater than 1 ms a sample and hold procedure was applied to the approximated STL and SL time series.

It is also insightful to explore the perceptual implications of model performance for different parameter sets. For each stimulus, the level change (ΔL) required for equal overall loudness between the reference and approximated loudness predictions was estimated using the sone ratios obtained from three global loudness descriptors: average LTL (LTL_μ) [4], peak STL (STL_{pk}) [38] and the 95$^{\text{th}}$ percentile of the STL distribution (STL_{95}) [39]. Because of the nonlinear relationship between intensity level and loudness, an iterative procedure was employed to find the optimal gain required for equal loudness. More specifically, ΔL is obtained for each descriptor by minimising the sone ratio in decibels using a tolerance of 0.01 dB. For example, the level change required for equal loudness according to the peak STL is found by minimising the magnitude of

$$\epsilon = 10 \times \log_{10}\left(\frac{STL_{pk}}{\hat{STL}_{pk}}\right), \quad (3)$$

where STL_{pk} and $\hat{STL}_{pk}$ is the peak STL of the reference and approximation. For convenience, ΔL is referred to as the global loudness descriptor error and is expressed in decibels.

In addition to profiling the models, the computational savings are also analysed in terms of the reduction in the number of frequency components and auditory filters with respect to the reference quantities.

4. RESULTS

Figures 4, 5 and 6 show the nRMSE for parameters hop size (R), spectral compression criterion (α) and filter spacing (β). The base 10 logarithm of the stimulus errors (see ordinate) was taken in order to reduce positive skew in the error distributions (and thus biassing the arithmetic mean), as well as providing visual clarity throughout the plots. Data points are the arithmetic means of the log nRMSE across stimuli for either the STL or SL. The shaded areas surrounding the lines are the 95% confidence intervals of the means and were estimated using a percentile bootstrapping procedure based on 5000 samples [40]. These intervals have been corrected to eliminate between-stimuli variability which bias the sampling error in repeated-measures designs and are useful for observing patterns in the population means [41]. For example, the intervals shown in subplot (a) of Figure 6 were calculated according to a 2 (models) x 2 (interpolants) x 16 (filter spacings) within-stimulus design.

4.1. Hop Size

Increasing the hop size widens the interval between instantaneous loudness samples, giving rise to greater error in the time integrated

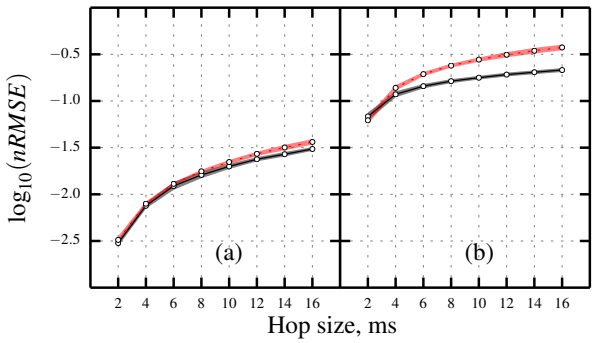


Figure 4: *Log normalised RMSE plotted as a function of hop size for (a) the STL and (b) SL. The data points are the arithmetic means of the log stimulus errors and shaded areas surrounding the lines represent the 95% confidence intervals of the estimates for the GM02 (black solid lines) and CH12 (red dotted lines).*

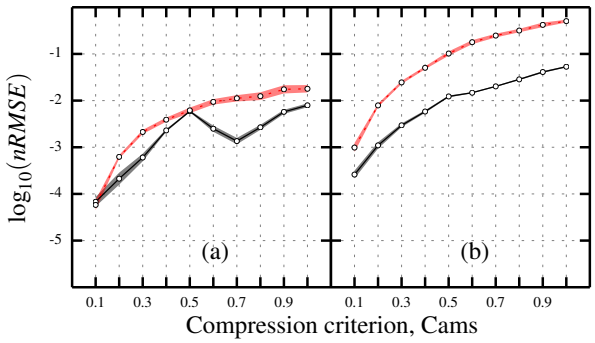


Figure 5: *Log normalised RMSE plotted as a function of compression criterion for (a) the STL and (b) SL. The data points are the arithmetic means of the log stimulus errors and shaded areas surrounding the lines represent the 95% confidence intervals of the estimates for the GM02 (black solid lines) and CH12 (red dotted lines).*

measurements. Although the STL error functions show a similar trajectory for both models, the errors appear to increase at a slightly faster rate for the CH12. A doubling of the hop size increases the geometric mean of the STL nRMSEs by an approximate factor of 2.1 for the GM02 and 2.2 for the CH12. Subplot (b) of Figure 4 shows the error introduced when SL patterns from previous processing frames are used as estimates for the current frame output by the reference model. The SL errors are higher than STL errors because they are instantaneous calculations whereas the STL is the combined result of integrating the SL patterns both in frequency and over time. For a hop size of 2 ms, the SL error is lower for the CH12, but rises above the GM02 at larger values, indicating that the CH12 sees greater variation in the auditory patterns over time.

4.2. Compression Criterion

For both models, spectral compression distorts the loudness patterns as shown in subplot (b) of Figure 5. This is to be expected, given that the input components are summed into sub-bands prior

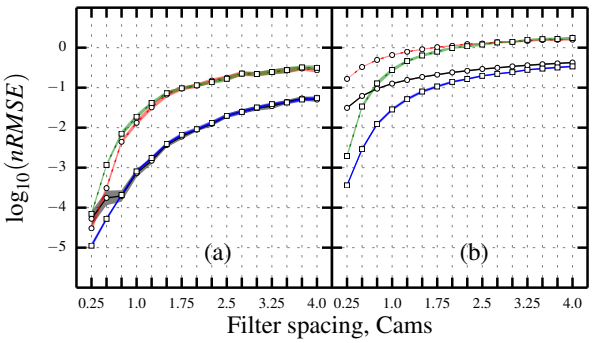


Figure 6: *Log normalised RMSE plotted as a function of filter spacing for (a) the STL and (b) SL. The data points are the arithmetic means of the log stimulus errors and shaded areas surrounding the lines represent the 95% confidence intervals of the estimates for the GM02 (black and blue solid lines) and CH12 (red and green dotted lines). The lines with square markers are the errors introduced when the approximated excitation patterns are interpolated at filter locations defined by the reference models.*

to auditory filtering. In general, the CH12 is more sensitive to spectral modifications and gives consistently larger errors in the SL domain where the average ratio of the nRMSE geometric means between the CH12 and GM02 was 9.1. The only criterion in which this model produced a lower average error (STL only) was 0.1 Cams although there is strong overlap between the two confidence intervals. For the GM02, summing the erroneous SL pattern leads to a non-monotonic STL error function; the error function shows a peak at 0.5 Cams and continues to increase above 0.7 Cams. This is an important consideration when optimising the GM02 based on spectral compression.

4.3. Filter Spacing

Figure 6 shows the STL and SL errors as a function of filter spacing, with interpolation type as parameter. With no interpolation, doubling the filter spacing increases the geometric mean of the STL nRMSEs by an approximate factor of 7.0 for the GM02 and 9.9 for the CH12, which is considerably higher than the relative increase in error when doubling the hop size. The STL error functions of the CH12 are especially steeper than those of the GM02 over 0.25-0.75 Cams, indicating that the former is more sensitive to modifications in this range. The only spacing in which the STL error of the CH12 was lower than the GM02 was 0.25 Cams.

The geometric mean of the STL nRMSEs, averaged across 0.25 and 0.5 Cams with cubic interpolation, was reduced by 74% for the GM02, though no such improvement was observed for the CH12. It is important to highlight that the nRMSE is a relative error metric; plotted data points below -3 translate to central RMS errors less than 0.1% of the mean loudness time series. The application of cubic interpolation to the SL patterns show an improvement in the approximations, though the benefit does not carry over to the STL measurements for the CH12.

4.4. The Bottleneck

The STL and SL errors have been presented for model parameters hop size (R), spectral compression criterion (α), and filter spac-

Table 4: *Number of components and filters (and percentage reduction) obtained with the optimal combination of compression criterion (α) and filter spacing (β) at $R = 1$ ms. The maximum global loudness descriptor error is also given.*

Model	α	$\hat{N}$	β	$\hat{M}$	max $\lvert\Delta L\rvert$ (dB)
GM02	0.7	63 (95 %)	1.25	30 (92 %)	0.08
CH12	0.3	148 (95 %)	0.5	78 (80 %)	0.11

Table 5: *Program profile report after optimisations.*

Process	GM02	CH12
FFTW [35]	1.4×10^7 (73.7)	3.4×10^7 (78.9)
Excitation transformation	1.6×10^6 (8.1)	1.6×10^6 (3.8)
Binaural inhibition	1.1×10^5 (0.6)	4.7×10^5 (1.1)

ing (β). When configured with the reference parameter sets (see Table 3), the models did not execute in real-time. The computational speed can be increased approximately two-fold by simply doubling the hop size, however Table 2 shows that the bottleneck resides in the excitation transformation stage. The findings indicate that initial values of the hop size introduce significantly larger STL error when compared to a range of α and β values.

Can a speedup factor greater than two with a lower average STL error than that introduced by a hop size of 2 ms be achieved by targeting the bottleneck directly?

The number of frequency components (N) was 1393 and 2971 for the GM02 and CH12. The number of auditory filters (M) was 372 and 388. Let $\hat{N}$ and $\hat{M}$ denote the number of components and filters resulting after increasing parameters α and β beyond the reference values. For each model, the error surface generated by different combinations of α and β (with $R = 1$ ms) was searched for all geometric average STL errors less than the error introduced by $R = 2$ ms alone. The target error was 0.3% for both models. All combinations which lead to average errors exceeding this threshold were discarded. Of the remaining parameter values, the combination that maximised the complexity reduction at the excitation transformation stage ($1 - (\hat{N}\hat{M})/(NM)$) was selected.

Table 4 shows the performance of the models with the optimal parameter sets, operating with a hop size of 1 ms. The average STL nRMSEs were less than 0.25% and lead to a total complexity reduction of approximately 99% for both models. Indicative of worst-case performance, the final column shows the maximum absolute level change required for equal loudness between the reference and approximation predictions across all stimuli for all three features. The largest deviation from the global loudness predictions given by the reference models was 0.08 dB for the GM02 and 0.11 dB for the CH12.

The profile reports associated with the optimal parameter sets are shown in Table 5. It can be seen that the processing time consumed by the excitation transformation has been significantly reduced (by two orders of magnitude for the GM02) and consequently, the bottleneck has shifted to the computation of the multiresolution DFT. The mean stimulus speedup factor (CPU time of the reference divided by the CPU time of the approximation averaged across ten executions) was 16.09 (s.d. 0.01) for the GM02 and 4.55 (s.d. 0.02) for the CH12. This demonstrates that large speedups can be achieved before having to resort to incrementing the hop size independently. It should be noted that these parameters were selected based on a target STL nRMSE at $R = 2$ ms. Running the same optimisation procedure using the LTL yields slightly lower values for α and β because this feature is less affected by hop size and thus lowers the target error to 0.1% for both models.

Table 6: *Performance of the optimised models at four hop sizes (R) in terms of maximum global loudness descriptor error and minimum speedup factor across all stimuli. The first and second row of each cell correspond to the GM02 and CH12 respectively. The row in bold indicates the only parameter set in which real-time processing was not achieved.*

R	max $\lvert\Delta L\rvert$ (dB)			Speedup
	LTL_μ	STL_{pk}	STL_{95}	
1	0.05	0.08	0.08	15.4
	0.06	**0.03**	**0.11**	**4.4**
2	0.05	0.07	0.10	30.5
	0.06	0.03	0.11	8.7
4	0.12	0.81	0.22	63.2
	0.08	0.21	0.20	17.7
8	0.29	2.57	0.24	124.8
	0.12	1.54	0.42	34.4

Finally, maximum level differences required for equal loudness readings across all sounds according to three global loudness descriptors are given in Table 6 for the two models configured with the parameter values listed in Table 4. The performance is evaluated at four hop sizes and worst-case speedup factors across all program executions and all stimuli are given. The row in bold indicates that the CH12 did not achieve real-time performance on the computer used in this study at the reference hop size.

5. DISCUSSION

The two dynamic loudness models presented in this study have been parameterised by hop size, spectral compression and filter spacing. The effect of hop size on STL error was comparable in both models and a hop size of 2 ms introduced a larger STL error than using the lower compression and filter values tested with a hop size of 1 ms. As shown in Table 6, large hop sizes can be detrimental when estimating the loudness of short duration sounds, which rely on accurate estimates of peak loudness [4]. Zwicker [38] highlighted the importance of maximum loudness for quantifying the perceived loudness of impulse and speech sounds. Indeed, of all the stimuli tested, the castanets recording (track 27 of the SQAM CD) occurred most frequently in the top 5% of the STL nRMSE distributions across all hop sizes tested for both models. Compared to the peak STL descriptor, the LTL and percentile measures stay within 0.5 dB of the reference values for hop sizes up to 8 ms.

A compressed spectrum can be incorporated into both models to reduce the number of components whilst attempting to maintain sufficient energy in the auditory filters to preserve the loudness density. The grouping of component intensities does introduce some error in the auditory patterns, the extent of which is dependent on the bandwidth used to average the spectrum. The GM02 requires a wider bandwidth to achieve a similar percentage reduction in the number of components as the CH12. This is because the GM02 uses half the frequency resolution as the CH12. When $\alpha = 0.1$ the compression criterion is satisfied above 264 Hz for the CH12 but not until 764 Hz for the GM02. In combination with the fact that most musical sounds are dominant in the low-mid frequency range, this explains why the average errors of the CH12 shown in Figure 5 are generally higher.

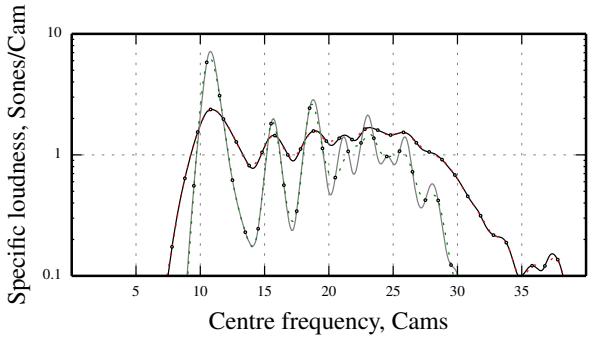


Figure 7: *Specific loudness pattern of a segment of track 42 (Accordion) from the SQAM CD using the GM02 (solid black line) and CH12 (solid grey line) with the reference parameter sets. Data points correspond to values obtained by sampling at increments of 1 Cam and dotted lines show the result of cubic interpolation.*

The choice of filter spacing determines the accuracy of the excitation and SL patterns. Too few filters per critical band can be severe for tonal signals where peak excitations that dominate perception are unlikely to be captured. Large intervals can also have a marked effect on broadband sounds in that the area under the SL pattern is estimated from an undersampled excitation pattern. The results showed that the CH12 is more sensitive to changes in filter spacing and introduced larger deviations from the target data compared to the GM02. This can be attributed to differences in DFT frequency resolution and filter bank architecture. For the CH12, the DFT mainlobe width is narrower and so the importance of filter spacing is greater. This issue was investigated by running the GM02 with twice the frequency resolution at multiple filter spacings (with the reference reconfigured) and indeed larger errors resulted. The SL patterns output by the two reference models were also compared across stimuli and it was observed that those of the CH12 had much larger peak-to-valley ratios compared to the GM02 (see Figure 7). The excitation pattern must be sampled more frequently to accurately capture peak excitations. Considering that the RMSE metric emphasises large deviations, it is of no surprise that the SL errors produced by the CH12 are notably high. As shown in Figure 7, large excitatory oscillations generated by the CH12 can be problematic when interpolating low-resolution patterns. For this particular analysis frame, the instantaneous loudness predicted by the reference GM02 was 30.86 sones and, with cubic interpolation applied to the excitation pattern sampled at 1 Cam intervals, 30.87. The CH12 predicted 25.41 sones and, for the same approximation, 24.68. Although the CH12 did not benefit from interpolation in terms of the integrated loudness, Figure 7 indicates that cubic interpolation should be preferred over linear interpolation when displaying SL versus frequency.

The parameter sets listed in Table 4 provide a good guideline for speeding up the models based on the initial bottleneck. Both models see a significant reduction in execution time by combining spectral compression with filter spacings above 0.1 Cams. Further speedup factors of ~ 2 can be obtained by doubling the hop size. The results in Table 6 indicate that for a hop size of 2 ms, combined with the parameters values given in Table 4, real-time performance can be achieved with estimates of global loudness within 0.15 dB of the reference predictions for a range of music and speech. Considering that discrimination thresholds for intensity can be as low as 0.2 dB for pure tones and 0.5 dB for most broadband noises [39], it is unlikely that there will be noticeable differences between identical stimuli aligned in loudness by the reference models and proposed approximations. More importantly, [42] found the average reproducibility of subjective relative loudness judgements involving *different* program material to be 1.24 dB. This suggests that greater error may be tolerable when the faster implementation is used for purposes of balancing the loudness of typical program material.

The authors in [33] showed that, based on STL errors of the GM02, a filter spacing of 1 Cam could be used to achieve real-time performance on their test machine. In the current study, a finer range of parameter values have been explored and for the GM02, greater computational savings and lower average STL nRMSE can be obtained using an interval of 0.75 Cams combined with a compression criterion of 0.2. Furthermore, the real-time proposal in [33] included a frequency domain weighting function to replace the FIR filter used by Glasberg and Moore. However, it has been shown [32] that spectral weighting is problematic at low-frequencies, mainly in terms of absolute threshold predictions, and hence why the implementation here incorporates a high-pass filter to improve the response. Our implementation can be applied to both laboratory and everyday sounds without having to switch filtering techniques. Finally, the authors in [31] employed the Hopping Goertzel DFT algorithm to optimise the parallel FFTs, which was then used by [32] to simplify the calculation of excitation patterns by means of nonuniform spectral sampling. Although faster performance can be obtained with that implementation at very low hop sizes, larger time-steps are unlikely to yield substantial computational gains due to the processing requirements of the Hopping Goertzel DFT. Preliminary experiments showed that for the GM02, nonuniform spectral sampling introduced larger average STL error compared to all compression criteria tested in this study.

6. CONCLUSIONS

Efficient implementations of the loudness models of Glasberg and Moore [4] and Chen and Hu [5] have been developed for the purpose of real-time binaural loudness metering. The meter exposes key parameters that govern model performance, enabling the user to control the speed-accuracy trade-off to meet the demands of a given application. The bottleneck of both models was identified as the transformation from power spectrum to excitation pattern. By incorporating a perceptually inspired method to obtain a compact spectrum and experimenting with different filter spacings, the total complexity of the excitation transformation stage was reduced by 99%, yielding significant speedup in execution time. The largest deviation was 0.11 dB when measured in terms of short-term and long-term loudness metrics commonly used to quantify overall loudness of time-varying sounds. The effect of hop size was also investigated both independently and in combination with the parameter sets that lead to high computational savings. In the latter case, a hop size of 2 ms was required to achieve real-time performance.

Future work is required to validate the predictions of the binaural models against empirical data, especially on sounds with complex spectro-temporal behaviour. Derivation of parameter sets that achieve maximum speedup whilst maintaining error limits to match required perceptual criteria would also be useful. Our meter does not currently simulate the effects of cross-talk between loudspeakers in multichannel setups typically used by sound engineers. This would require a more sophisticated filtering stage that accommodates appropriate head-related transfer functions.

7. REFERENCES

[1] N. Zacharov, "An Overview of Multichannel Level Alignment," in *Proceedings of the 15th AES International Conference on Audio, Acoustics & Small Spaces*, 1998.

[2] S. Mansbridge, S. Finn, and J. D. Reiss, "Implementation and Evaluation of Autonomous Multi-track Fader Control," in *Proceedings of the 132nd Audio Engineering Society Convention*, 2012.

[3] D. Ward, J. D. Reiss, and C. Athwal, "Multi-track mixing using a model of loudness and partial loudness," in *Proceedings of the 133rd Audio Engineering Society Convention*, 2012.

[4] B. R. Glasberg and B. C. J. Moore, "A Model of Loudness Applicable to Time-Varying Sounds," *Journal of the Audio Engineering Society*, vol. 50, no. 5, pp. 331–342, 2002.

[5] Z. Chen and G. Hu, "A Revised Method of Calculating Auditory Excitation Patterns and Loudness for Time-varying Sounds," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2012.

[6] ITU, "ITU-R BS.1770-3: Algorithms to measure audio programme loudness and true-peak audio level," Tech. Rep., International Telecommunication Union, 2012.

[7] EBU, "EBU-R 128: Loudness normalisation and permitted maximum level of audio signals," Tech. Rep., European Broadcast Union, 2014.

[8] EBU, "EBU-Tech 3341: Loudness Metering: 'EBU Mode' metering to supplement loudness normalisation in accordance with EBU R 128," Tech. Rep., European Broadcast Union, 2011.

[9] ATSC, "ATSC Recommended Practice: Techniques for Establishing and Maintaining Audio Loudness for Digital Television," Tech. Rep., Advanced Television Systems Committee, 2013.

[10] E. Skovenborg and S. H. Nielsen, "Evaluation of Different Loudness Models with Music and Speech Material," in *Proceedings of the 117th Audio Engineering Society Convention*, 2004.

[11] P. D. Pestana, J. D. Reiss, and A. Barbosa, "Loudness Measurement of Multitrack Audio Content using Modifications of ITU-R BS.1770," in *Proceedings of the 134th Audio Engineering Society Convention*, 2013.

[12] E. Skovenborg and S. H. Nielsen, "Real-time Visualisations of Loudness Along Different Time Scales," in *Proceedings of the 10th International Conference on Digital Audio Effects*, 2007.

[13] E. Zwicker, G. Flottorp, and S. S. Stevens, "Critical Band Width in Loudness Summation," *Journal of the Acoustical Society of America*, vol. 29, no. 5, pp. 548–557, 1957.

[14] R. D. Patterson, "Auditory Filter Shapes Derived with Noise Stimuli," *Journal of the Acoustical Society of America*, vol. 59, no. 3, pp. 640–654, 1976.

[15] B. R. Glasberg and B. C. J. Moore, "Derivation of auditory filter shapes from notched-noise data," *Hearing Research*, vol. 47, pp. 103–138, 1990.

[16] W. Hartmann, *Signals, Sound and Sensation*, Springer, 1997.

[17] S. S. Stevens, "The Measurement of Loudness," *The Journal of the Acoustical Society of America*, vol. 27, no. 5, pp. 815–829, 1955.

[18] E. Zwicker and B. Scharf, "A Model of Loudness Summation," *Psychological review*, vol. 72, no. 1, pp. 3–26, 1965.

[19] B. C. J. Moore and B. R. Glasberg, "A Revision of Zwicker's Loudness Model," *Acta Acustica united with Acustica*, vol. 82, no. 2, pp. 335–345, 1996.

[20] B. C. J. Moore, "Development and Current Status of the "Cambridge" Loudness Models," *Trends in Hearing*, vol. 18, pp. 1–29, 2014.

[21] B. C. J. Moore and B. R. Glasberg, "Suggested formulae for calculating auditory-filter bandwidths and excitation patterns," *Journal of the Acoustical Society of America*, vol. 74, no. 3, pp. 750–753, 1983.

[22] ANSI, "ANSI S3.4-2007 Procedure for the Computation of Loudness of Steady Sounds," Tech. Rep., American National Standards Institute, 2007.

[23] Z. Chen, G. Hu, B. R. Glasberg, and B. C. J. Moore, "A new method of calculating auditory excitation patterns and loudness for steady sounds," *Hearing research*, vol. 282, no. 1-2, pp. 204–15, 2011.

[24] L. Robles and M.A. Ruggero, "Mechanics of the mammalian cochlea," *Psychological review*, vol. 81, pp. 1305–1352, 2001.

[25] M. A. Stone, B. C. J. Moore, and B. R. Glasberg, "A Real-Time DSP-Based Loudness Meter," in *Contributions to Psychological Acoustics*, A. Schick and M. Klatte, Eds., pp. 587–601. Bibliotheks- und Informationssystem der Universität Oldenburg, 1997.

[26] B. C. J. Moore, B. R. Glasberg, and M. A. Stone, "Why Are Commercials so Loud ? - Perception and Modeling of the Loudness of Amplitude-Compressed Speech," *Journal of the Acoustical Society of America*, vol. 51, no. 12, pp. 1123–1132, 2003.

[27] O. Tuomi and N. Zacharov, "A Real-Time Binaural Loudness Meter," in *Proceedings of the 139th meeting of the Acoustical Society of America*, 2000.

[28] B. C. J. Moore and B. R. Glasberg, "Modeling Binaural Loudness," *The Journal of the Acoustical Society of America*, vol. 121, no. 3, pp. 1604–1612, 2007.

[29] H. Krishnamoorthi, V. Berisha, and A. Spanias, "A Low-Complexity Loudness Estimation Algorithm," in *Proceedings of the 2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008.

[30] H. Krishnamoorthi, A. Spanias, and V. Berisha, "A Frequency / Detector Pruning Approach for Loudness Estimation," *IEEE Signal Processing Letters*, vol. 16, no. 11, pp. 997–1000, 2009.

[31] R. J. Cassidy and J. O. Smith III, "Efficient Time-Varying Loudness Estimation via the Hopping Goertzel DFT," in *Proceedings of the 50th Midwest Symposium on Circuits and Systems*, 2007, pp. 421–422.

[32] D. Ward, C. Athwal, and M. Kokuer, "An Efficient Time-Varying Loudness Model," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2013.

[33] E. Burdiel, L. Vetter, A. J. R. Simpson, M. J. Terrell, A. McPherson, and M. Sandler, "Real-time Implementation of Glasberg and Moore's Loudness Model for Time-Varying Sounds," in *Proceedings of the 133rd Audio Engineering Society Convention*, 2012.

[34] D. Cabrera, S. Ferguson, and E. Schubert, "PsySound3: An integrated environment for the analysis of sound recordings," in *Proceedings of ACOUSTICS*, 2008.

[35] M. Frigo and S. G. Johnson, "The Design and Implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.

[36] EBU, "EBU-Tech 3253 Sound Quality Assessment Material recordings for subjective tests," Tech. Rep., European Broadcast Union, 2008.

[37] A. T. Reddy, *Applied Data Analysis and Modeling for Energy Engineering and Scientists*, Springer, 2011.

[38] E. Zwicker, "Procedure for calculataing loudness of temporally variable sounds," *Journal of the Acoustical Society of America*, vol. 62, no. 3, pp. 675–682, 1977.

[39] H. Fastl and E. Zwicker, *Psychoacoustics: Facts and Models*, Springer, third edition, 2007.

[40] K. Singh and M. Xie, "Bootstrap : A Statistical Method," Tech. Rep., Rutgers University, USA, 2010.

[41] F. O'Brien and D. Cousineau, "Representing Error bars in within-subject designs in typical software packages," *The Quantitative Methods for Psychology*, vol. 10, no. 1, pp. 56–67, 2014.

[42] G. A. Soulodre, M. C. Lavoie, and S. G. Norcross, "The Subjective Loudness of Typical Program Material," in *Proceedings of the 115th Audio Engineering Society Convention*, 2003.

EFFECT OF AUGMENTED AUDIFICATION ON PERCEPTION OF HIGHER STATISTICAL MOMENTS IN NOISE

Katharina Vogt, Matthias Frank, Robert Höldrich

Institute of Electronic Music and Acoustics,
University of Music and Performing Arts
Graz, Austria
{vogt, frank, hoeldrich}@iem.at

ABSTRACT

Augmented audification has recently been introduced as a method that blends between audification and an auditory graph. Advantages of both standard methods of sonification are preserved. The effectivity of the method is shown in this paper by the example of random time series. Just noticeable kurtosis differences are effected positively by the new method as compared to pure audification. Furthermore, skewness can be made audible.

1. INTRODUCTION

Sonification is still a relatively young field building up a canon of methodologies. Two of the standard approaches to sonification are audification and auditory graphs. Recently, the method of augmented audification has been introduced [1] as a seamless interpolation between these approaches. Augmented audification combines well-known techniques of signal processing: single-sideband modulation utilizing the Hilbert transform, and an exponential frequency modulation. This method allows to control both the mean position in the frequency range *and* the bandwidth of the sonification by free model parameters, independently to the rate of the data display. Fundamental properties of audification are conserved, notably the compact temporal support and the translation of high frequency content of the data into transient events in the sound. Furthermore, data sets can be explored interactively at various time scales and in different frequency ranges.

Frauenberger et al. [2] studied the audification of random data time series with varying higher order momentums. The third moment, skewness, is a measure for the asymmetry of the probability density function. The fourth moment is called kurtosis and serves as a measure of the peakedness of the distribution. In the study it has been shown that participants could discriminate a kurtosis difference in the audification of above 5. Qualitatively, they reported an increase of roughness with rising kurtosis. Distinguishing different values of skewness could not be proven. This is not surprising, as skewness depends strongly on the mean of the data series which results in an indiscernible DC value, and furthermore, human hearing does not perceive the "sign" of a signal.

In the following section, basic properties and limits of audification and auditory graphs are reviewed shortly. Sec. 3 introduces the signal processing algorithms of augmented audification[1]. Sec. 4 discusses the use case of random time series and the generation of data sets used in the listening experiment, which is discussed in Sec. 5. Finally, we conclude and give an outlook to further research.

[1] Accompanying sound examples can be found at: http://iaem.at/Members/vogt/augmentedaudification

2. COMPARISON OF AUDIFICATION AND AUDITORY GRAPHS

Audification, on the one hand side, has been defined by Kramer in 1991 (cited in [3], p. 186): "a direct translation of a data waveform to the audible domain". Today, audification has many "puristic" supporters within the sonification community who are in favor of a direct playback of data, with the only adjustable factor being the playback rate (see [4]). This factor also determines the typical size of data sets that are apt for audification, thus they are reasonably large.

A crucial advantage of audification is the following: by conserving the time regime of the data signal, audifications of real physical processes are usually broad-band with a pronounced proportion of high frequencies during rapid transients. In the task of identifying natural sounds, e.g., the attack of musical instruments or speech signals, the transient parts of the signal provide features for human hearing that serve as the basis for pattern recognition tasks. The same is true for audified signals.

In general, audification suffers from a trade-off between the macroscopic time scale and the frequency range of the relevant information. The ideal audification signal has relevant auditory gestalts within time and frequency regimes that can be well-perceived by the human auditory system. Some suggestions have been made to cope with this trade-off and argue in favor of a more adjustable audification paradigm (e.g., [5]).

On the other hand side, *auditory graphs* mostly sonify reasonably small data sets, up to a few hundred data points, often in a pitch-time-display. Thus, with respect to data size, we may find typical data of auditory graphs on the very other end of the scale than audification (e.g., on opposite sides on the sonic design space map, [6]). Obvious benefits of auditory graphs are the straightforward analogy to visual graphs, which make them intuitively understandable, at least for sighted users. The Sonification Sandbox [7] was possibly the largest effort to develop a general tool for auditory graphs. From the experience with the toolbox it can be concluded that most real-world sonification applications need a more flexible adjustment between the data set and the auditory graph.

Flowers [8] discussed promises and pitfalls of auditory graphs. He suggested that successful displays follow these strategies: numeric values should be pitch-encoded; the temporal resolution of human audition shall be exploited; loudness changes in a pitch mapped stream shall be manipulated in order to provide contextual cues and signal-critical events; distinct timbres shall be chosen in order to minimize stream confusions and unwanted perceptual grouping; and time in sound shall be used to represent time in the data.

Augmented audification allows to blend between audification and auditory graphs (in the form of a pitch-time display). Advantages of both methods can be combined, overcoming the time-scale trade-off of pure audification.

3. AUGMENTED AUDIFICATION: THE MODEL

For explaining Augmented Audification (henceforth: AugAudif), we start with a basic audification. We assume a data set $x(n)$ with $n = 1..N$ data points and a playback rate or sampling frequency f_p, i.e., f_p data points are displayed per second. The rendering over a D/A converter with a reconstruction filter leads to a continuous signal $x(t)$ with a bandwidth B between zero and $1/2f_p$ Hz. If the playback rate is as low as a few hundred data points per second, the resulting sound will be in a low frequency range, where the human ear is not very sensitive.

3.1. Frequency Shifting

Therefore, as a first step, we perform frequency shifting by a single-side-band modulation. Using a Hilbert transform (see, e.g., [9]), the original audification signal $x(t)$ becomes the complex-valued signal $x_a(t)$,

$$x_a(t) = x(t) + j\ \mathcal{H}\{(x(t)\} \tag{1}$$

with the imaginary constant j. This analytical signal can be written using a real-valued envelope $env(t) = |x_a(t)|$ modulated by a phasor with the instantaneous phase $\theta(t) = angle[x_a(t)]$:

$$x_a(t) = env(t)\ e^{j\theta(t)}. \tag{2}$$

Performing a frequency shift by Δf and taking the real part of this signal leads to a SSB-modulated sound signal $x_{SSB}(t)$:

$$\begin{aligned} x_{SSB}(t) &= Re\left[env(t)\ e^{j(\theta(t)+2\pi\Delta f t)}\right] &(3)\\ &= x(t)\cos(2\pi\Delta f t) - \mathcal{H}\{x(t)\}\sin(2\pi\Delta f t). &(4)\end{aligned}$$

The spectrum of the analytical signal, which contains (only non-negative) frequencies between zero and B Hz, is shifted to the range between Δf and $(\Delta f + B)$. Discarding the imaginary part re-builds a symmetric spectrum.

The frequency shift Δf is a free parameter of the method, which helps to yield a perceptually optimal frequency range of the sonification, i.e., somewhere within the range of 100 Hz and 2 kHz. If $\Delta f = 0$, there is no difference to a pure audification.

In the case of high playback rates, e.g., $f_p = 20$ kHz, which lead to a broad-banded audification, a frequency shift of $\Delta f = 100$ Hz hardly changes the overall signal, but might make low frequency components of the signal audible, as the spectrum is now shifted to the range between 100 Hz and 10.1 kHz.

A strong frequency shift, especially in combination with slow playback rates, results in a very narrow-banded signal which might be problematic from a perceptual point of view. The frequency shift squeezes the original - conceptually infinite - pitch range to a range of $(\Delta f + B)/\Delta f$. For example, if $f_p = 200$ Hz, hence the bandwidth of the primary audification signal is max. 100 Hz, and the spectrum is shifted by $\Delta f = 500$ Hz, the resulting bandwidth is 500 to 600 Hz. Speaking in musical terms, all frequency components of the original data stream are now concentrated within a minor third. Fluctuations of such narrow-banded signals are difficult to perceive.

3.2. Exponential Frequency Modulation

Therefore the method is extended by modulating the frequency of the phasor of the analytic signal $x_a(t)$. The instantaneous frequency of the modulator, $f_i(t)$, exponentially encodes the numeric data values of $x(t)$ as pitch, following to Flowers' recommendations:

$$f_i(t) = 2^{cx(t)} f_0. \tag{5}$$

f_0 is the carrier frequency and c a freely choosable parameter that controls the magnitude of the modulation: Setting $c = 0$ results in a constant instantaneous frequency of the frequency modulation which is then independent of the data values $x(t)$. This results in a pure frequency shift as described in Sec. 3.1. Setting $c = 1$ leads to a transposition of one octave higher/ lower for signal values $x(t) = +1/-1$. The value of c has to be chosen carefully depending of signal amplitude and bandwidth to prevent aliasing resulting from strong FM sidebands.

For the AugAudif, the parameter of frequency shift is used as carrier frequency, $f_0 \equiv \Delta f$. Integrating over the instantaneous frequency results in the instantaneous phase $\phi_i(t)$, which serves as a phase modulating term for the analytical signal.

$$\phi_i(t) = \int_0^t 2\pi\ \Delta f\ 2^{cx(\tau)} d\tau. \tag{6}$$

This leads to the complete model of Augmented Audification:

$$\begin{aligned} x_{AA}(t) &= Re\left[env(t)\ e^{j\ (\theta(t)+\phi_i(t))}\right] &(7)\\ &= x(t)\cos(\phi_i(t)) - \mathcal{H}\{x(t)\}\sin(\phi_i(t)). &(8)\end{aligned}$$

The model is controlled by two freely choosable model parameters, Δf and c, that can be set according to the explorative goals of the sonification.

4. USE CASE: STATISTICAL PROPERTIES OF RANDOM DATA TIME SERIES

4.1. Data generation

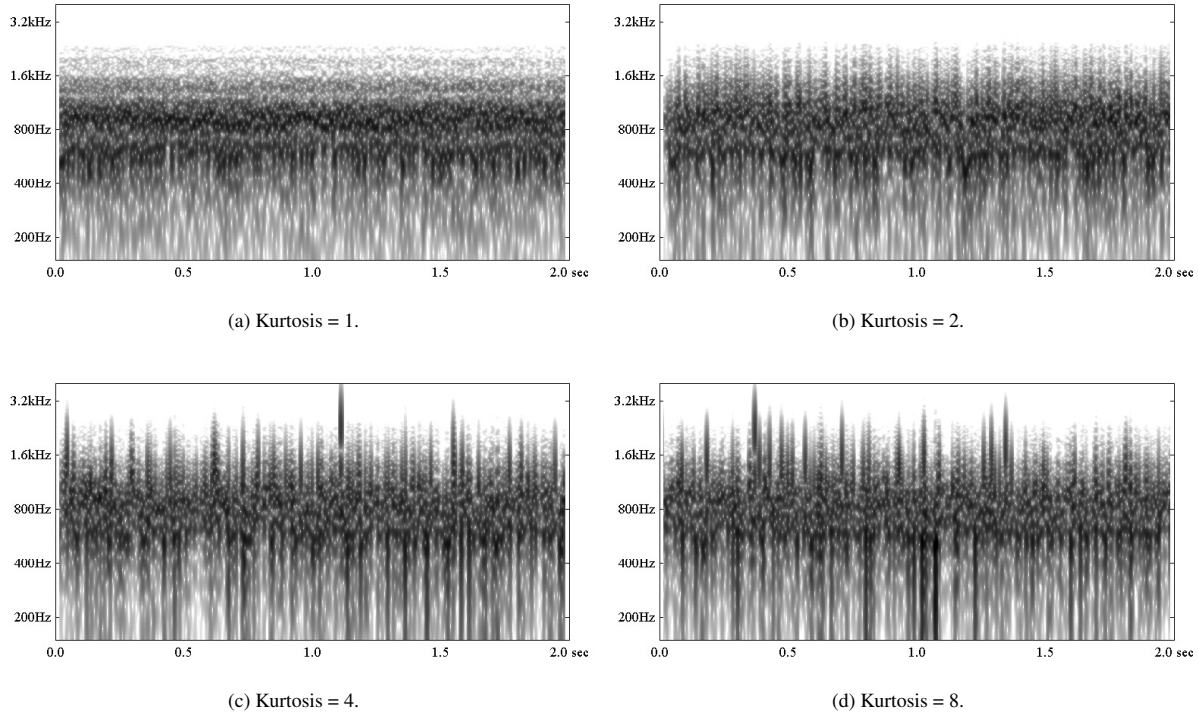
As a challenging use case for augmented audification, we generated random time series with different statistical properties. Frauenberger et al. [2] used a Levy alpha-stable distribution to create their data sets. The Levy alpha-stable distribution is not defined for a distribution parameter (skewness) of < 2 and for kurtosis values that are smaller than the Gauss distribution. Therefore, we implemented a type IV Pearson distribution for the generation of the examples in this section and the formal listening experiment in Sec. 5.

Neglecting a normalization constant, the type IV Pearson distribution is given as

$$p(x) = \left[1 + \left(\frac{x-a}{b}\right)^2\right]^{-m} * e^{-\nu * \arctan\left(\frac{x-a}{b}\right)} \tag{9}$$

where a is the location parameter (mean value), b the scale parameter (variation), and the shape parameters m (kurtosis) and ν (skewness).

Noise samples of this distribution have been generated in Matlab for a mean value of zero and given standard deviation ($= 0.1$), but with various values for skewness and kurtosis. The distributions have been generated at a low sampling rate, thus limited in bandwidth to a cutoff frequency f_c.



(a) Kurtosis = 1.

(b) Kurtosis = 2.

(c) Kurtosis = 4.

(d) Kurtosis = 8.

Figure 1: *Spectrograms of the 4 consecutive sounds in Soundfile 1. The fixed parameters are: skewness = 0;* $f_p = 800$ *Hz;* $c = 5/12$*;* $\Delta f = 600$ *Hz.*

Then, they have been interpolated to obtain the standard sampling rate of 44.1 kHz. Due to the random character of the generation process itself and the effects caused by the subsequent interpolation scheme, the actual statistical moments of the noise samples deviated from the target values. Therefore, an iterative procedure has been adopted to select noise samples with the intended preset values for skewness and kurtosis.

The actual values for skewness and kurtosis were calculated from the zero-mean samples $x(i)$, $i = 1..N$:

$$skew(x) = \frac{\mu_3}{\sigma^3} = \frac{\frac{1}{N}\sum_{i=1}^{N} x(i)^3}{\left(\frac{1}{N}\sum_{i=1}^{N} x(i)^2\right)^{3/2}}, \quad (10)$$

$$kurt(x) = \frac{\mu_4}{\sigma^4} = \frac{\frac{1}{N}\sum_{i=1}^{N} x(i)^4}{\left(\frac{1}{N}\sum_{i=1}^{N} x(i)^2\right)^{2}}. \quad (11)$$

4.2. Sound examples

A first, informal listening of the authors of this paper showed a much lower threshold for discriminating kurtosis and even the ability to defer different values of skewness using AugAudif as compared to direct audification. Two sound examples shall illustrate the effect of the method:

Soundfile 1 is an AugAudif of time series with a white noise spectrum, zero skewness and varying kurtosis (consecutively 1, 2, 4, and 8). The playback rate of the data has been chosen as 800 Hz. Fig. 1 shows the spectrograms of these 4 sounds. (All spectrograms were calculated using a 4096 sample Hanning window and are displayed on a logarithmic frequency scale up to 10 kHz.)

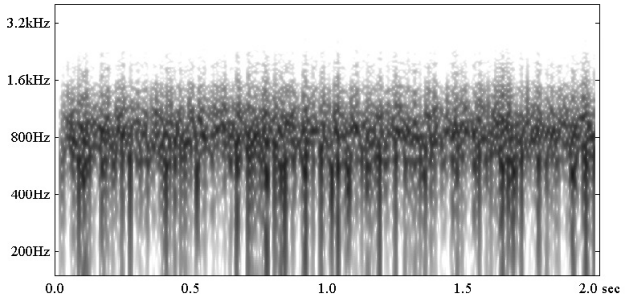
Soundfile 2 is an AugAudif of time series with constant kurtosis but varying skewness. The parameters are the same as above ($f_p = 800$ Hz; $c = 5/12$; $\Delta f = 600$ Hz). Kurtosis is set at 12, while skewness takes the values of -2, 0, and 2, respectively. The spectrograms shown in Fig. 2 clearly indicate the asymmetric frequency excursions due to the different skewness.

5. EXPERIMENT

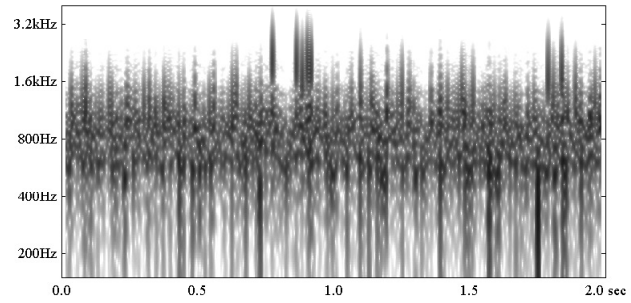
The above method has only recently been introduced. Evidence for its usefulness have to be found on an inter-subjective level. Therefore, a formal listening experiment was performed to test the following hypotheses:

- The just noticeable differences (JNDs) of kurtosis of random data time series may be lowered when treated with augmented audification as compared to direct audification.
- The skewness of random data time series may be detected when treated with augmented audification as compared to direct audification.

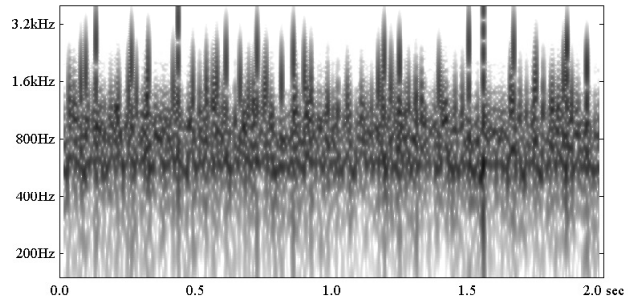
Additionally, the experiment should investigate the influence of different parameters for augmented audification on JNDs of kurtosis, i.e. provide estimates for optimal parameter settings in different frequency regimes.



(a) Skewness = -2.



(b) Skewness = 0.



(c) Skewness = 2.

Figure 2: *Spectrograms of the 3 consecutive sounds in Soundfile 2. The fixed parameters are: kurtosis = 12;* $f_p = 800$ *Hz;* $c = 5/12$*;* $\Delta f = 600$ *Hz.*

5.1. Setup, Method, and Conditions

The experiment evaluated the JNDs in skewness and kurtosis for 19 conditions shown in Table 1 using an adaptive 1-up/2-down triangle test [10]. Thus, the participant's task was to identify the odd one within a triplet of sounds. As all sound files had to be rendered before the experiment, the skewness/ kurtosis values have been quantized in discrete steps. For all parameters (conditions, quantized skewness/ kurtosis values) 5 different realizations of random time series have been created. During the experiment, the realizations have been selected in order that no triplet contained identical sound files (but two realizations with the same parameters, and one other). In this way, the experiment considered the perception of variation within realizations with same parameters as well. Participants could listen to the sounds as often and in any order they wished.

As a reference value for all conditions with varying kurtosis, a kurtosis of 3 was taken, which corresponds to Gaussian noise; within mathematics it is common to use this reference (with the notions of platykurtic or leptokurtic distributions referring to a negative or positive excess related to a kurtosis of 3). Furthermore it seems plausible from an evolutionary point of view that the auditory system is "gauged" to the normal distribution as a frequent case in natural systems.

In order to minimize the number of iterations in the adaptive procedure, a variable step size has been used: each procedure started with a comparison of a parameter set with largest available skewness/ kurtosis value and one with the reference value. The step size (difference of skewness/ kurtosis value to reference value) was halved after each two correct answers in a row and after each wrong answer. The new step size was calculated from the old one, plus/ minus half of the actual step size. In case of all correct answers, and due to the quantization of the skewness/ kurtosis values, the smallest step size could be achieved after 7 iterations. The maximum number of iterations has been limited in dependence of the number of quantization steps, cf. Table 1. This measure should prevent the participant from fatigue and account for cases where s/he could not discriminate between different settings at all.

In addition to the psychometric measurement of the JNDs, participants were asked to verbally describe the perceived differences in an accompanying questionnaire. For each condition, the participants should do so right after the first triplet, when the differences were as large as possible.

For playback of the sound files, an RME Multiface and Beyerdynamic DT-770 Pro headphones have been employed. A total of 10 subjects (all experienced listeners with hearing loss of less than 15dB, 8 of them part of a trained expert listening panel [11, 12, 13]) participated in the experiment. On average, the whole experiment took 115 min. and was divided into two parts.

Table 1: *Conditions in the experiment with variable or fixed values for skewness and kurtosis (reference values in brackets), number of quantization steps, and maximum number of iterations.*

con.	f_c/Hz	Δf/Hz	c	kurtosis	skew.	steps	iter.
1	10000	-	-	var(3)	0	12	25
2	5000	-	-	var(3)	0	12	25
3	500	-	-	var(3)	0	12	25
4	100	-	-	var(3)	0	12	13
5	5000	150	5/12	var(3)	0	12	25
6	500	150	5/12	var(3)	0	12	25
7	100	150	5/12	var(3)	0	12	25
8	5000	600	5/12	var(3)	0	12	25
9	500	600	5/12	var(3)	0	12	25
10	100	600	5/12	var(3)	0	12	25
11	5000	600	3/12	var(3)	0	12	25
12	500	600	3/12	var(3)	0	12	25
13	100	600	3/12	var(3)	0	12	25
14	5000	-	-	var(1.27)	0	16	25
15	5000	-	-	var(2.1)	0	14	25
16	5000	-	-	var(3.5)	0	10	20
17	5000	-	-	var(4.5)	0	8	18
18	500	600	5/12	8	var(0)	5	13
19	5000	600	5/12	8	var(0)	5	13

5.2. Quantitative Results

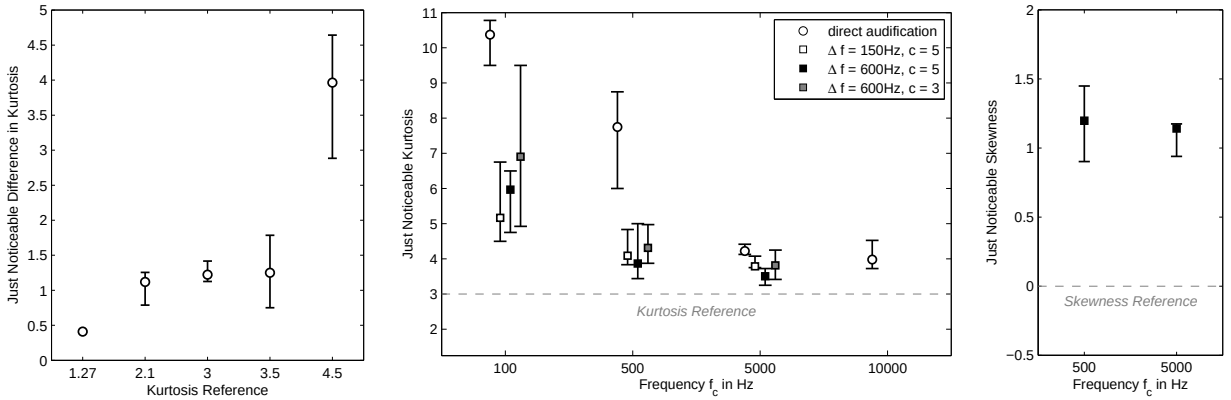
The above procedure lead to a standard oscillating between two values of smallest step size that supposedly lie around the JND. The thresholds for the just noticeable values were calculated as the minimum of the averages over the last 4 and 6 reversals [14]. For condition 4, this calculation did not always converge. Some participants never answered two times correctly in straight succession. In this case, the just noticeable value has been set to the maximum available value for this condition. Similarly, most participants could always perceive the smallest available difference in condition 14. Thus, the just noticeable kurtosis difference was set to the smallest available difference. In comparison to our results, the actual just noticeable difference might hence be larger for condition 4 and smaller for condition 14. However, this would not change the general statements of our study.

Figure 3a shows the different just noticeable differences in kurtosis for direct audification in dependence of the reference kurtosis. Analysis of variance and Kruskal-Wallis tests revealed a significant increase towards higher reference kurtosis values ($p < 0.001$). This also holds for the variance of the just noticeable difference. However, only the neighboring conditions 14/15 and 16/17 yield significantly different means ($p < 0.001$) and median values ($p \leq 0.002$). The results indicate varying JNDs for different values of reference kurtosis. However, a modeling of the difference threshold departing from Weber's law could not be deduced from the data. Nevertheless, a first attempt to establish a psychophysical scale for kurtosis of band-limited noise in direct audification can be found in the Appendix.

Figure 3b encompasses the central findings of the experiment. Just noticeable kurtosis decreases for higher f_c using either direct or augmented audification. For all evaluated cutoff frequencies f_c, AugAudif yields smaller just noticeable kurtosis as compared to direct audification. The different variations of augmented audification perform similarly well, although $c = 3/12$ tends to be the worst variation for $f_c = 100$ Hz and $\Delta f = 600$ Hz, $c = 5/12$ tends to be the best variation for $f_c = 5000$ Hz. The statistical analysis revealed that kurtosis sensitivity increases significantly for higher f_c ($p < 0.001$) in direct audification.
All neighboring conditions yield significantly different mean ($p < 0.001$) and median values ($p \leq 0.001$), except for conditions 2/1 ($p \geq 0.54$). Just noticeable kurtosis also decreases for AugAudif ($p \leq 0.04$), although conditions 6/5 yield no significantly different mean ($p = 0.15$) and median values ($p = 0.18$). It can be argued, that the magnitude of the difference between augmented and direct audification depends on f_c in relation to Δf: while $\Delta f = 150$ Hz is a large difference compared to $f_c = 500$Hz, it is relatively small compared to $f_c = 5000$ Hz. As might be expected, the conditions where $c = 3/12$ tend to lead to worse results than the ones with $c = 5/12$. This can be argued from the sensitivity of the human hearing, which, in general, decreases for lower frequencies. For $f_c = 100$ Hz, all variations of augmented audification yield significantly lower median values ($p \leq 0.002$) compared to direct audification. However, comparing the different variations among each other, none of them show significant differences in means ($p \geq 0.11$) and median values ($p \geq 0.1$). Condition 13 suffers from a remarkably strong inter-subjective variation.

The authors would have expected that direct audification saturates from the low frequency range towards the most sensitive hearing range (even if the value for high frequencies, i.e. the 10 kHz regime, should exhibit a degradation following this argument, which is not reflected in the data). Similarly, for $f_c = 500$ Hz, all variations of augmented audification yield significantly lower median values ($p \leq 0.001$) in comparison to direct audification. Again, there are no significant differences between the means ($p \geq 0.31$) or median values ($p \geq 0.29$) of the different variations. For $f_c = 5000$ Hz, although all augmented audifications yield significantly smaller means ($p \leq 0.07$) and median values ($p \leq 0.07$), condition 5 achieves significantly smaller means/medians ($p = 0.02$) than condition 8. However, the difference between conditions 8 and 11 is not significant ($p = 0.11$). The optimal parameter setting within this experiment has been reached for condition 8, with $f_c = 5000$ Hz, $\Delta f = 600$ Hz, and $c = 5/12$. In percentages, this corresponds to a frequency shift in the order of 8% of the mean frequency range, and a factor c equalling roughly one percent of the frequency shift.

Considering pure audification of band-limited noise signals, the sensitivity for kurtosis depends strongly on the bandwidth. The increase of sensitivity up to a cutoff frequency of a few kHz can be well-explained by general sensitivity properties of the human auditory system. The kurtosis sensitivity shows a saturation effect above 5 kHz which can be attributed to the fact that high kurtosis values result in a very peaked signal and therefore in a broadband "synchronization" of energy portions up to very high frequencies, but the further information contained in the extended bandwidth cannot be exploited by the auditory system due to masking and the reduced sensitivity above 5 kHz. The JND of kurtosis also depends on the kurtosis reference. Nevertheless, our attempts to establish a Weber's model for the JND dependence on the stimulus' value failed for various reasons. First, such models require an absolute zero value for the stimulus, but kurtosis lacks a natural and/or plausible zero point although there is a mathematically defined minimum value of 1. (A kurtosis of 1 could be achieved because of the band-limited nature of the noise signals.) Secondly, the perceptive attribute correlated with varying kurtosis is not a prothetic continuum like sound intensity or brightness, as is also supported by the qualitative analysis.



(a) direct audification at cutoff frequency $f_c = 5000$ Hz (conditions 14,15,2,16,17)

(b) direct and augmented audification with different Δf, c (conditions 4,7,10,13, 3,6,9,12, 2,5,8,11, 1)

(c) augmented audification with $\Delta f = 600$ Hz, $c = 5$ (conditions 18,19)

Figure 3: *Median and corresponding 95% confidence intervals of just noticeable skewness/ kurtosis with regard to reference skewness/ kurtosis.*

The results in Figure 3c show that variation of skewness is audible. However, f_c has no significant effect on just noticeable skewness variation ($p = 0.59$ for means, $p = 0.49$ for median values). The reference case for direct audification has not been studied in this experiment, but in informal listening of the authors prior to the experiment and by Frauenberger et al. [2]. For detecting skewness, the pure frequency shift should not lead to an improvement, whereas by the additional frequency modulation the asymmetric behavior of the distribution values becomes apparent.

5.3. Qualitative Results

The accompanying questionnaire was analyzed following [15] with regard to possible different criteria that the subjects may have used in different conditions. Furthermore the analysis should help defining common terms for perceptive attributes related to skewness and kurtosis.

In general, a very diverse set of terms have been mentioned: only two categories have been subsumed in the first place, these are terms related to the (in-)homogeneity of the noise and its frequency bandwidth. The category of (in-)homogeneity included, among others, the notions "static sound", "variation", "fluctuation", "outliers", or "(un-)evenness". Notions related to frequency bandwidth were "high/ low frequency", "f-spread" etc. Apart from these two groups of terms, subjects indicated 47 different adjectives describing sound attributes, plus another 19 noun groups indicating sound sources from a natural or technical context. These terms could not be further grouped: they consist of a quite extensive list of notions describing different noise-like sounds, often using colloquial words[2].

The terms were annotated (by numbers 1 to 47, the natural sound sources as N1 to N19) and grouped according to different experiments' parameters. Apart from the two term groups above, the top five were "knistern" (crinkle/ crackle), "britzeln" (colloquial term, something like crumbling), "blubbern" (bubbling), "körnig" (granular), and "knacksen" (crackle/ click). The most common sound sources cited are related to water drops and different types of rain, but a few participants were very creative here (e.g., *"aggressive electro-mosquitos"* or *"futuristic laser-bursting bubbles"*).

Cross-subject coincidences could be found in 11 out of the 18 conditions (defined by 4 or more participants using the same word/ category). The group category of (in-)homogeneity (4 times) or frequency bandwidth (one time) was used commonly the most often. "Knistern" (crinkle/ crackle) was the only other frequently shared term (used 4 times), but all in the 5000 Hz conditions, thus the term refers only to higher frequencies and cannot be generally applied; "blubbern" (bubbling) and "knacksen" (crackle/ click) were commonly used in one experiment each (conditions 5 and 8).

In general about half of the subjects use rather uniform vocabulary, whereas the other half indicated a wide spread of diverse terms. Therefore the applied method of counting common words has to be treated with caution. Still, a few findings are interesting. For pure audification, we find the most intersubjectively repeated categories, i.e., it can be argued that the augmented audification results in more varied sounds. At low frequency ($f_c = 100$Hz) the general category of (in-)homogeneity is by far most important (very common for 4 subjects). It can be argued that it is harder to differentiate sound properties in the low frequency range, where we are less sensitive. On the other hand, the higher the frequency the more varied the terms become, and the analysis will neither lead to coincidences of commonly used terms.

The factor c seems to not have a large influence, at least comparing $c = 3/12$ to $c = 5/12$. The indicated terms stayed the same for more than half of the subjects within the conditions with the same Δf and f_c but varying c. As skewness was only varied in two experiment settings, no different wording could be found there.

Overall, the above analysis supports to introduce "crackling" as the most general term to describe the influence of kurtosis in noise.

[2]Therefore the authors of this paper cannot provide a translated version of the original data in this paper; a specialized translator would be needed for this task. An example are the two German terms "knittern" and "knistern" (crinkle/ crackle), that are not only phonetically similar but have also a very similar meaning. Thus, it cannot be inferred that subject A has the same notion of "knistern" as subject B has of "knittern".

6. CONCLUSIONS AND OUTLOOK

Augmented audification allows to interpolate seamlessly between pure audification and an auditory graph in the form of a pitch-time-display. As opposed to pure audification, where only the playback rate can be changed, two more model parameters can be chosen independently in the augmented audification. One parameter, Δf, controls the magnitude of a frequency shift. The second, c, sets the excursion of the exponential frequency modulation (FM), i.e. pitch modulation.

The effectivity of augmented audification has been shown in this paper by a listening experiment. The two hypotheses given in Sec. 5 are supported by the results of this experiment. When treated with augmented audification as compared to direct audification, in random data time series both the JNDs of kurtosis may be lowered and skewness may be detected. Furthermore, qualitative data on the denotation of a perceptive attribute related to skewness and kurtosis have been collected and analyzed. The English word "crackling" seems to be a promising candidate for intersubjective naming.

Cautiously, the optimal parameter settings for augmented audification can be given by a frequency shift in the order of 8% of the mean frequency range, and a factor of c equalling roughly one percent of this frequency shift. In general, the method of augmented audification has to be studied by other examples than random time series.

The regime between $f_c = 500$ and 5000 Hz is most interesting for further research, in order to get more information on the saturation of the JNDs with rising frequency and how the perceptual function might be modeled. Aspects of interactivity in the research paradigm (e.g., the possibility to acoustically zoom into an audified data set) have to be examined further.

7. ACKNOWLEDGMENT

We would like to thank all participants for their sustained effort in the (noisy!) listening experiment.

8. REFERENCES

[1] Robert Höldrich and Katharina Vogt, "Augmented audification," in *Proc. of the International Conference on Auditory Display (ICAD)*, 2015.

[2] Christopher Frauenberger, Alberto deCampo, and Gerhard Eckel, "Analysing time series data," in *Proc. of the International Conference on Auditory Display (ICAD)*, 2007.

[3] Gregory Kramer, Ed., *Auditory Display. Sonification, Audification and Auditory Interfaces.*, Proceedings Volume XVIII. Santa Fe Institute, Studies in the Sciences of Complexity, 1994.

[4] Florian Dombois and Gerhard Eckel, "Audification," in *The Sonification Handbook*, T. Hermann, A. Hunt, and J.G. Neuhoff, Eds., chapter 12, pp. 237–272. Logos Publishing House, Berlin, 2011.

[5] David Worrall, *Sonification and Information – Concepts, Instruments and Techniques*, Ph.D. thesis, University of Canberra, 2009.

[6] Alberto de Campo, "Toward a data sonification design space map," in *Proc. of the International Conference on Auditory Display (ICAD)*, 2007.

[7] Benjamin K. Davison and Bruce N. Walker, "Sonification sandbox reconstruction: Software standard for auditory graphs," in *Proc. of the International Conference on Auditory Display (ICAD)*, 2007.

[8] John H. Flowers, "Thirteen years of reflection on auditory graphing: Promises, pitfalls, and potential new directions," *Proceedings of International Conference on Auditory Display*, 2005.

[9] Alan Oppenheim and Ronald Schafer, *Digital Signal Processing*, Prentice Hall, 1975.

[10] Harry Levitt, "Transformed up-down methods in psychoacoustics," *The Journal of the Acoustical Society of America*, vol. 49, no. 2B, pp. 467–477, 1971.

[11] Alois Sontacchi, Hannes Pomberger, and Robert Höldrich, "Recruiting and evaluation process of an expert listening panel," in *Fortschritte der Akustik, NAG/DAGA*, Rotterdam, 2009.

[12] Matthias Frank, Alois Sontacchi, and Robert Höldrich, "Training and guidance tool for listening panels," in *Fortschritte der Akustik, DAGA*, Berlin, 2010.

[13] Matthias Frank and Alois Sontacchi, "Performance review of an expert listening panel," in *Fortschritte der Akustik, DAGA*, Darmstadt, 2012.

[14] Sygal Amitay, Amy Irwin, David J. C. Hawkey, Justin A. Cowan, and David R. Moore, "A comparison of adaptive procedures for rapid and reliable threshold assessment and training in naive listenersa)," *The Journal of the Acoustical Society of America*, vol. 119, no. 3, pp. 1616–1625, 2006.

[15] Philipp Mayring, *Qualitative Inhaltsanalyse*, Beltz Verlag, 11. edition, 2010.

[16] GÖSta Ekman, "Weber's law and related functions," *The Journal of Psychology*, vol. 47, no. 2, pp. 343–352, 1959.

[17] Kenneth H Norwich and Willy Wong, "Unification of psychophysical phenomena: The complete form of fechner's law," *Perception & Psychophysics*, vol. 59, no. 6, pp. 929–940, 1997.

[18] John Z Sun, Grace I Wang, Vivek K Goyal, and Lav R Varshney, "A framework for bayesian optimality of psychophysical laws," *Journal of Mathematical Psychology*, vol. 56, no. 6, pp. 495–501, 2012.

[19] David Worrall, "The use of sonic articulation in identifying correlation in capital market trading data," in *Proc. of the International Conference on Auditory Display (ICAD)*, 2009.

Appendix:

In this supplementary discussion, we investigate the construction of a psychophysical scale for kurtosis of band-limited noise for the case of 5 kHz bandwidth and direct audification (see Table 1, conditions 14, 15, 2, 16, and 17).

In general, a monotonic psychophysical scale of sensation Ψ as a smooth function of the stimulus ϕ can be established from measurements of stimulus difference thresholds $\Delta\phi$ if a model for the subjective JND $\Delta\Psi$ is assumed. The most important JND models are Fechner' s law of a constant subjective JND, $\Delta\Psi = 1$, and Ekman's principle [16] which states a constant relative JND, $\Delta\Psi/\Psi = k$.

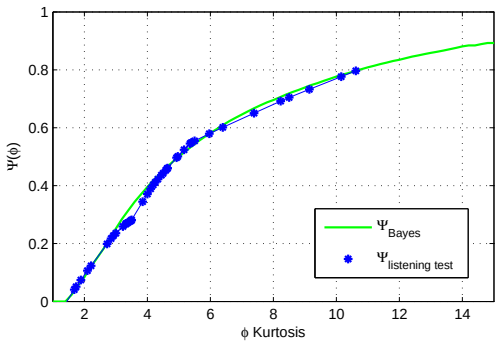


Figure 4: Psychophysical scale for kurtosis.

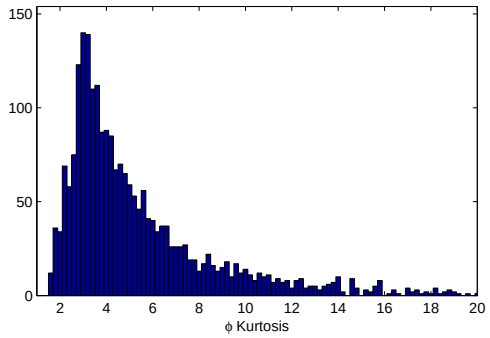


Figure 5: Kurtosis histogram of 20 min. of human speech.

However, though there has been a long-lasting debate about which of the two "laws" might be more appropriate, they are interrelated with the mathematical structure of the underlying psychophysical scale [17]. Considering an explicitly given or postulated psychophysical magnitude function $\Psi(\phi)$ and its derivative $d\Psi(\phi)/d\phi$, one can substitute finite differences for the differential

$$\frac{d\Psi(\phi)}{d\phi} = \frac{\Delta\Psi}{\Delta\phi}. \tag{12}$$

Assuming constant relative JND, $\Delta\Psi = k\Psi$, the psychophysical scale Ψ can theoretically be established utilizing numerical integration of the difference thresholds

$$\Psi(\phi) = exp\left[\int \frac{k * d\phi}{\Delta\phi}\right]. \tag{13}$$

Alternatively considering Fechner's model, the scale function reads $\Psi(\phi) = \int(k * d\phi/\Delta\phi)$. Thus, difference threshold $\Delta\phi$, JND $\Delta\Psi$ and psychophysical scale Ψ are intertwined as functions of the stimulus ϕ.

The difference thresholds of our listening test have not been measured with enough density on the stimulus scale. Therefore, and for intersubjective differences, the calculation of $\Psi(\phi)$ has to be achieved by optimizing a cost function based on a certain assumption of JND model. We chose constant relative JND. Combing five reference values $\phi_j = 1.27, 2.1, 3, 3.5, 4.5$ with the difference thresholds of our ten subjects results in 42 different kurtosis values and therefore in 42 values of the psychophysical scale $\Psi_i = \Psi(\phi_i)$ to be modeled. (In some conditions, thresholds for different subjects yielded identical values.) For each reference value ϕ_j and for each of the associated individual difference thresholds $\phi_{j,m}$, Ekman's law postalutes: $\Psi_{j,m} = (1+k) * \Psi_j$ which enters the optimization problem as a quadratic cost function $J = \sum(\Psi_{j,m} - (1+k)\Psi_j)^2$.

Because of the different $\phi_{j,m}$-values for a single reference kurtosis, the resulting function $\Psi(\phi)$ would exhibit plateau regions and perhaps rather steep transitions between them which conflicts the required smoothness. Therefore, the cost function was extended by a smoothness term based on the distance in stimulus between adjacent scale values,

$$\lambda \sum_{i=1}^{40} \left(\frac{\phi_{i+2} - \phi_{i+1}}{\phi_{i+2} - \phi_i}\Psi_i - \Psi_{i+1} + \frac{\phi_{i+1} - \phi_i}{\phi_{i+2} - \phi_i}\Psi_{i+2}\right)^2,$$

and by a term setting a reference scale value as $\Psi(\phi_{ref}) = \Psi_{ref}$.

Optimization results have been obtained for different smoothness weights λ and a wide range of JND constants $k \approx 0.2 - 0.8$. They revealed minor sensitivity to the actual choice of parameter values. A typical scale function for $\lambda = 50$ and $k = 0.4$ is displayed in Fig. 4 (solid line with asterisks).

To test the psychophysical plausibility of our scale function, we compare it with the predictions of a theoretical model for limited perception proposed by Sun et al. [18]. Their approach relies on the neurophysiologically well-motivated assumption that the acquisition of information in a stimulus is Bayes-optimal at a computational level. Generally, it is argued that the actual mapping function $\Psi(\phi)$ is the outcome of an optimization process taking place during evolution. The optimization minimizes the expected relative error of the quantized representation of the stimulus and therefore depends on its probability density function. Assuming a stimulus pdf $f_{kurt}(\phi)$ and a bounded stimulus range $0 < \phi_0 \leq \phi \leq \phi_1 < \infty$, the psychophysical scale function has to satisfy

$$\frac{d\Psi(\phi)}{d\phi} \propto \phi^{-2/3} f_{kurt}(\phi)^{1/3} \quad for \quad \phi \in [\phi_0, \phi_1] \tag{14}$$

according to the proposed model (cf. Sun et al).

To compare Sun's model with our psychophysical scale function, a probability density function for kurtosis reasonably motivated from an ecologial and evolutionary point of view has to be established. Following Sun et al., we argue that the auditory system may have evolved to optimally process vocalization sounds like human speech. Therefore, we estimated $f_{kurt}(\phi)$ for human speech based on frames of 100ms length in 20 min of different speech recordings (the histogram is shown in Fig. 5) and calculated the Bayes-optimal psychophysical scale function $\Psi_{Bayes}(\phi)$. A direct comparison between the scale model based on the listening test and the Bayes-optimal model is displayed in Fig. 4. The values of our model have been normalized to match the range of the Bayes model. Though the two functions fit surprisingly well which at least indicates the appropriateness of this line of argumentation, it has to be stated that this is neither a direct proof for the validity of Ekman's principle of constant relative JND nor for the Bayesian perception model. In fact, it primarily reveals that the constant relative JND corresponds to the assumption of the relative error criterion in the Bayes model.

COMPUTATIONAL STRATEGIES FOR BREAKBEAT CLASSIFICATION AND RESEQUENCING IN HARDCORE, JUNGLE AND DRUM & BASS

Jason A. Hockman,
DMT Lab,
Birmingham City University
Birmingham, United Kingdom
jason.hockman@bcu.ac.uk

*Matthew E. P. Davies**
Sound and Music Computing Group,
INESC TEC
Porto, Portugal
mdavies@inesctec.pt

ABSTRACT

The dance music genres of hardcore, jungle and drum & bass (HJDB) emerged in the United Kingdom during the early 1990s as a result of affordable consumer sampling technology and the popularity of rave music and culture. A key attribute of these genres is their usage of fast-paced drums known as breakbeats. Automated analysis of breakbeat usage in HJDB would allow for novel digital audio effects and musicological investigation of the genres. An obstacle in this regard is the automated identification of breakbeats used in HJDB music. This paper compares three strategies for breakbeat detection: (1) a generalised frame-based music classification scheme; (2) a specialised system that segments drums from the audio signal and labels them with an SVM classifier; (3) an alternative specialised approach using a deep network classifier. The results of our evaluations demonstrate the superiority of the specialised approaches, and highlight the need for style-specific workflows in the determination of particular musical attributes in idiosyncratic genres. We then leverage the output of the breakbeat classification system to produce an automated breakbeat sequence reconstruction, ultimately recreating the HJDB percussion arrangement.

1. INTRODUCTION

1.1. Background

During the early 1990s, DJ-oriented electronic musicians in the United Kingdom embraced affordable sampling technologies (e.g., Akai S950), allowing them to replicate and manipulate recorded sounds without the need for reverse engineering or synthesis. These technologies, and the innovative techniques developed to harness these technologies resulted in the development of three new genres: hardcore, jungle and drum & bass (HJDB). A unique attribute of these genres is their integration of short segments of percussion solos from funk and jazz recordings known as breakbeats.

While melody and harmony are often-used attributes in the determination of an artist's ability within many genres, perhaps the most revealing characteristic by which an HJDB producer, track (herein used to describe a complete musical piece) or subgenre might be individuated, is through breakbeat selection and creative usage of breakbeats. Breakbeat selection is the choice of one or more breakbeats taken from the vast amount of existing funk and jazz recordings. Breakbeats are typically recorded into a sampler's memory, and an arrangement is created by reordering MIDI notes associated with segments of the breakbeat within a sequencer—also termed *resequencing*. Breakbeat usage in HJDB centres on the creative transformation of an initial source breakbeat through any number of techniques available to HJDB producers. These transformations may include pitch-shifting, time-stretching, filtering, resampling, and resequencing. If the undertaken process is viewed as a pipeline with the original source breakbeat as the audio input and the transformed breakbeat as the output, we can consider a breakbeat transformation to be a kind of complex audio effect. HJDB producers (e.g., Bay B Kane, Justice, DJ Krust) are well known for their choice of source breakbeat material and recognisable for the types of transformations they employ. More recently, artists such as Fracture and Om Unit have led a resurgence in the popularity of using breakbeats, and continue to redefine the aesthetic of breakbeat manipulation. Hardcore, jungle and drum & bass exist as genres within a continuum of musical influence that involves the cultural and generational borrowing of stylistic cues and sonic artefacts (i.e., breakbeat samples) [1], similar to North America's hip hop, which also relies on the history and sound palette of funk and jazz.

1.2. Motivation

Automated analysis of breakbeat usage is of interest both from a musicological perspective—to gain insight on a technology-driven composition process—as well as in terms of how to recreate the stylistic idiom of HJDB producers. Analysis of rhythmic modification of breakbeats can provide insight into the resequencing practices undertaken by HJDB producers in individual tracks, which could, in turn, be used to assess these practices across an artist's career or across subgenres. Automating this procedure would be useful to music producers in search of novel drum sequences for use in their productions, or as an educative recommendation tool for musicians to learn how others have structured their own percussion arrangements.

In pursuit of both these goals, we explore techniques for the automatic recognition of breakbeats as a critical first step towards understanding how they have been used and repurposed within HJDB. Identification of source samples has become a popular online activity, and the ability to "spot" samples is considered a mark of a good sample-based musician. Whosampled is an online community of musicians and avid listeners that collectively attempts to document information related to sample usage.[1] The site allows users to explore how sampled music has been appropriated

* MD is financed by National Funds through the FCT - Fundação para a Ciência e a Tecnologia within post-doctoral grant SFRH/BPD/88722/2012.

[1] www.whosampled.com

by other musicians and to provide links between new and old material.

At present there are only a handful of tools that have been presented that seek to automate the sample identification process. Balen et al. [2] adapted the audio fingerprinting method of Wang [3] to the task of sample identification. Here the *spectral peak* or *landmark* method is used to identify a series of salient peaks in the spectrogram that are similar to those in other recordings. While the spectral peak method has been shown to be robust to noise and distortion, it is less useful for the detection of percussion, or in the context of heavily transposed, or *pitched* material [2], which is often the case with breakbeats. Alternatively, Dittmar presents a sample plagiarism tool designed to assess similarity between musical excerpts by evaluating the correlation between the time-varying gains representing basis activations as generated through non-negative matrix factorisation [4]. Whitney adapted Dittmar's model to assess similarity using Pearson's correlation [5].

In this paper we investigate two classification scenarios for determining the presence of breakbeats in music that has incorporated them. First, we attempt to identify tracks that use a given breakbeat within a dataset with a limited set of known labels to determine the general validity of the approach under controlled conditions. Second, we attempt a more realistic formalisation as a binary classification problem directed towards the presence or absence of a given breakbeat within a database of HJDB music with a much larger number of breakbeats. Following this classification stage, we then automatically extract rhythmic and metrical information in the form of onset and downbeat locations to facilitate the segmentation of breakbeats and hence the eventual reverse engineering to relate the output (i.e., transformed breakbeat) back to the source input.

We consider the presented techniques to be part of computational research in DJ-oriented electronic music, much of which has been pioneered by Collins [6, 7, 8], who developed the *bbcut* real-time breakbeat segmentation and resequencing tools intended to replicate the idiomatic breakbeat manipulations of the genre.

The remainder of this paper is structured as follows: Section 2 outlines our breakbeat classification method. Section 3 presents our evaluations and datasets. Section 4 presents our proposed method for breakbeat sequence reconstruction and reproduction that utilises the breakbeat classification performed in Section 2. Finally, Section 5 provides conclusions and areas for future work.

2. METHOD

Figure 1 depicts an overview of the components in our breakbeat resequencing analysis. At the core of the system is the proposed breakbeat classification method depicted as black boxes. The white boxes show additional processing stages required for breakbeat rearrangement (Section 4).

HJDB producers select breakbeats for their unique rhythms and timbres [9], which are the result of a percussionist's performance on a drum set captured in a particular room using a specific set of recording devices (e.g., microphone, preamplifier, mixing board). The difference between drum hits of different breakbeats can therefore be relatively small, however for a sample identification system to be useful for the task of finding breakbeats it must be capable of identifying heavily manipulated and relatively short audio queries within large collections [2]. The system should also be able to identify the individual segments of a breakbeat (e.g., an individual bass drum hit), as breakbeats are very often resequenced,

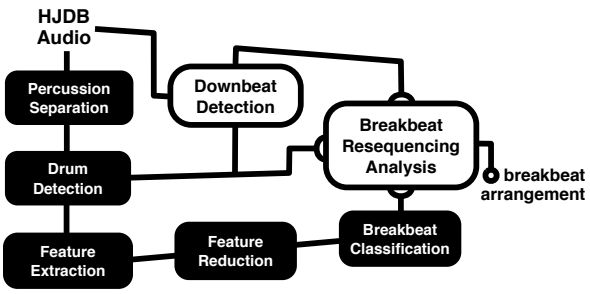


Figure 1: *Overview of proposed breakbeat resequencing analysis method with specialised processing. HJDB audio enters into stages of the breakbeat classification system (black boxes). Additional processing stages (white boxes) are undertaken to achieve the breakbeat rearrangement.*

or may only appear as brief segments—that is, not in their entirety. We therefore propose a specialised solution, which extracts and classifies individual drums from the musical timeline using a combined signal processing and machine learning approach. Our formalisation is motivated by the actual breakbeat usage in HJDB music, in which segments of multiple breakbeats may be used within the same track.

2.1. Multi-breakbeat Classification

The first breakbeat classification method attempts to select the underlying breakbeat from a set of known breakbeat classes in a multi-class classification problem; for example, to determine if the breakbeat in an HJDB track is of *Amen*,[2] *Funky Mule*,[3] or *Apache* [4] origin. We apply this design as a simplified problem to test the validity of the proposed model, however it is also a necessary formalisation for solving ties in the event that more than one breakbeat has been selected in a series of harder binary classification problems (e.g., *Amen* versus *non-Amen*, where *non-Amen* is comprised of examples with any number of breakbeats present).

Within the above context, we first concentrate our efforts on the detection and classification of bass drums as they are the drum class within the standard drum kit that tend to exhibit the least amount of timbral overlap with other sounds. Snare drums in comparison tend to exhibit more overlap with other sounds as they extend over a wider range of frequencies. The presented breakbeat classification method therefore seeks to take advantage of the uniqueness of salient bass drum timbres in each breakbeat. Our proposed method is shown in Figure 1 as the collection of black boxes. To identify regions containing bass drums, HJDB audio (mono .wav files sampled at 44.1 kHz with 16-bit resolution) is entered into a two-stage process of harmonic suppression and drum detection. We incorporate the median-filtering approach of FitzGerald [10] for harmonic suppression (window size = 4096 samples, hop size = 1024 samples) to reduce the contribution of the non-percussive instruments prior to drum detection. To perform drum detection, we create a prototypical spectral profile of a bass drum d as the average spectrum of multiple breakbeat bass

[2] The Winstons – *Amen, Brother* (1969)
[3] Ike Turner & The Kings of Rhythm – *Funky Mule* (1969)
[4] Michael Viner's Incredible Bongo Band – *Apache* (1973)

drums. As shown in Eq. (1) we cross-correlate the harmonically-suppressed spectrogram $S(m,k)$ (with frames m and bins k) with bass drum spectral profile $d(k)$ to produce a bass-drum frequency spectrogram $B(m,k)$.

$$B(m,k) = S(m,k)d(k) \tag{1}$$

Following the standard spectral difference approach for generating an onset detection function (e.g., [11]) we then create a bass-drum detection function by measuring spectral difference in the bass-drum frequency spectrogram, B, similar to the approach presented by Davies et al. [12]:

$$\Gamma_{BD}(m) = \sum_k H(|B(m,k)| - |B(m-1,k)|) \tag{2}$$

where $H(x)$ is the half-wave rectifier operation. Peak-picking Γ_{BD} provides a set of temporal indices used as the initial frames for feature extraction in the bass-frequency spectrogram features (BFS). These features are extracted starting at each detected bass drum position and are six time frames in length (approximately 210 msec) and represent frequencies between approximately 50 Hz to 325 Hz (bins 4–30). In addition to features extracted from the BFS, 13 MFCCs are extracted from the same bass drum regions as found by the bass-drum detection function. In the final step of the feature extraction stage, the means and standard deviations of each row and column of the BFS and MFCC features are extracted (time and frequency independently) from each bass drum event to create feature matrix J (100 features x # bass drum events).

Prior to training a model we first obtain reduced dimensionality feature matrix J' by applying principal component analysis (PCA) to J, reducing the number of features to t principal components, where t represents the smallest value that retains 95% variance. To perform the classification we use the support vector machine (SVM) algorithm, which is trained using feature matrix J', and an associated class vector C that contains the breakbeat class names associated with each bass drum. The SVM is implemented using the LIBSVM C-SVC algorithm with an RBF kernel [13]. The γ and c parameters for the SVM were established ($\gamma = 2^{-3}$, $c = 2^0$) using grid search with three-fold cross-validation using ten tracks in each class. To perform classification of test audio A, feature matrix J_A is created in a similar fashion to feature matrix J. J_A is projected onto the PCA coefficients from the training stage resulting in the reduced set of features J'_A. A classification decision is made for each extracted bass drum in J'_A, resulting in multiple classifications for each test example. An overall class is then determined using majority voting. Ties are resolved using additional classification stages performed with only those classes that are tied.

2.2. Binary Classification

The method presented in 2.1 is capable of determining the presence of an arbitrary breakbeat if provided sufficient training material for each of the breakbeats under assessment. However, collecting a sufficient amount of ground truth for a large number of breakbeats is a difficult task that requires expert listeners. To approach the real problem of identifying an underlying breakbeat within a set of music that contains an unknown, large number of breakbeats, we alter the model in two ways: first, we reduce the number of possible class labels to two, and second, we remove the majority voting component needed to break ties. The latter alteration also affords the more grounded scenario in which multiple breakbeats may be present in one example track.

In addition to classification by SVM, we investigate the use of a deep network classifier in the provision of binary class labels with the aim of increasing classification accuracy through learning additional relationships in the data. Deep network classification involves the transformation of input features into output labels through a series of layered interconnected networks that produce transitional representations. The classifier is constructed in the Theano Python library for deep learning ([14, 15]) and uses same initial set of features J as constructed in 2.1. The network contains a single hidden layer (with 256 nodes) and is trained through minibatch stochastic gradient descent (learning rate = 0.02, epochs = 50000) with batch-wise loss assessed using mean negative log-likelihood.

3. EVALUATION

In order to assess the appropriateness of our selected problem formalisation and the suitability of the methods applied to breakbeat classification, we perform two evaluations based on the multi-class and binary classification systems outlined in Section 2.

3.1. Evaluation 1: Multi-class Breakbeat Classification

The first evaluation is used to determine the validity of a classification based on subtle timbral variations inherent between breakbeats, and to test the worth of the specialised processing stages (i.e., drum segmentation, specialised feature set). In this experiment we consider the simplified problem of a multi-class classification with three breakbeat classes—*Amen*, *Apache*, and *Funky Mule*. These breakbeats were chosen based on their prominence in the HJDB music catalog.[5]

The evaluation is conducted using a dataset comprised of examples of HJDB music, each containing one of the three breakbeats. Annotations for breakbeats were provided by expert listeners (HJDB producers and DJs) through queries in separate threads on the Facebook social media platform, in which HJDB musicians were asked to list their favourite HJDB tracks that use one of the three breakbeats. Musicians listed candidate tracks, and often confirmed or rejected candidate tracks provided by others. In addition, HJDB tracks with the specified breakbeats that were mentioned in interviews by musicians were also added as candidate tracks [9]. All excerpts were originally in .wav or .mp3 format ($\geq$192 kbps) and between 15 seconds and two minutes in length. In total there were 93 excerpts (31 per class).

We test three configurations of the system presented in 2.1. The first configuration is trained using only the MFCC-based features (`M-SVM`); the second configuration is trained using only BFS-based features (`B-SVM`); the third configuration is trained using both BFS-based and MFCC-based features (`BM-SVM`). In addition to these specialised models, we include a fourth, generalised music classification model (`M-GMM`), as it is our hypothesis that a generalised music classification system would not be capable of differentiation between the subtle timbres exhibited between tracks containing different breakbeats. Whereas the `M-SVM` uses an SVM to classify the temporal indices of detected bass drum events, the `M-GMM` uses Gaussian mixture models to learn parameters from MFCCs extracted across frames of audio [16].

[5] e.g., www.whosampled.com/The-Winstons/Amen,-Brother/

	M-GMM	M-SVM	B-SVM	BM-SVM
Amen	74.2%	80.6%	61.3%	**83.9**%
Apache	74.2%	67.7%	77.4%	**90.3**%
Funky Mule	41.9%	54.8%	**93.5**%	87.1%
Avg.	63.4%	67.7%	77.4%	**87.1**%

Table 1: *Accuracies of breakbeat classification systems* (M-GMM*,* M-SVM*,* B-SVM *and* BM-SVM) *for multi-class classification using HJDB examples containing only Amen, Apache, and Funky Mule breakbeats along with cumulative mean accuracies (Avg.). Bold scores denote the best scores in each breakbeat class and average score.*

The four methods are evaluated using leave-one-out cross validation. For the three versions of the specialised system, class membership was determined by majority voting for the number of drum events g extracted (g = 7). The selection of the events was based on the amplitude of the peaks in the bass-drum detection function (see Section 2.1). The M-GMM assigns breakbeat class membership to the set of extracted features with the smallest negative log-likelihood from trained class models.

Table 1 summarises the results of the tested methods (M-GMM, M-SVM, B-SVM, and BM-SVM) using the HJDB breakbeat dataset. Accuracies are provided in percentages for each breakbeat class, and total accuracies are calculated as the mean across classes for each system. The generalised audio classification system, M-GMM, performed reasonably well in classifying tracks from the *Amen* and *Apache* breakbeat categories (74.2% accuracy in each class); however, performance was drastically lower for the *Funky Mule* breakbeat class (41.9%). The M-SVM system achieved slightly better results than the M-GMM system for the *Amen* and *Funky Mule* breakbeat classes, with a slight reduction in performance in the *Apache* breakbeat class.

These results, when considered with the improved *Amen* breakbeat class performance, indicate the potential of breakbeat classification based on individual drum sounds rather than entire tracks.

When compared with the M-GMM and M-SVM systems, the B-SVM showed improved results for the *Apache* and *Funky Mule* breakbeat classes—77.4% and 93.5%, respectively. Of interest was the accuracy for the *Amen* breakbeat class, which was substantially lower than that for the M-GMM and M-SVM systems. In comparison to the *Apache* and *Funky Mule* breakbeats, the *Amen* is a sonically brighter breakbeat, having a greater idiophone presence above the bass drums than the other two breakbeats. A potential reason for the lower accuracy of the *Amen* breakbeat class in the B-SVM system is the lack of spectral modelling for frequencies not represented by the BFS-based features, which focus only on frequencies approximately between 50–325 Hz. Using both BFS- and MFCC-based features, the BM-SVM system achieved the highest accuracies for the *Amen* breakbeat class (83.9%) and *Apache* breakbeat class (90.3%), the second highest for the *Funky Mule* breakbeat class (87.1%), and the highest overall accuracy for the tested systems (87.1%).

3.2. Evaluation 2: Binary Classification

Whereas the first experiment was designed to test the separability of examples containing three different breakbeats, the second evaluation is performed to establish the viability of breakbeat classification as a binary classification problem, as it is unlikely that any dataset would be limited to examples containing a very small number of breakbeats. Furthermore, development of such a database for training purposes—inclusive of all breakbeats and the variety of manipulations they might take—would also be unlikely. The tested methods are evaluated using two classes: one which contains a particular breakbeat, and the other may contain any breakbeat (or drum from another source) other than the breakbeat in question. As this evaluation is based on a more difficult task than the previous three-class test, we hypothesise that the results will reflect this increase in difficulty, even if the effective baseline will be higher.

To perform our evaluation, we expand the first dataset by focusing on one of the breakbeats used in the first evaluation, and increasing the dataset size such that the number of examples of breakbeat x is comparable to that of not breakbeat x. For this example we chose the *Amen* breakbeat, as it the most well-known breakbeat in the literature on HJDB (e.g., [17]). The dataset contains 148 examples of audio files that exclusively use the *Amen* and 132 examples which do not contain (*non-Amen*). The latter class contains a large number of breakbeats and some examples of HJDB tracks (e.g., early Hardcore tracks) that contain samples from drum machines rather than breaks. All additional breakbeat annotations were made by the first author.

Here we test the two systems presented in Section 2.2. The first is the top-performing configuration of the specialised method from the previous evaluation (BM-SVM), and second, the deep network classification method (BM-DN) which uses the same feature set as the BM-SVM, with one alteration: the PCA stage is not performed to allow for non-linear transformations to occur between layers of the network.

It is important to note that in this second evaluation we have removed the majority voting component. Since we provide results based on each drum assessed (rather than per track), the evaluation is performed using three-fold cross validation in which the dataset is split at the track level to ensure that no drums from a given track appear in both training and testing subsets for a given fold.

The results of the tested methods (BM-SVM and BM-DN) using the extended two-class breakbeat dataset are summarised in Table 2; accuracies are provided as percentages for each breakbeat class, and total accuracies are calculated as the mean across classes per system.

	BM-SVM	BM-DN
Amen	78.4%	**81.1**%
$Non-Amen$	77.1%	**86.1**%
$Avg.$	77.8%	**83.6**%

Table 2: *Accuracies of breakbeat classification systems* (BM-SVM *and* BM-DN) *for binary classification of HJDB examples as either Amen or non-Amen, along with cumulative mean accuracies (Avg.). Bold scores denote the best scores in each breakbeat class and average score.*

As expected, we find a slight drop in performance for the winning system from the previous evaluation, BM-SVM. However, the system performs equally well for both *Amen* (78.4%) and non-*Amen* (77.1%) classes, further demonstrating the appropriateness

of the features selected. The second system, `BM-DN`, outperforms the `BM-SVM` in both the *Amen* (81.1%) and the non-*Amen* (86.1%) classes, demonstrating the benefit of deep learning classification in this context.

4. BREAKBEAT RESEQUENCING

Once a breakbeat has been classified for a given track, this information can then be leveraged for a variety of purposes. One such purpose would be for the musicological task of evaluating the sampling trends of producers in the HJDB genres to learn which breakbeats are more commonly used than others and by which artists, and within which subgenres. Another use of this information is to estimate the arrangement of the breakbeat, which is intended to match the HJDB producer's reordering of the original breakbeat segments. A musician might, for example, decide to create a reordering of the segments from two measures of the breakbeat in the creation of a percussion arrangement in an HJDB track. An overview of this procedure is demonstrated in Figure 2.

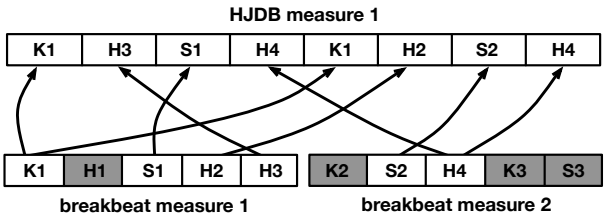


Figure 2: *Example of breakbeat resequencing in practice. The white segments from two separate measures of the breakbeat (bottom) are used to create a single measure of the percussion arrangement in the HJDB track (top). The grey segments are not used.*

To provide an illustrative application of the developed method we can model the percussion arrangement of a target HJDB track using the individual drum segments of the source breakbeat. Unlike the task of drum detection, which requires an explicit declaration of drum class labels, the presented method is able to circumvent this requirement by exploiting the timbral similarity between original breakbeats and the percussion arrangements in tracks that incorporate them. While drum transcription could be used to identify an event as a snare, it is not likely that it could determine that the event was, for example the *second* snare of the *second* measure of the *Amen* breakbeat. We believe that this approach may result in a closer mapping between versions of drums used for a similar purpose (i.e., a well-hit snare drum) due to the knowledge of the timbral character of the percussion. A benefit we see in this approach is that an arbitrary number of drum types could conceivably be included in the pairing, however an obstacle is that the breakbeat being used must be identified. We therefore utilise the breakbeat information extracted in Section 2 to identify the correct audio source (e.g., *Amen* breakbeat), which will serve as the original audio input for the transformation to the target pattern (e.g., PHD & Funky Technicians' *Above and Beyond* (1996)).

The general overview of the process involves the stages of feature extraction, segmentation, and similarity matching to find the best fit segment from the source file that matches that of the target. The feature extraction and segmentation stages are identical for both the source and target files. Features are extracted from each file in a similar method as explained in Section 2.1, with three key differences. First, as we are attempting to match all segments rather than bass drum regions alone, we extract features from entire input audio files. Second, we extend the set of spectral features (i.e., MFCCs and BFS) to include snare drum frequency spectrograms (SFS), which represent frequencies between approximately 430–5000 Hz. In addition to these features, we include the bass-drum detection function Γ_{BD} (Eq. (2)) and a similarly constructed snare drum detection function Γ_{SD} obtained from the SFS.

We then reduce the number of features in the BFS and SFS feature matrices through the application of PCA dimensionality reduction. PCA is applied to reduce the dimensionality of the BFS and SFS matrices to the smallest value that retains 95% variance. The transformation is applied to each feature type (i.e., BFS and SFS) independently to ensure preservation of the drum types exhibited within each feature matrix.

Both source and target signals along associated features are partitioned into individual percussion events by selecting peaks from the summed bass and snare drum detection functions. As mean-segment values will represent each feature dimension, segment boundaries at the beginning and end of each bar are determined through the use of a downbeat detection algorithm [18]. Mean features are extracted from each segment, and features are then normalised across the time axis (i.e., rows) in the source and target representations, separately.

We then evaluate the the similarity of source breakbeat segments to those of the HJDB track through the construction of a cosine similarity matrix M [19]. Selection of a source segment based only on maximum similarity to the target segment resulted in many reconstruction errors. Many of these errors contained large gaps of silence due to an incorrect pairing of short drum hits where a longer more substantial hit was required (e.g., an off-beat *ghost note* in place of a salient snare). We therefore attempted to observe the top r ranking source segments as potential candidates (r = 5). Each segment is weighted by the normalised ratio between the candidate segment length $\rho(r)$ to that of the HJDB segment $\varsigma(n)$ for time segment n as in Eq. (3):

$$w(r) = \frac{\rho(r)/\varsigma(n)}{max(\rho(r)/\varsigma(n))} \tag{3}$$

where w is a series of weights applied to the segments b. The segment exhibiting the maximum interaction after weighting is selected as the winning segment ν for time iteration n:

$$\nu(n) = max(w \cdot b) \tag{4}$$

We replicate the data from the winning source segment $\nu(n)$ in the n^{th} position of the target vector, scaled to match the amplitude of the target segment.

Examples of the presented transformation are made available here.[6] The resequenced breakbeats in the examples sound coherent and similar to the target patterns, albeit without effects such as pitching and distortion, which are generally applied by HJDB producers. We identify three main challenges that will potentially cause errors in the presented method. First and most obvious, errors in breakbeat classification will result in a transformation that will not sound as intended. As the spectral character of drum types differs across breakbeats, incorrect drum type matching may occur. Second, if the onset detection results in spurious or missed notes, then the matching stage will produce additional drum events or

[6] www.music.mcgill.ca/~hockman/projects/breakscience/examples

spaces where drums should occur, respectively. Third, if breakbeats are heavily pitched, one drum type may be matched with another—for example, a bass drum that has been transposed upwards may be matched with snare. Errors due to these factors could be remedied through a semi-automatic method, that would allow users access to the parameters of the transformation. An initial estimation of the percussion arrangement could then be improved through modification of such parameters.

5. CONCLUSIONS

In this paper we present a computational approach for the analysis of breakbeats used in hardcore, jungle and drum & bass recordings. Our chosen approach to this problem is that of music classification, with a specialised procedure of individualised drum classification. To evaluate the plausibility of this solution we first attempted a simplified multi-class problem to determine if our problem formalisation was appropriate. Results of an evaluation with three breakbeat classes demonstrated the effectiveness of specialised processing, which is used to isolate bass drums through suppression of harmonic content and segmentation. We then attempted the more realistic formalisation of a binary classification problem, in which the two classes are tracks that contain a specific breakbeat (*Amen*) and tracks that contain breakbeats other than the specified breakbeat, or in some examples, no breakbeat at all (not-*Amen*). This formalisation is more practical than the multi-class classification approach, as the latter requires ground truth for each breakbeat under analysis. If a HJDB producer wishes to recreate the idiomatic style of a particular breakbeat, then the binary classifier requires ground truth for a single breakbeat. To test the efficacy of this formalisation, we conducted an evaluation with the top-performing model from the multi-class problem (`BM-SVM`) and a deep network classifier `BM-DN`, in which the `BM-DN` outperformed the `BM-SVM`.

For future work, we intend to look into the incorporation of rhythmic features to aid in both the breakbeat classification system, as well as in the breakbeat resequencing analysis. In addition, we will investigate the prospect of deep architecture feature-learning for our classification and transformation.

6. ACKNOWLEDGMENTS

The authors would like to thank the many HJDB producers and DJs who contributed annotations for the datasets used in this research. We would also like to extend our thanks to Alexander Foy, Conor O'Dwyer (Code), Daniel Lajoie (ESB), and Kian Joyner (Godfather Sage/Kian) for providing music from their collections.

7. REFERENCES

[1] S. Reynolds, *Energy Flash: A Journey through Dance Music and Rave Culture*, Soft Skull Press, Berkeley, CA, 2012.

[2] J. Van Balen, "Automatic identification of samples in hip hop music," in *Proceedings of the International Symposium on Computer Music Modelling and Retrieval*, 2012, pp. 544–551.

[3] A. Wang, "An industrial strength audio search algorithm," in *Proceedings of the 4th International Conference on Music Information Retrieval*, 2003, pp. 7–13.

[4] C. Dittmar, K. Hildebrand, D. Gaertner, M. Winges, F. Müller, and P. Aichroth, "Audio forensics meets music information retrieval—a toolbox for inspection of music plagarism," in *Proceedings of the 20th European Signal Processing Conference*, 2012, pp. 1249–1253.

[5] J. L. Whitney, "Automatic recognition of samples in hip-hop music through non-negative matrix factorization," M.S. thesis, University of Miami, 2013.

[6] N. Collins, "Algorithmic composition methods for breakbeat science," in *Proceedings of the International Conference: Music Without Walls? Without Instruments?*, 2001, pp. 21–23.

[7] N. Collins, *Towards autonomous agents for live computer music: Realtime machine listening and interactive music systems*, Ph.D. thesis, University of Cambridge, 2006.

[8] N. Collins, "Influence in early electronic dance music: An audio content analysis investigation," in *Proceedings of the 13th International Society for Music Information Retrieval Conference*, 2012, pp. 1–6.

[9] J. A. Hockman, *An ethnographic and technological study of breakbeats in hardcore, jungle and drum & bass*, Ph.D. thesis, McGill University, 2014.

[10] D. Fitzgerald, "Harmonic/percussive separation using median filtering," in *Proceedings of the 13th International Conference on Digital Audio Effects*, 2010, pp. 203–206.

[11] S. Dixon, "Onset detection revisited," in *Proceedings of the 9th International Conference on Digital Audio Effects*, 2006, pp. 133–137.

[12] M. E. P. Davies, P. Hamel, K. Yoshii, and M. Goto, "Automashupper: Automatic creation of multi-song music mashups," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 1726–1737, 2014.

[13] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2011.

[14] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: A CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference*, 2010.

[15] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio, "Theano: New features and speed improvements," in *Proceedings of the NIPS 2012 Deep Learning Workshop*, 2012.

[16] J.-J. Aucouturier and F. Pachet, "The influence of polyphony on the dynamical modeling of musical timbre," *Pattern Recognition Letters*, vol. 28, no. 5, pp. 654–661, 2007.

[17] The Economist, "Seven seconds of fire," *The Economist*, pp. 145–146, December 2011.

[18] J. A. Hockman, M. E. P. Davies, and I. Fujinaga, "One in the jungle: Downbeat detection in hardcore, jungle, and drum & bass," in *Proceedings of the 13th International Society of Music Information Retrieval Conference*, 2012, pp. 169–174.

[19] M. Cooper and J. Foote, "Automatic music summarization via similarity analysis," in *Proceedings of the 3rd International Symposium on Music Information Retrieval*, 2002, pp. 81–85.

SPATIALIZED AUDIO IN A VISION REHABILITATION GAME FOR TRAINING ORIENTATION AND MOBILITY SKILLS

Sofia Cavaco, Diogo Simões and Tiago Silva

NOVA LINCS, Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
2829-516 Caparica, Portugal

scavaco@fct.unl.pt

ABSTRACT

Serious games can be used for training orientation and mobility skills of visually impaired children and youngsters. Here we present a serious game for training sound localization skills and concepts usually covered at orientation and mobility classes, such as front/back and left/right. In addition, the game helps the players to train simple body rotation mobility skills. The game was designed for touch screen mobile devices and has an audio virtual environment created with 3D spatialized audio obtained with head-related transfer functions. The results from a usability test with blind students show that the game can have a positive impact on the players' skills, namely on their motor coordination and localization skills, as well as on their self-confidence.

1. INTRODUCTION

Due to the entertaining factor, well designed (serious) computer games can be a very useful tool to teach school curriculum or skills. The past few years have witnessed an increasing interest for serious games and it has been shown that serious (educational) games can be used effectively to study school curriculum [1]. While playing, students feel more motivated and engaged in the studying and learning process. With games, school material can be presented in a fun way to help the students in the learning and studying process. In addition, if the games include a reward system that is a function of the learned school material, students can actually end up putting a bigger effort on learning the material in order to receive the reward. Additionally, games can be used for training skills like problem resolution strategies, data interpretation, problem analysis, ability to create mental representations of abstract concepts, among others [2, 3].

Though most games have a strong visual component and have been designed for sighted users, visually impaired children can also benefit from playing such games to study the school material or for vision rehabilitation purposes, such as orientation and mobility training. Orientation and mobility can be defined as the ability to move independently, safely and efficiently from one place to another. This includes the ability to cross streets independently, to use public transportation, to go to work, school, or other places independently. In order to achieve independent mobility in adult life, blind and low vision students need to develop some specific skills that are taught and trained at orientation and mobility classes.

In order to understand if serious games can help blind children and teenagers with their mobility skills, we have developed the Audio Space Station. This is a serious game that was designed for visually impaired children and youngsters and that aims at training orientation and simple mobility skills (figure 1). The game has three challenges that focus on training different skills. The goal of the first challenge is to train sound localization in the azimuth, while the second challenge also includes elevation. In the third challenge distance to the sound sources is also included. In order to maintain the child motivated in playing, the difficulty of the challenges adapts to the player's performance and the game includes a reward system. The results from a usability test with blind students, show that apart from training sound localization (orientation) skills, the game can also be used for training body rotation skills and to help blind youngsters to be more confident on moving without assistance (from parents, teacher, friends, etc.).

Figure 1: A blind youngster playing Audio Space Station.

The game uses 3D spatialized audio to create a virtual environment that the player can enjoy while training these skills. The 3D sound is implemented with head-related transfer functions (HRTF). The usability test showed that the players can correctly localize the sounds created in this manner.

In section 2 we discuss educational and vision rehabilitation games designed for visually impaired children. Section 3 describes the game and its three challenges. It also describes the technical details and how 3D spatialized sound is obtained. The tests are discussed in section 4 and the conclusions in section 5.

2. RELATED WORK

Even though serious games can be a very effective tool for learning and training skills, including orientation skills, most educational games are not accessible to blind children. In order to be accessible to visually impaired children, games must use non-visual modalities, such as audio. In this section we discuss a few serious games developed for developing cognitive or mobility skills of visually impaired children, that illustrate how audio can be used to substitute the images and convey information to the users.

Sánchez and colleagues have developed a few educational games for blind children that use audio for accessibility purposes. These include the AudioDoom, Audiobattleship and AudioMUD [4, 5, 6]. AudioDoom includes a highly interactive acoustic environment obtained with 3D spatialized audio. The aim of the game is to test the hypothesis that this type of environment can be used to stimulate and reinforce some abilities of blind children, such as spatial representation. Audiobattleship is a version of the classic battleship game that was adapted to blind children. The game uses spatialized sound to substitute the visual cues of the classic version of the game. This game was developed to stimulate the cognitive development of blind children using interaction with machine as well as with other children. AudioMUD is a 3D virtual environment that instead of using spatialized sound, it uses speech to give the user all information about the environment and navigation instructions.

There are also educational games designed for specific school subjects. The *Código Pitágoras* is a math game that aims at motivating blind students to learn and enjoy mathematics [7]. All the features in the game are complemented with audio and the game uses 2D audio to guide blind players while traveling in the game's maps. The game can be played individually by blind students but sighted students can also play. Therefore, while it was designed for blind students, this game also has graphics so as to be more interesting to sighted students. This characteristic can be an incentive for the collaboration of blind and sighted students and serve as a means to help blind children to integrate better in their inclusive classrooms and society.

AudioPuzzle and Terraformers are games accessible to blind children that, while not educational, are worth mentioning because they illustrate many different forms of using audio effectively to give information to blind users [8, 9]. AudioPuzzle consists of a musical puzzle in which the players use the Android's haptic screen to sort music pieces. Terraformers uses speech and 3D spatialized audio in several features, such as in verbal hierarchical menus, and to simulate an acoustic compass and a sonar.

As for orientation and mobility games, also only a few games have been proposed in the literature. *Blindfarm* is a game that was designed to help visually impaired children to memorize paths they use in their daily life routines [10]. The game uses both the GPS and the compass sensor and includes a feature that adults can use to mark a path in the real world by placing virtual animals in specific locations. The goal of the players is to follow the real path by listening to the vocalizations of the virtual animals in stereo sound. These sounds signal which direction the player should follow.

Other serious games and tools for orientation and mobility have been developed by Sánchez and colleagues. These include an outdoor navigation tool that uses the GPS of the Pocket PC to estimate the user's position and its sound to communicate with the user [11], and an audio-haptic game in which the players have to navigate in a virtual space while collecting objects [12]. The goal of this game is to analyze the usability of an audio-haptic game and its influence in orientation and mobility skills. The limitation of this game is that even though the virtual world is three-dimensional, the sound is not.

These orientation and mobility games lack the immersion needed to train certain orientation and mobility skills. None of them uses three-dimensional sound, which is of great importance given the special condition of the target users. On the other hand the proposed game, Audio Space Station, uses 3D spatialized sound to train audio localization and body rotation skills.

Another game developed for blind users that uses 3D spatialized sound and that can be used for orientation and mobility training is Demor [13]. This game requires the players to localize the target sounds, and while the game was primarily designed for entertainment, it can be used to train localization skills. Nonetheless, we feel that the theme is not appropriate for children as it is a shooting game. On the contrary, the Audio Space Station game has a theme that is appropriate for all ages, and we were careful not to use any kind of violence in the game. As it will be seen below, users have to localize target sounds (sounds of alien insects or a robot) and catch, photograph or follow them (there is no firearm shooting in the game).

3. THE 3D AUDIO SPACE STATION GAME

The Audio Space Station was designed to train orientation and simple mobility skills of visually impaired children and youngsters. More specifically, the game aims at helping players to perform accurate sound localization and training other orientation concepts that are taught and trained at orientation and mobility classes, such as the use of landmarks and sound cues for navigation. In addition, the game can also be used to train simple body rotation skills and to help the players to surpass their fear of moving in unknown environments.

The game's theme, which is science in a space shift, was chosen with the purpose of having a theme adequate for a wide age range. The game is about a scientist working in a space station, who has to capture or photograph alive alien insects for posteriori analysis, or follow a robot in a laboratory while avoiding some (sonified) obstacles that lay in the room. The game includes three challenges in which the player uses sound localization skills and simple body rotation motion to control the game's main character. The challenges can be chosen in any order and can be played more than once.

The game uses 3D spatialized audio to create a virtual environment that the player can enjoy while training these skills. The 3D sound is implemented with head-related transfer functions (HRTF).

3.1. Technical details

Since one of the goals of this game is to train simple body rotation movements, we needed a mobile platform that can easily be used while standing and moving. We also needed to be able to estimate the orientation of the player and detect his/her movements, therefore we needed a platform with sensors that allow us to estimate this information. In addition, we wanted the game to be accessible to the general public and in particular to visually impaired users. After consulting a blind person who regularly uses modern technology, we realized that tablets and iPads can be harder to use by visually impaired people due to their large screen size and the

used layouts. For those reasons, we decided to develop the game for Android smartphones.

We used the free version of Unity3D[1] for developing the game. While this is an audio game, which should be played without access to the graphics, we opted to maintain a simple visual interface, to allow the teachers or parents to monitor the player's progress.

The only specific requirements of the game are that the smartphone must have a gyroscope, and users should wear a set of headphones. To correctly play the game, the players should always position the smartphone facing their faces (figure 1). Also, the users should stand while playing the game because they will need to freely rotate over themselves.

The gyroscope is used to input information about the player's rotation and spatial orientation into the game. This information is used to control the main character, whose orientation depends on the player's orientation. More specifically, the player controls the main character by rotating and moving the smartphone.

The headphones are required to give three-dimensional audio feedback to the player, which can only be achieved with the reproduction of sounds through two channels, directly to the player's hears. We used OpenAL[2] to produce 3D sounds with HRTFs.

3.2. 3D Spatialized Audio

The game uses verbal and non-verbal audio. The verbal audio is used to give information to the player, such as instructions and menu options. Non-speech audio is used to indicate the existence and position of objects, insects, etc. All non-speech sounds consist of 3D spatialized sounds.

In order to produce spatialized audio we can change the right and left channel signals to simulate what happens in the real world. The signals that reach our right and left ears are not exactly the same and the brain uses their differences to determine the location of the sound source. These differences can be for instance temporal or in intensity (interaural differences). Yet, when using only interaural differences, our brain cannot unambiguously determine the exact direction of a sound source (unless we move our head and hear the sound again).

There are other cues that the brain uses. In particular, sound is modified by the head, torso and pinnae, and our brain uses this direction-dependent acoustic filtering of the sound waves to unambiguously determine the sound's direction. HRTFs can be used to reproduce this direction-dependent acoustic filtering. By changing the left and right channel signal with HRTFs, we obtain a pair of signals that when heard simultaneously (at the left and right ear) produce the perception of 3D spatialized sound.

The Audio Space Station game uses HRTFs to produce 3D spatialized audio. In a preliminary test with visually impaired students from an inclusive school, we compared the sounds obtained with Unity3D's audio engine with sounds produced with HRTFs. In this test we used two sounds: one obtained with Unity3D's audio engine and the other with OpenAL's HRTFs. Then we asked the subjects to localize the sounds. The goal of this simple test was to determine if Unity3D's sounds are good enough for 3D localization or if we required a more complex technique, like using HRTFs, to process the sounds.

The results clearly demonstrated that the techniques used by Unity3D's sound engine are not capable of conveying proper localization cues, which is due to the fact that Unity3D simply uses sound intensity panning. Unlike HRTFs, sound panning does not allow our brain to distinguish sounds coming from opposite sides (front versus back, etc.), which leads to confusion and very poor performance on a game such as this.

[1] http://unity3d.com/

[2] http://openal.org/

3.3. The Orientation Audio Challenges

Being a virtual reality audio game, the player controls the main character in a first-person perspective, similarly to what happens in Terraformers or any first-person shooter [9, 14]. The players listen and move just like the character would, which contributes to the game's immersive and engaging qualities. Since the game aims at training body rotation movements, the players do not have to walk but they need to rotate over themselves holding the device with the screen facing their faces or chest. This way, they are able to move in the game's virtual environment.

Since the game is designed for visually impaired users, it consists of an audio game and it is supposed be played without seeing the graphics. All information is output as audio (speech and 3D spatialized audio) and vibrations of the device. The players can interact with the game and control the main character by touching the screen and moving the device. In order to keep track of the users orientation, the game uses information from the device's gyroscope.

In each of the three challenges, the players have to heavily use their hearing. The first challenge, called *Cockroach Hunt*, takes place in one of the spaceship's rooms. Here, the players have to capture some alien roaches that escaped from the science laboratory. In this challenge, the main character is standing in the middle of the room and hears the alien insects around him/her. The sounds of the roaches are static 3D spatialized sounds, that is, the insects are standing somewhere in the lab. Also the roaches appear in turn in the virtual environment, that is, at a time the players hear only one roach.

The players must locate the roaches, that is, they must estimate their 3D direction, and then they must capture them. To achieve this, the players should turn around themselves (holding the smartphone in front of them) until they hear the roach right in front of them. Once they are facing a roach, they can capture it by touching the device's screen. This action is followed by a sound representing a capturing gadget in action, and a sound suggesting that the insect was caught, in case of success. If the player takes too long to capture the roach, the insect escapes. To indicate this, the device vibrates and the players hear the roach running out of the room. Since this is an audio game, we need all these sounds (like the sound from the capturing gadget, the sound of the roach being captured, etc.) to give feedback to the player about what is happening in the game.

The second challenge, which we called *Space Bees*, is very similar to the first challenge. Again, the player has to locate and turn towards alien insects, which here are alien bees, but in this challenge the creatures are not static. Instead, they are flying around the character, while describing a sinusoidal motion (flying up and down). In this challenge the players have to photograph the bees as fast as they can, for documentation purposes. Once the insects are photographed they disappear. Also, after a short period the insects that were not photographed disappear. Again the main character is standing in the middle of the room and there is only one insect present in the lab at each moment. Like before, the players have to use their hearing sense to localize the sound sources (the bees) and rotate over themselves to face the insects in order to photograph

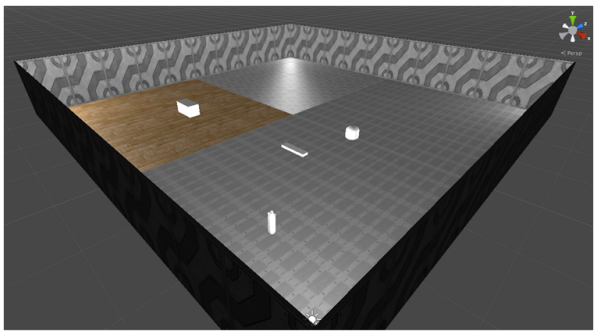


Figure 2: A scene from the third challenge.

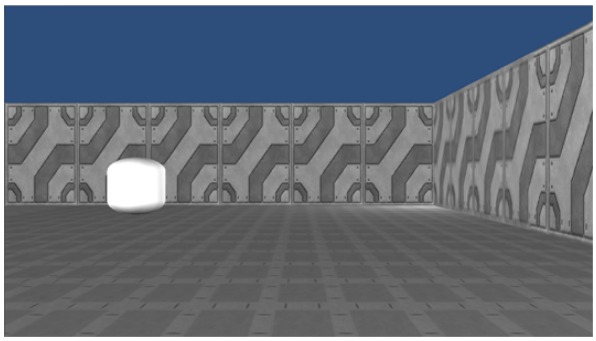


Figure 3: A scene from the third challenge in the user's perspective.

them (by touching the device's screen).

In the third challenge, the *Sound Path* challenge, the sounds have still another dimension: distance. In this challenge, the player navigates in a virtual room to follow a robot. When the challenge starts, the player will hear the sound of the robot, which is standing in a random initial position in the room. The player must navigate towards the robot. Once the player gets there, the robot will navigate to another position. The process is then repeated: the player hears the robot's sound and has to follow it again. the challenge ends when the player reaches the robot three times. (While this number is fixed, it can easily be configured.)

In order to navigate and reach the robot in the virtual room, the main character can walk. As before, the player can rotate the device to make the main character turn around, but here the player can also make the main character walk by touching the screen (the player does not need to walk in the real world).

In order to increase the difficulty of the challenge, there may also be some obstacles in the room that the main character must avoid. All obstacles produce sound, so that it is possible to locate and identify them in the audio virtual environment. More specifically, in this challenge the player can choose to enter one of two labs: In the first lab there are no obstacles and the player only needs to follow the robot. In contrast, the second lab has some obstacles: a dog, which is barking and therefore can be identified and localized, and a bee from the Space Bees challenge. The player must localize them and make the main character avoid them while he/she is following the robot. While these obstacles do not move, their positions are random.

As an example, figure 2 shows a scene from the third challenge. (Note that the graphics are not used to play the game, these can be used by the teachers or parents just to help the child if needed.) Here the player (the thin cylinder) has to follow a robot (the large cylinder) while avoiding obstacles in the room (the two parallelepipeds). Figure 3 shows the user's perspective for the same scene. In this scene the user will hear the sound of the robot, which is represented by the fat cylinder in the figure, coming from the front-left.

To give players feedback on when the main character is walking, the sound of footsteps is reproduced whenever the main character is in walking mode. Note that there are three types of floor in the Sound Path challenge room (figure 2). The sound of the footsteps varies according to the flooring so that the player can better identify in which region of the room he/she is.

3.4. The reward system

The reward system was implemented to motivate the players to play often. The targeted orientation and mobility skills can improve better with repetitive training. Therefore we wanted the players to feel motivated to play frequently. In order to increase their motivation to play, we added a reward system to the game. This reward system consists of a score that increases when the player manages to capture or photograph insects (in the Cockroach Hunt and Space Bees challenges, respectively).

Before the proposed version of this game was ready, we run a preliminary test with blind and low vision students in an inclusive school to ascertain if blind and low vision students enjoyed the game and if it had any compromising faults [16]. In order to catch/photograph as many insects as possible, one of the students who participated in that study adopted a technique that consisted of touching the screen repeatedly and quickly (even before he could hear the insect) while moving the phone around. The student was so eager to catch/photograph the insects quickly that he did not pay the necessary attention to the sounds.

That type of behavior is not desirable (it would be preferable that the student would have payed more attention to the sounds to make more correct localization estimations, even when that means taking a bit longer to do it). In response to this behavior, we adapted the scoring system and added the adaptive difficulty mechanism described in section 3.5, which do not favor the type of behavior shown by this subject. In more detail, the score can decrease when the player is not able to capture/photograph an insect; if the player touches the screen to catch/photograph an insect and misses the target, the score will decrease.

3.5. Adaptive difficulty

The two first challenges, that is, the Cockroach Hunt and the Space Bees, have background noises to increase their difficulty. The intensity of these noises depends on the difficulty level of the challenges: the higher the difficulty, the louder these noises become. Another parameter that depends on the difficulty level of these two challenges is the time the scientist has to capture each insect, that is, the time the insects remain in the room before they disappear: the higher the difficulty level, the less time the insects remain in the room. Finally, as the difficulty increases, the intensity of the sounds decreases, as if the insects were further away from the player, which makes the localization of the insects a little harder. All these three parameters (background sound level, the time to

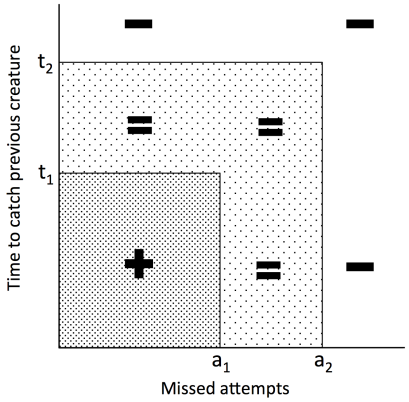


Figure 4: Adaptive difficulty rules for the training mode.

catch the insects and sound intensity) have minimum and maximum values, so as to prevent that the game becomes impossible to play.

The two first challenges have two distinct game modes. One for competition and one for training. In the competition mode, the challenges contain a fixed number of insects, n_c and n_b, for the Cockroach Hunt and Space Bees respectively. The difficulty level of the challenges increases as a linear function on the number of creatures that have appeared so far. In this game mode the player never really loses, but the final score reflects the player's degree of success.

As opposed to the competition mode, the training mode does not have a maximum number of creatures. The number of insects that appear in the room depends on the player's performance. Basically, the player is allowed to let escape up to three creatures. When the third one escapes the challenge stops running.

In this mode the player's performance also defines the difficulty level. This mode uses adaptive difficulty, that is, the difficulty increases or decreases depending on the player's performance. If the player has a very good performance, the difficulty increases. On the other hand, if the player is struggling to succeed, the difficulty will decrease. By adapting the difficulty to the player's performance, the game allows the player to learn and improve at his/her own pace and according to his/her abilities.

The adaptive difficulty function is illustrated in figure 4 and follows the same difficulty-adaptation scheme suggested in [15]. The figure shows that the difficulty increases when the player's performance falls in the left lower square, that is when the child takes less than t_1 seconds to catch or photograph the insects, and misses less than a_1 insects (in other words, the child shows good performance, she is fast and precise on localizing the sounds). The difficulty does not change (region with $=$ in the figure) when (a) the player is fast but not very precise (that is, the time, t, to localize the sound is less than t_1 seconds but the number of missed insects, a, is between a_1 and a_2), (b) when the player is precise in localizing the sounds but takes a bit longer to do it (that is, $a \leq a_1$ and $t_1 < t \leq t_2$ seconds), or (c) when the performance is not great but also not too weak ($t_1 < t \leq t_2$ and $a_1 < a \leq a_2$). Finally, the difficulty decreases when the player takes to long to localize the sounds ($t_2 < t$) or misses many of them ($a_2 < a$).

The parameters t_1, t_2, a_1 and a_2 can be easily reconfigured and have different values for the two challenges:
$t_1 = 3, t_2 = 6, a_1 = 2$ and $a_2 = 4$, for the Cockroach Hunt challenge, and
$t_1 = 4, t_2 = 7, a_1 = 3$ and $a_2 = 5$, for the Space Bees challenge.

3.6. Game menus

One of our major concerns when designing this game was to make it fully accessible to visually impaired children. It is important that blind players can use the game independently and without great effort or frustrating moments. For this end, the game uses sound, text-to-speech technologies and menus that allow players to navigate between game modes and levels according to their own preference. The user interaction techniques and the mechanics of the game menus were carefully designed so that these are suited to blind users.

First of all, as the players are visually limited, the game menu entries are spoken. The menu entries were recorded and the sound was modified to fit the game theme. As the players navigates between options, the corresponding sound of the menu option is played. The navigation (switching menu options) is achieved by simply sliding a finger on the device's screen, either to the left or to the right. The menu can be mentally visualized as a cylinder with the multiple options written on its surface. The sliding action would make the cylinder rotate on its vertical axis.

4. TESTS WITH BLIND AND LOW VISION USERS

We have run a usability test with visually impaired students in an inclusive school[3]. The goal of the test was to determine how the students reacted to the game and to observe the impact it had on their localization and mobility skills. Only the first and second challenges were tested.

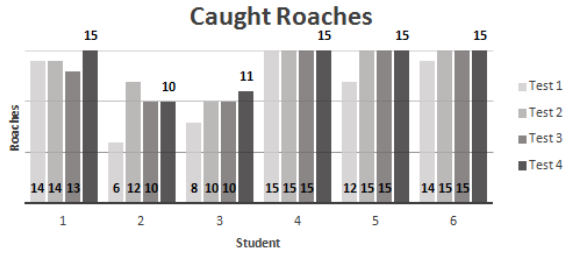
Six blind and low vision students (one girl and five boys with ages between 11 and 14 years old) participated in the test. All students had severe visual impairments: four were blind and two had low vision. The two students with low vision (students number 1 and 4) played the game without seeing any graphics. Other identified problems of the students were: Bardet-Biedl syndrome (student number 1), cerebral palsy (student number 2) and hyperactivity (student number 6). Four of the students had independent mobility (that is, they walked independently), while two of them were dependent on the help of a friend or teacher (students number 2 and 3), that is, they held someone's arm while walking.

4.1. Protocol

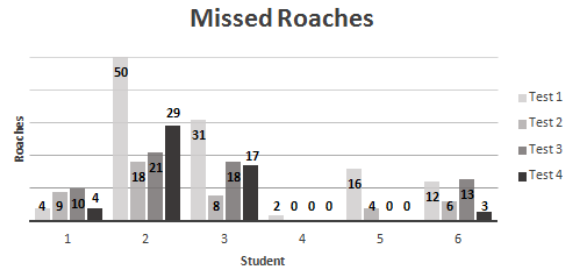
During this test, the students played the Cockroach Hunt and Space Bees challenges in a mobile phone and wearing headphones (figure 1). The test was done indoors, in a spacious school office where the students had space to move around without constantly bumping into objects (this indoor space is shown in figure 1). Since the tests were performed during the school breaks, there was background noise from children playing in the school playground that could be heard in the office.

In order to understand if the results improved, each student played the game on four different days during two or three weeks. On each day, they played each challenge once in the competition mode (there were 15 insects for each trial). There was an exception of one student who played the game four times on three days

[3]Inclusive schools are schools where special needs children are integrated in the same classrooms as regular children.

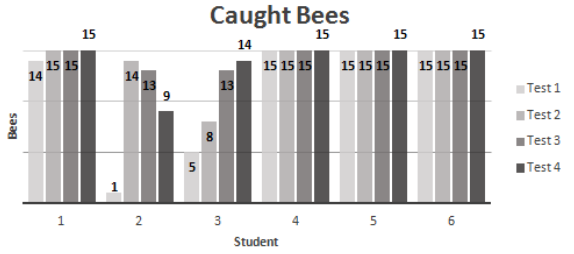


(a)

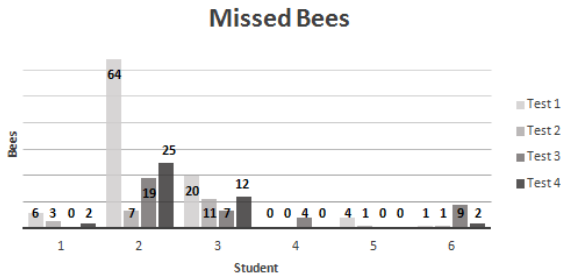


(b)

Figure 5: CockRoach Hunt challenge. Results from the four trials of all students: (a) total number of caught static roaches and (b) total number of missed attempts.



(a)



(b)

Figure 6: Space Bees challenge. Results from the four trials of all students: (a) total number of photographed flying bees and (c) total number of missed attempts.

because he missed school several days during the trials. On the last day of the test, this student played the game once in the morning school break and another time at lunch break.

Before starting the test on the first day, we explained the students how to interact with the game. They were also told that they had to hold the phone in front of them and that to capture the insects they had to turn themselves along with the phone. As suggested by a blind adult, in order to avoid having unexpected reactions to the sounds, the students also heard all the sounds before starting the actual test.

In order to overcome any initial difficulties and to verify that the students held the phone properly and could interact with the game in the expected manner, there was a training period before the first two trials (two days). The training consisted on playing the challenges until the students were able to capture five insects. After that, most of the students adapted quite quickly to the game and needed little or no further instructions. There was an exception of a student who needed extra help and further training: this was the blind student with cerebral palsy.

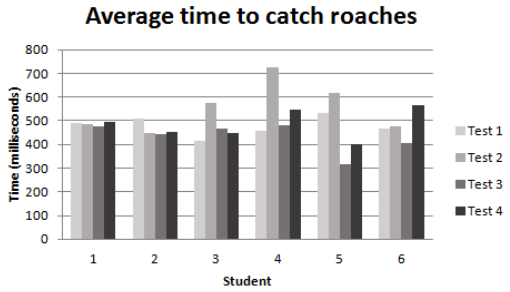
After the last trial, the students answered a small set of questions about their personal opinion on the game, difficulties felt, etc. While the test was run by the authors, there was a special education teacher present in the office during all the trials. After the test, she also gave us very valuable feedback.

4.2. Results

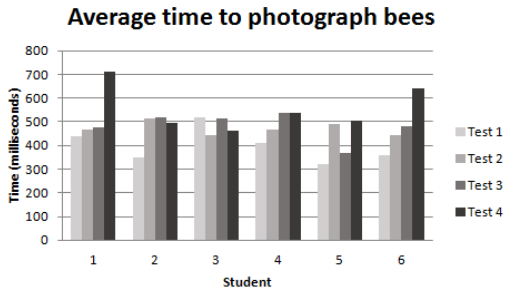
We observed that there was a learning pattern in the two challenges. Some students showed very good results, that is, a high number of caught insects, on all four trials, but others showed a clear improvement from the first trial to the latter trials. Figures 5.a and 6.a show the number of caught/photographed insects for each student in all four trials. Note that since there are 15 insects in each trial (and for each challenge), if the student is able to catch x insects, it means that $15 - x$ insects are able to escape, which happens when the student takes too long to catch the insects. As it can be observed, the students who did not catch many insects on the first day showed improvements on the remaining three trials. More precisely, students 2 and 3 can only catch 6 and 8 roaches in the first trial, but those numbers increase to 10 and 11 in the last trial. The same observation can be made for the bees challenge: students 2 and 3 had the poorest results on the first trial but were able to improve in subsequent trials.

Interestingly, students 2 and 3 are the two students with dependent mobility, which suggests that since they were used to depend on a friend to move, either at the beginning of the test they were not so confident on turning around by themselves to catch the insects or they were also used to rely on a friend or adult to do the localization for them. Either way, the results show that the game can help the children with dependent mobility to gain more confidence on moving by themselves without the need to hold a friend's arm.

Another observation is that the students improved their localization estimation and/or their confidence on their estimations. Figures 5.b and 6.b show the missing attempts to catch/photograph the insects. When an insect appears, the players can make several attempts to catch it before they actually manage to catch it or before the insect disappears. As it can be observed, in average, the number of missed attempts decreased from the first day to subsequent days (especially in the roaches challenge). This shows that the localization estimations improved.



(a)



(b)

Figure 7: Average time spent to catch or photograph the insects. Each bar indicates the average time spent by one student (a) to catch all roaches or (b) to photograph all bees in one trial.

Although one could expect that the time to catch or photograph the insects would decrease with adaptation to the game, that was not observed. As shown in figures 7.a and b, while for some students that time decreases, for others that time can even increase.

We observed that as students learned how to play with confidence, they tried to obtain higher scores. The time to catch or photograph the insects has no effect on the score, provided the insects do not escape (i.e., disappear from the room). On the contrary, if the students miss the insects, that is if they try to target them by touching the screen and fail the target, the score decreases. Therefore, in order to obtain higher scores, the students paid more attention to the sounds and risked less often, even if that means taking a bit longer to catch the insects. They attempted to catch/photograph the insect only when they were sure they would not miss the target, that is, when they had a high degree of confidence on their estimation of the insects' location.

We also concluded that the game has impact on the players motor coordination skills. In order to rotate in the virtual environment, the game requires that the players rotate (turn) around themselves. We observed that on the first day of the trials, the blind participants tended to rotate the upper body, or moved only their arms and phone, keeping the feet static on the ground. The special education teacher who attended the tests concluded that this behavior was probably caused by the students' lack of security regarding the surrounding environment. Low vision participants did not show this behavior as pronounced as blind participants. On subsequent days, this behavior was less pronounced or disappeared. This shows that the game can have a positive impact in such motor coordination details.

All of the students who participated in the test enjoyed the game, were interested in playing it further and acquiring it to their mobile phones. Even subject number 2, who was the student with more difficulties had fun playing the game.

Some of the students (including student number 2) were so immerse in the game that even gave a few steps in the room trying to get closer to the insects (even though this type of movement was not necessary). While some of these students were dependent on other students to support them while walking, the game made them to forget about their limitations and compelled them to move freely in the room. The special education teacher who watched the test commented that she noticed many improvements on the mobility of the players while they were playing but that once they returned to their normal routines, the mobility inhibition returned. This suggests that playing the game further may have benefits on the students' independent mobility.

While most trials were done with only one student present in the room, there were a few trials in which students 5 and 6 were both in the room. We noticed a high competitiveness between the two, as both wanted to obtain the higher scores. For these two students, competitiveness was an important factor in the test, as the higher scores were obtained in the trials in which both students were together in the room.

The three-dimensional sonification of the game was also implicitly tested and approved by all subjects. The results from the Cockroach Hunt challenge showed that the sounds are easily located in the horizontal plane. On the other hand, the results from the Space Bees challenge show that the movements of the sound source on both horizontal and vertical plane are also easily noticeable.

5. CONCLUSION

In order to understand if the use of a well designed serious game can help blind children with their mobility skills, we have developed a vision rehabilitation game, the Audio Space Station, that uses 3D spatialized sound to create a virtual environment to train orientation and mobility skills of visually impaired children and youngsters. The game aims at training audio localization skills, simple body rotation motion, and helping the users with concepts that are usually covered at orientation and mobility classes. In addition, the game's entertaining characteristics have proven to help users on surpassing self-confidence problems commonly felt by blind children.

The audio virtual environment was created using 3D spatialized sound. We used HRTFs for this end. Using the HRTFs we were able to create a realistic virtual 3D sound environment, with which the user can interact by localizing sounds. The game was designed for smartphones and is characterized by its immersiveness and ease of use. To interact with the game, the user has only to rotate him/herself along with the phone, and touch the screen.

The game was especially designed for visually impaired children and teenagers, and therefore it can (and should) be played without seeing the graphics. Blind and low vision users can play the game without any help from sighted people, nonetheless, for convenience of the teacher or other sighted person who might be accompanying the user, the game has very simple graphics.

Some important characteristics of the game are the adaptive difficulty and reward system. The objective of these two characteristics is to maintain the players interested in the game without getting frustrated (when the difficulty is too high for them) or finding the game tedious (when the difficulty is too low).

In a usability test with visually impaired students, we observed that the students enjoyed the game and understood how to play it quite quickly. We also observed the immediate impact of the game in the motor coordination of the participants and their self confidence on moving freely in the room: Some participants had a very noticeable evolution on the movements they made, starting by turning around while keeping their feet static on the ground, to moving freely in the room without fear of the unknown surroundings. This shows that access to these activities can have a positive impact in this type of motor coordination abilities and self confidence of the users, especially for blind users without independent mobility.

We also observed an improvement on the localization skills of the participants. While on average, they made several missed attempts to catch the insects on the first trial of the game, on subsequent trials, they were able to catch the insects on the first attempt much more often.

As future work, we plan on having another challenge that requires the players to actually walk a few steps. According to the opinion of teachers of special needs children, this type of challenge would be very useful to help the children loose the fear of moving in unknown spaces and also to help those who do not like to use the cane on feeling more motivated to use it (as it would allow them to more easily walk those few steps needed in the challenge.)

6. ACKNOWLEDGMENTS

We thank Peter Colwell and Carlos Ferreira for their advice on orientation and mobility skills training techniques. Also, special thanks to teacher Joana Silvestre for her feedback on our work and help on running the tests. Finally, but not least, we thank all the students from Centro Helen Keller who took part in the tests.

This work was partially funded by NOVA LINCS through grant PEst- OE/EEI/UI0527/2011.

7. REFERENCES

[1] A. Marques, B.D. Silva, and N. Marques, "A influência dos videojogos no rendimento escolar dos alunos: uma experiência no 2º e 3º ciclo do ensino básico," *Educação, Formação & Tecnologia*, vol. 1, no. 4, pp. 17–27, 2011.

[2] F. Paraskeva, S. Mysirlaki, and A. Papagianni, "Multiplayer online games as educational tools: Facing new challenges in learning," *Computers & Education*, vol. 54, no. 2, pp. 498–505, 2010.

[3] J. Schick, "The decision to use a computer simulation," *The History Teacher*, vol. 27, no. 1, pp. 27–36, 1993.

[4] M. Lumbreras and J. Sánchez, "3D aural interactive hyperstories for blind children," in *International Journal of Virtual Reality*, 1998, pp. 119–128.

[5] J. Sánchez, N. Baloian, T. Hassler, and U. Hoppe, "Audiobattleship: Blind learners collaboration through sound," *Proceedings of ACM CHI*, pp. 798–799, 2003.

[6] J. Sánchez and T. Hassler, "AudioMUD: A multiuser virtual environment for blind people," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 15, pp. 16 – 22, 2007.

[7] F. Ferreira and S. Cavaco, "Mathematics for all: a game-based learning environment for visually impaired students," in *Proceedings of the IEEE Annual Frontiers in Education Conference (FIE)*, 2014.

[8] J. Carvalho, T. Guerreiro, L. Duarte, and L. Carriço, "Audio-based puzzle gaming for blind people," in *Proceedings of the Mobile Accessibility Workshop at MobileHCI (MOBACC)*, 2012.

[9] T. Westin, "Game accessibility case study: Terraformers - a real-time 3D graphic game," in *In Proceedings of the The International Conference on Disability, Virtual Reality and Associated Technologies*, 2004, pp. 95–100.

[10] C. Magnusson, A. Waern, K.R. Gröhn, Å Bjernryd, H. Bernhardsson, A. Jakobsson, J. Salo, M. Wallon, and P.-O. Hedvall, "Navigating the world and learning to like it: mobility training through a pervasive game," in *Proceedings of the ACM International Conference on Human Computer Interaction with Mobile Devices and Services*, 2011, pp. 285–294.

[11] J. Sánchez, F. Aguayo, and T. Hassler, "Independent outdoor mobility for the blind," *Virtual Rehabilitation*, pp. 114–120, 2007.

[12] J. Sánchez, M. Sáenz, and J.M. Garrido, "Usability of a multimodal video game to improve navigation skills for blind children," *ACM Transactions on Accessible Computing (TACCESS)*, vol. 3, no. 2, pp. 7, 2010.

[13] Y. Cohen, J. Dekker, A. Hulskamp, D. Kousemaker, T. Olden, C. Taal, and W. Verspage, "Demor, location based 3d audiogame," 2004.

[14] A. McMahan, "Immersion, engagement and presence," *The video game theory reader*, pp. 67–86, 2003.

[15] H. Ketamo, "An adaptive geometry game for handheld devices," *Educational Technology & Society*, vol. 6, no. 1, pp. 83–95, 2003.

[16] D. Simões and S. Cavaco, "An orientation game with 3d spatialized audio for visually impaired children," in *Proceedings of Advances in Computer Entertainment Technology Conference (ACE)*, 2014.

IMPLEMENTING A LOW-LATENCY PARALLEL GRAPHIC EQUALIZER WITH HETEROGENEOUS COMPUTING

Vesa Norilo, *

Centre for Music & Technology
University of Arts
Helsinki, Finland
vesa-petri.norilo@uniarts.fi

Math Verstraelen

Computer Architecture for Embedded Systems
University of Twente
The Netherlands
m.j.w.verstraelen@utwente.nl

Vesa Välimäki

Department of Signal Processing and Acoustics
Aalto University
Espoo, Finland
vesa.valimaki@aalto.fi

ABSTRACT

This paper describes the implementation of a recently introduced parallel graphic equalizer (PGE) in a heterogeneous way. The control and audio signal processing parts of the PGE are distributed to a PC and to a signal processor, of *WaveCore* architecture, respectively. This arrangement is particularly suited to the algorithm in question, benefiting from the low-latency characteristics of the audio signal processor as well as general purpose computing power for the more demanding filter coefficient computation. The design is achieved cleanly in a high-level language called *Kronos*, which we have adapted for the purposes of heterogeneous code generation from a uniform program source.

1. INTRODUCTION

The graphic equalizer is a signal processor with a variety of purposes, ranging from tonal correction of rooms and transducers to sound design and mastering [1]. One of the key benefits of the design is the immediate correspondence of user interface and frequency response. The quality and precision of this correspondence is key to a successful graphic equalizer design. In addition, many of its applications require low-latency processing—such as tonal correction of a loudspeaker system in a live situation.

In the subsequent sections, we present a recent parallel graphic equalizer design and show how to adapt it to a heterogeneous system of a PC and a WaveCore signal processor, utilizing a high-level programming language called Kronos. Kronos and WaveCore share a declarative programming methodology, greatly facilitating code generation. In addition, the signal rate factorization capabilities of Kronos are directly applicable to the problem of heterogeneous code generation.

This paper is organized as follows. In Section 2, *Background*, we discuss the history and state of art of the three central themes of this research: graphic equalization, dedicated signal processors, and musical programming languages. Section 3, *Methodology*, presents our equalization algorithm as well as describes the dedicated signal processor hardware, *WaveCore*, this research is founded on. The high-level programming method, *Kronos*, is also presented. Section 4 details the results of this work, discussing the aspects specific to heterogeneous computing and the adaptation of the equalizer algorithm to *WaveCore*, as well as avenues for future research. Section 5 concludes the article.

* This work was supported by the Emil Aaltonen Foundation.

2. BACKGROUND

2.1. Graphic Equalization

There are two types of equalizers, parametric and graphic ones [1]. In a parametric equalizer the user has access to the gain, center frequency, and bandwidth of the filter. A graphic equalizer can be composed of several parametric equalizers or other similar filters, but their center frequencies and bandwidths are fixed. In the graphic equalizer, the user only controls each band gain by using a set of sliders, which form approximately the desired magnitude response [1, 2, 3, 4]. The band gains are usually called 'command gains'. The main challenge in designing a graphic equalizer is the interaction of neighboring band filters. Problems arise when to adjacent command gains have a large difference.

In this work we implement one of the recent graphic equalizers, which solves the interaction problem by optimizing the parameters with a least-squares technique [4]. For processing the input signal, this graphic equalizer uses the parallel IIR filter structure, which is well suited to parallel computing [5].

2.2. Musical DSP Chips

Digital Signal Processing for music applications is usually based on a sound production model. In general, the nature and complexity of the application of such a model is directly related to the available computational capacity within the constrained boundaries of cost and technological state of the art. The nature of a sound production model can roughly be divided into three areas [6][7]. (1) Digitized analog models. The origin of these models are usually analog electronics which are mimicked by discrete-time models, e.g. discrete time models of analog stomp-box gui-

tar effects [8]. The required computational capacity for this class of production models is usually modest. The sampling rates are usually relatively low, and depend on the amount of non-linearity in the applied models (e.g. amp models) (2) Digital WaveGuide (DWG). DWG models are often abstract representations of physical/acoustical phenomena. An examples is DWG based reverberation [9].(3) Physical Modeling. This class of modeling is based on detailed aspects (e.g. geometry, material, etc.) of physical sound production devices [6]. The associated modeling is often based on multi-dimensional wave equations and often requires a vast amount of computational capacity.

2.2.1. Relevant processor technologies

DSP in embedded musical applications is predominantly implemented with dedicated DSP processor chips. These chips are optimized to cost and energy effectiveness. These DSP chips are usually programmed within a C-based development methodology. However, the inherent parallelism within these processors can often only be addressed with processor dependent compiler "intrinsics" which makes it often difficult to port existing application code to different chips. Likewise, the learning curve to program these devices is often steep. DSP processor chips are widely applied for digitized analog modeling and/or digital waveguide. Unlike the often data-flow oriented nature of the application, the programming methodology is merely imperatively, which conceptually adds an extra complexity dimension. Digital Audio Workstation (DAW) are software applications, mostly running on PC platforms. A wide variety of sound production models are implemented with this technology. Development environments for DAW are merely C-based. However, also declarative programming methodologies, like Faust [10], are applied within this domain. Typically, keeping the processing latency within acceptable limits is challenging within PC based real-time audio processing systems.

We mentioned that detailed physical modeling (i.e. based on finite difference modeling) requires a vast amount of computational capacity. Both General Purpose Processors (GPP) and DSP chips cannot deliver the required computational capacity necessary for detailed real-time physical modeling. Moreover, a large physical model may need to be partitioned over several GPP cores when the application is required to be mapped on a PC based computer platform. Given these problems, different processor technologies such as GPGPU (General-Purpose computing on Graphics Processing Units) or FPGA (Field Programmable Gate Array) are applied. The associated programming methodologies are usually data-flow oriented and hence link more naturally to a declarative or functional programming style rather than an imperative C-based programming environment.

2.2.2. Implications of the multi-core era

The advance of semiconductor technologies, related to Moore's law, has a huge impact on the evolution of processor technologies (GPP, GPU, FPGA, DSP). Most dominantly, the energy- ILP- (Instruction Level Parallelism) and memory walls have driven the processor evolution into the "multi-core" era [11]. Multi-core has a huge impact on programming methodologies, linked to scalability (number of processor cores). In particular, the imperatively based programming methodologies and associated multi-core processors (e.g. DSP, GPP) are faced with a big challenge. Parallelism needs to be extracted from the imperatively described algorithm and subsequently partitioned and mapped on a multi-core processor. On the contrary data-flow oriented declarative programming styles, associated to GPGPU or FPGA are naturally scalable and are inherently "Moore-proof" to a larger extend. This is exactly the reason why GPUs and FPGAs are gaining attention as processing platforms for scalable and computational intensive algorithms like detailed physical modeling. An FPGA is basically a fabric of primitive programmable logic functions and embedded memories which can be connected in an almost arbitrary way. This means that a non-configured FPGA can be seen as an undefined chip. Therefore, designing an FPGA based application implies that the configurable circuitry of the FPGA (i.e. the "softcore") needs to be developed in a Hardware Description Language (HDL). This softcore development in itself often is a costly process which is partially caused by the abstraction level of commonly applied HDL methodologies (e.g. VHDL). Emerging methodologies, based on mathematical modeling using functional languages [12] are intended to raise the abstraction level and hence shorten the development cycle of softcores. On top of softcore development, the HW/SW interface and the software part needs to be developed. This makes FPGA design a multi-disciplinary task which requires both HW and SW engineering skills.

2.3. Programming Signal Processors

2.3.1. Programming for DSP Chips

Perhaps the most utilized method of programming signal processors is via low-level languages such as C or the native assembly language of the target chip. Libraries of typical signal processing primitives (such as digital filters, delay lines and signal transforms) are offered by chip vendors and third parties alike. This approach is relatively straightforward and tends to result in efficient utilization of the hardware, as long as the algorithms being implemented can be expressed in terms of typical primitives.

When the algorithm in question is more complicated, the traditional method of low level programming is no longer quite as attractive. Implementation of novel processing primitives typically requires a high level of expertise on both the algorithm and the hardware in question, and the resulting work is tightly coupled with a specific architecture. These factors make signal processing code averse to being portable.

2.3.2. High-Level DSP Languages

Several approaches exist for programming signal processors without a high level of C or Assembly expertise. National Instruments LabVIEW [13] is a graphical interface to the *G language*, which is based on data flow, similar to our methodology. However, the LabVIEW system is proprietary and opaque, so utilization of custom hardware like the *WaveCore* is not easily achievable.

Another commercial DSP design methodology is based on MathWorks' MatLab and SimuLink (e.g. [14]). SimuLink supports generation of C code or direct synthesis of signal processing cores via a hardware description language. These approaches are focused on multi-domain simulation and are likely too elaborate for design and application of musical signal processors.

The advantage of our proposed method is in its domain-specificity; as both *Kronos* and *WaveCore* are specifically designed for musical signal processing, several assumptions can greatly simplify the task for automatic heterogeneous code generation. In the present

study, we utilize a straightforward clock domain mapping for code factorization – discussed further in Section 3.3.2.

2.3.3. Musical Programming Languages

Musical Programming Languages are a topic of active research. The present study is based on our *Kronos* programming language [15]. This section offers a brief overview of influential design paradigms and language implementations specific to musical signal processing.

The most widely adopted paradigm for musical signal processing is the unit generator concept [16, pp. 787–810]. The ugen concept was solidified by the MUSICn family, of which CSound [17] is the contemporary example. Pure Data [18] and SuperCollider [19] are important developments of the concept. Ugen programming is declarative by nature. The programming task focuses on signal flow and topology, rather than the chronological ordering of program statements.

Ugen languages can be criticized for their lack of higher level program constructs. Especially graphical front ends such as Pure Data [18] – an example of a very successful implementation – make it quite difficult to express data types, control flow or program composition, all of which enhance programmer productivity.

Several attempts have been made to address this problem. CLM [20] attempts to merge the ugen concept with Common Lisp [21], an acclaimed high-level language. Since making high-level signal processing programs efficient is difficult, CLM delimits signal processors to a certain subset of Common Lisp and transcompiles that to C. The resulting programming paradigm is a mixture of styles, featuring a C-like programming style with Common Lisp syntax.

SuperCollider [19] is a more actively developed idea in the same vein. SuperCollider is influenced by SmallTalk (e.g. [22]), and integrates ugens built in the more performant C language. As a result, SuperCollider programs are tied with the implementation of the run time library, and the actual signal processors are opaque to the user.

Faust [23] attempts to address signal processing in a functional high-level idiom. Faust features an expressive block diagram composition system capable of succinctly describing many typical signal flows. Faust transcompiles to C, with recent work aimed towards direct compilation using the LLVM [24] framework.

Kronos [15] is the language used and adapted for the present study. Influenced by Faust, it offers a functional signal processing paradigm, enhancing it with advanced metaprogramming capabilities, automatic factorization and an advanced compiler pipeline [25]. Kronos makes use of LLVM [24] for native code generation. For the purposes of the present study, an experimental WaveCore code generator has been developed, along with facilities for heterogeneous compilation of a uniform source program for several distinct hardware targets.

3. METHODOLOGY

3.1. Graphic EQ Algorithm

In this work we implement a recently developed parallel graphic equalizer [4]. The filter itself is a parallel IIR structure in which each filter block has a special second-order transfer function: a second-order denominator transfer function but a first-order numerator transfer function. Additionally, there is a direct path with a real weight from the input to the output.

The poles of the graphic equalizer are set in advance at predesigned frequencies determined by the frequency resolution of the graphic equalizer. For example, when a third-octave graphic equalizer is designed, the poles go at 31 standard frequencies between 20 Hz and 20 kHz (20 Hz, 25 Hz, 31.5 Hz, 40 Hz etc.). To obtain high accuracy, additional poles are assigned at 10 Hz and between each standard center frequency, so that there will be altogether 62 poles. The pole radii as chosen so that the magnitude responses associated with neighboring poles meet at their -3 dB points. All of this is done off-line before running the filter. The mathematics related to this design are detailed in [4].

During real-time operation, the user can adjust the command gains of the graphic EQ. To achieve a great accuracy, this EQ design uses least-squares optimization to adjust the numerator (feedforward) coefficients, two per pole. This is similar to FIR filter design and only requires a matrix operation. However, the nonnegative weighting function is needed to ensure that attenuation is implemented correctly. As a result, every time a command gain is changed, the matrix inversion needs to be executed and all feedforward coefficients of the graphic EQ updated [4].

3.2. WaveCore – a High Performance Audio DSP Core

WaveCore is a programmable many-core processor which is optimized to real-time acoustical and physical modeling. This processor concept aims to address the scalability problem which is described in section 2.2.2. Target applications are all the sound-production models which are mentioned in section 2.2. Classical "digitized analog", and digital waveguide modeling with ultra-low latency using WaveCore has been published in [26]. A recently carried out feasibility analysis on the usability of WaveCore for real-time physical modeling of musical instruments using finite-difference time-domain techniques has yielded promising results [27].

3.2.1. Programming Model

The WaveCore processor can be programmed by means of a description of a data-flow graph. Such a graph consists of one or more processes, that are interconnected by means of edges. The data-elements (i.e. tokens) that are carried over the edges consist of one or more Primitive Token Element (PTE, a floating point number). All processes in the graph are periodically executed (i.e. fired) explicitly by a centralized scheduler, where multi-rate execution is fully supported. When a process is fired, it consumes one token per inbound edge and produces one token per outbound edge. A process may be composed by one or more process partitions (WPP). Ultimately each process consists of a number of interconnected Primitive Actors (PA). The PA has at most two inbound edges and as such consumes at most two PTEs x_1 and x_2 when the PA is fired. The PA produces one PTE $y^{n+\lambda}$ when it is fired through an optional delay-line.

The programming model supports a limited set of different PAs. A few examples are: (1) C-type PA: $y^{n+1} = p$, (2) MAD-type PA: $y^{n+\lambda} = p \cdot x_1^n + x_2^n$, (3) ADD-type PA: $y^{n+\lambda} = x_1^n + x_2^n$ (4) MUL-type PA: $y^{n+\lambda} = x_1^n.x_2^n$, and (5) AMP-type PA: $y^{n+\lambda} = p.x_1^n$.

This data-flow oriented programming model enables a declarative way of implementing a wide variety of signal processing algorithms, like flanger, wah-wah, reverberation, EQ, etc. [26]. The declarative nature of the programming methodology enables

a straightforward mapping to functional languages like Kronos, as we will outline in Section 3.3.

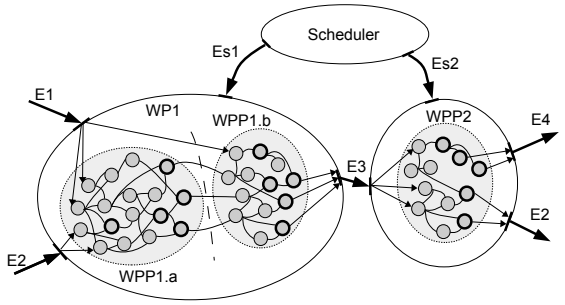


Figure 1: Data-flow oriented WaveCore programming model.

3.2.2. Processor Architecture

The WaveCore processor consists of a scalable cluster of Processing Units (PU). Each PU embodies a small Reduced Instruction-Set Computer (RISC). The instruction set of the PU is fully optimized to the execution of a group of PAs (WPP) where each PA is mapped on a single instruction. The PU can be classified as a pipelined Harvard processor (instructions and operands are located in separate memories). This implies that the PU is capable of executing one PA per clock cycle. The heart of the PU is a single precision floating-point ALU which supports basic arithmetic operations such as add, subtract, multiply, multiply/add, divide, compare, etc. Next to the instruction pipeline the PU is equipped with a DMA controller, called Load/Store Unit (LSU). This LSU is loosely coupled to the processor pipeline and is responsible for moving token data and/or delay-line data between external memory and the processor pipeline. An on-chip network, called Graph Partition Network (GPN) connects all PUs.

The WaveCore compiler automatically partitions and maps a WaveCore process onto the processor hardware. Each WPP is scheduled and mapped on a PU, and the connections between the WPPs are mapped on the GPN. The compiler also takes care of external memory allocation and as such maps all tokens and delay-lines on the LSU parts of the associated PUs.

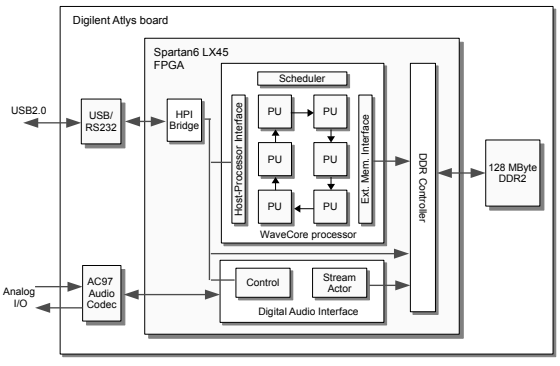


Figure 2: WaveCore processor on Atlys FPGA board.

The WaveCore processor is implemented as a soft-core in a Hardware Description Language (HDL). We have integrated a core instance with 6 PUs in System-on-Chip which subsequently is mapped on a Xilinx Spartan6 LX45 device on the Digilent Atlys development board. The block diagram of this development board and FPGA is depicted in fig. 2. The PU cluster can be initialized (i.e. loading a compiled WaveCore program to the embedded instruction memories within the PU cluster) by means of en externally connected host computer through the USB interface. Run-time control (i.e. run-time modification of control-tokens) is possible through the same USB interface, which also has full access to the external DDR2 memory on the board. The board contains an AC97 compliant audio codec chip (stereo audio DAC and ADC). This codec is enabled to stream autonomously into/from the DDR memory. The WaveCore processor cluster itself, which is capable of executing a process graph with up to 12288 PAs at 44.1kHz audio rate, is also capable of autonomously accessing the DDR memory.

3.2.3. The Graphic Equalizer as a WaveCore process graph

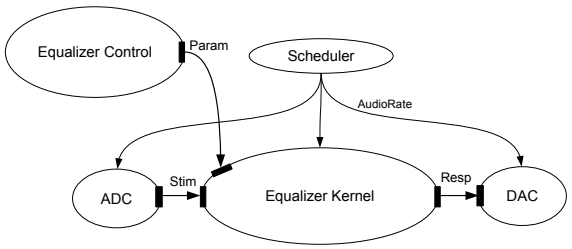


Figure 3: Process graph of equalizer

The equalizer application as a WaveCore process graph is depicted in fig.3. The "Equalizer Control" process is mapped on the externally connected computer and the "Equalizer Kernel" process runs on the WaveCore processor. This kernel process consists of one WPP and hence utilizes only one out of the 6 PUs. The "param" edge is mapped on the USB interface and hence implements the run-time control of the kernel process. The associated control tokens are allocated in DDR memory. The ADC and DAC processes are mapped on the AC97 codec chip and the associated audio edges "Stim" and "Resp" are mapped on DDR memory. The tokens which are moved over the "Stim" edge are produced by the AC97 codec and consumed by the kernel process which is mapped on the WaveCore. Similarly, the tokens which are moved over the "Resp" edge are produced by the kernel process which is mapped on the WaveCore, and consumed by the DAC process on the AC97 chip. The "Stream Actor" block within the "Digital Audio Interface" takes care of moving tokens between the AC97 chip and DDR memory. The scheduler periodically generates fire tokens to the real-time kernel, ADC and DAC processes at audio rate. Note that the "Equalizer Control" process is not linked to the scheduler, and responds to human interaction. The processing latency is short because there is almost no buffering between the ADC/DAC processes and the equalizer kernel (one token per edge, and hence 2 T_s streaming latency).

3.3. Kronos – a High Level Music DSP Language

Kronos [25] is a high level functional signal processing language designed for musical applications. The declarative dataflow principle is common to Kronos and WaveCore, making WaveCore a natural compile target for the Kronos compiler. An experimental Kronos code generator for WaveCore was implemented for the purposes of this research.

The user-facing aspect of the Kronos language resembles a high level functional programming language with very little data type notation, along with features designed for musical signal processing. The most significant of these include language level memory operators, such as unit delays and recursions, that exhibit pure functional semantics[28] and are reified into stateful constructs by the compiler. In addition, the compiler performs full-program type derivation and dataflow analysis to generate a highly performant statically typed run time representation that can be automatically factored into several clock domains. The factorization process results in a set of driver routines that share and mutate a state buffer that represents the signal memories of the running application as well as the requisite state for transferring signals between clock regions. For a more detailed discussion, the reader is referred to prior work[15].

3.3.1. WaveCore Code Generation

The lowered runtime program representation generated by Kronos is typically compiled into native code with LLVM[24], an open source compiler back end framework. Generating code for the WaveCore compiler is much simpler, as the program format of the WaveCore compiler is declarative, exactly like the intermediate representation produced by the Kronos compiler. The WaveCore backend is a simple idiom translator implemented as a pattern matcher, in which a group of N Kronos primitives is mapped into a group of M WaveCore Primitive Actors. As WaveCore is much more streamlined than a typical CPU, not all Kronos operations map efficiently or even at all onto WaveCore programs, but the overlap is considerable, especially considering that the two designs were not coordinated initially. As of this writing, most Kronos programs dealing with single precision floating point DSP can be mapped to WaveCore, including the equalizer design discussed in this paper.

3.3.2. Heterogeneous Code Generation

Since Kronos already does signal clock factorization, heterogeneous code generation simplifies to the problem of adopting different compile targets for different signal clocks, and generating a transport layer to enable the clock regions to communicate.

Our heterogeneous code generation technique utilizes the Kronos program representation after the global type derivation and data flow analysis passes have completed. At this point, we have a static signal flow graph annotated with clock regions. Normally, the compiler would generate driver routines for each clock region, but for heterogeneous compiling, we have added an option to filter the set of clock regions included in the current compilation unit.

The equalizer design features two main clock regions: one is the audio clock region, including the parallel biquad filter bank and its summation. The other is the control clock region, which is driven by the user interface and includes the coefficient generation code. Because an update to any command gain causes recomputation of all the feed forward coefficients, it is best to use a single clock region for all the command gains.

Compiling the equalizer program source once for WaveCore, including only the audio clock region, and once for PC, including the control region, results in the requisite program objects for the two architectures. These program objects correspond to the *Equalizer Control* and *Equalizer Kernel* shown in Figure 3. Next, we address the transfer of filter coefficients from the PC component to the WaveCore component.

Since the Kronos data flow analysis detects clock region boundaries, communication can be enabled by special handling of the boundaries that involve a transition between compile targets. Since version 2.0, WaveCore has a well-defined protocol for external control. A ForeignProcess description is generated by the Kronos/WaveCore compiler, including tokens for each signal graph edge that crosses from the PC to the DSP. In the case of the graphic equalizer, these edges represent the feed forward coefficients of the biquad filter bank. On the PC side, such edges are represented by a section of the state buffer generated by the compiler for the Kronos program.

To facilitate transport, an option to invoke a user-defined callback function was added to the compiler whenever a particular boundary is updated. A special compiler driver uses the Kronos JIT Compiler to generate the PC-specific section of the signal processing system, and hooks into the callback mechanism to propagate boundary edge updates to the WaveCore submodule. These updates are sent over the serial port to the WaveCore board as per the control protocol.

Table 1: Equalizer error maxima, decibels

Setting	RM	EQ4	PGE
All up	9.8	3.6	0.00
Zigzag	6.3	2.8	0.75
Every 3rd up	4.1	1.5	0.32

4. DISCUSSION

4.1. Equalizer Response

The presented equalizer algorithm offers a highly precise frequency response. Three command gain settings were used to find the maximum error in the actual magnitude response of the equalizer in comparison to the command gains.

Table 1 lists the results of our algorithm (PGE) contrasted to the second order Regalia-Mitra EQ (MR) [29] and a higher order design (EQ4) [2]. The *All up* settings features all the command gains set to $+12$ dB. In *Zigzag*, the command gains alternate between $+12$ dB and unity. The final setting of *Every 3rd up* features one band at $+12$ dB followed by two bands at unity.

The proposed algorithm features the smallest error maxima, staying within a decibel of the target curve on all settings. For a more detailed evaluation and comparison of the PGE algorithm, the reader is referred to [4].

4.2. Performance

As the described implementation is heterogeneous, the performance characteristics that concern us are also varied. As the audio processing on WaveCore obeys hard real time constraints, we are mostly concerned about chip utilization.

Table 2: Equalizer performance summary

Equalizer Kernel	PAs	of PU	% of chip
per channel @ 44.1kHz	740	81%	13.5%
Equalizer Control	avg	max	min
time per update	32ms	51ms	31ms

On the PC side, the computational complexity manifests as control latency – the delay between user interaction and the corresponding change in the equalizer response. This latency is dominated by the time required for coefficient computation.

Table 2 summarizes the central performance characteristics of our solution. Our current WaveCore chip is a cluster of six Processing Units running at 86MHz, achievable on the Digilent Atlys board wiath a Xilinx Spartan6 FPGA. The equalizer control timings were measured on a Windows PC with an Intel Core i7 CPU running at 2.8GHz. Minimum, maximum and average control latency were collected from 1000 simulated coefficient updates. As shown, one audio channel can be equalized by one PU in hard real time. This results in a throughput latency of two sample periods, excluding the latency of A/D/A conversion. Our WaveCore cluster is computationally capable of 6 channels of real time equalization, although the Atlys board is limited to stereo audio I/O.

4.3. Future Work

In the future, the design could be adapted to an embedded setting with, for example, a low power CPU combined with a WaveCore chip and a dedicated control surface. Such a setup would function well as a dedicated hardware equalizer. Alternatively, a room correction module could be developed based on the technology, with a PC-based analysis and filter design solution combined with hardware equalization.

The support of the Kronos language on WaveCore could also be further developed. The areas of interest range from the emulation of double precision floating point arithmetic to automatic factorization of programs to several concurrent WaveCore process partitions to better utilize parallelism.

5. CONCLUSION

In this paper, we presented an implementation of a recent graphic equalization algorithm on a heterogeneous computing platform consisting of a commodity PC and a WaveCore-based signal processing board. The system was shown to exhibit excellent latency characteristics due to the use of dedicated hardware, as well as excellent precision due to the advanced coefficient computation technique made possible by a powerful CPU. The graphic equalizer serves as an example of our proposed heterogeneous signal processor development workflow, which enables automatic factorization of a single source program to two distinct program objects.

The Kronos compiler suite is available in binary and source form at `https://bitbucket.org/vnorilo/k3`. WaveCore is available for the published Digilent Atlys FPGA development board.

6. ACKNOWLEDGMENTS

Vesa Norilo's work has been supported by the Emil Aaltonen foundation.

7. REFERENCES

[1] D. A. Bohn, "Operator adjustable equalizers: An overview," in *Proc. AES 6th Int. Conf.*, Nashville, TN, May 1988, pp. 369–381.

[2] M. Holters and U. Zölzer, "Graphic equalizer design using higher-order recursive filters," in *Proc. Int. Conf. Digital Audio Effects*, Montreal, Canada, Sept. 2006, pp. 37–40.

[3] J. Rämö and V. Välimäki, "Optimizing a high-order graphic equalizer for audio processing," *IEEE Signal Process. Lett.*, vol. 21, no. 3, pp. 301–305, Mar. 2014.

[4] J. Rämö, V. Välimäki, and B. Bank, "High-precision parallel graphic equalizer," *IEEE/ACM Trans. Audio Speech Language Processing*, vol. 22, no. 12, pp. 1894–1904, Dec. 2014.

[5] J. A. Belloch, B. Bank, L Savioja, A. Gonzalez, and V. Välimäki, "Multi-channel IIR filtering of audio signals using a GPU," in *Proc. IEEE Int. Conf. Acoust. Speech and Signal Processing (ICASSP-2014)*, Florence, Italy, May 2014, p. 6692–6696.

[6] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, Wiley, 2009.

[7] U. Zölzer (ed.), *DAFX: Digital Audio Effects*, Wiley, second edition, 2011.

[8] V. Välimäki, S. Bilbao, J. O. Smith, J. S. Abel, J. Pakarinen, and D. Berners, *DAFX: Digital Audio Effects*, chapter 'Virtual analog effects', pp. 473–522, U. Zölzer (ed.), Wiley, second edition, 2011.

[9] J. O. Smith, "A new approach to digital reverberation using closed waveguide networks," in *Proc. Int. Computer Music Conf.*, Vancouver, Canada, Sept. 1985, pp. 47–53.

[10] Y. Orlarey, D. Fober, and S. Letz, "Faust (programming language)," Available at `https://faust.grame.fr`, accessed Dec. 8, 2013.

[11] S. Borkar and A. A. Chien, "The future of microprocessors," *Communications of the ACM*, vol. 54, no. 5, pp. 67–77, 2011.

[12] J. Kuper et al., "Cλash, from haskell to hardware," Available at `https://http://www.clash-lang.org/`, accessed June 8, 2015.

[13] J. Travis and J. Kring, *LabVIEW for Everyone: Graphical Programming Made Easy and Fun (National Instruments Virtual Instrumentation Series)*, Prentice Hall PTR, 2006.

[14] A. Krukowski and I. Kale, "Simulink/Matlab-to-VHDL route for full-custom/FPGA rapid prototyping of DSP algorithms," in *Proc. Matlab DSP Conf. (DSP'99)*, 1999, pp. 1–10.

[15] V. Norilo, "Introducing Kronos—A novel approach to signal processing languages," in *Proc. Linux Audio Conf.*, Maynooth, Ireland, May 2011, pp. 9–16.

[16] C. Roads, *The Computer Music Tutorial*, MIT Press, Cambridge, MA, 1996.

[17] R. Boulanger, *The Csound Book*, MIT Press, 2000.

[18] M. Puckette, "Pure data: another integrated computer music environment," in *Proc. 1996 Int. Computer Music Conf.*, 1996, pp. 269–272.

[19] J. McCartney, "Rethinking the computer music language: SuperCollider," *Computer Music Journal*, vol. 26, no. 4, pp. 61–68, 2002.

[20] B. Schottstaedt, "Machine tongues XVII: CLM: Music V meets Common Lisp," *Computer Music Journal*, pp. 30–37, 1994.

[21] G. L. Steele, *Common Lisp*, vol. 2, Digital Press, 1984.

[22] D. Ingalls, T. Kaehler, J. Maloney, S. Wallace, and A. Kay, "Back to the future: The story of Squeak, a practical Smalltalk written in itself," in *ACM SIGPLAN Notices*. ACM, 1997, vol. 32, pp. 318–326.

[23] Y. Orlarey, D. Fober, and S. Letz, "Syntactical and semantical aspects of Faust," *Soft Computing*, vol. 8, no. 9, pp. 623–632, 2004.

[24] C. Lattner and V. Adve, "LLVM: A compilation framework for lifelong program analysis & transformation," in *Proc. Int. Symp. Code Generation and Optimization (CGO 2004)*, 2004, pp. 75–86.

[25] V. Norilo, "Recent developments in the Kronos programming language," in *Proc. Int. Computer Music Conf.*, Perth, Australia, 2013.

[26] M. Verstraelen, G.J.M. Smit, and J. Kuper, "Declaratively programmable ultra low-latency audio effects processing on FPGA," in *Proc. 17th Int. Conf. Digital Audio Effects*, Erlangen, Germany, Sept. 2014, pp. 263–270.

[27] M. Verstraelen, F. Pfeifle, and R. Bader, "Feasibility analysis of real-time physical modeling using WaveCore processor technology on FPGA," in *Proc. 3rd Vienna Talk on Music Acoustics*, Vienna, Austria, 2015.

[28] P. Hudak, "Conception, evolution, and application of functional programming languages," *ACM Computing Surveys*, vol. 21, no. 3, pp. 359–411, 1989.

[29] P. A. Regalia and S. K. Mitra, "Tunable digital frequency response equalization filters," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 35, no. 1, pp. 118–120, Jan. 1987.

ADAPTIVE MODELING OF SYNTHETIC NONSTATIONARY SINUSOIDS

Marcelo Caetano[*]
INESC TEC
Sound and Music Computing Group
Porto, Portugal
mcaetano@inesctec.pt

George Kafentzis
University of Crete
Department of Computer Science
Heraklion, Greece
kafentz@csd.uoc.gr

Athanasios Mouchtaris[†]
FORTH
SPL - Institute of Computer Science
Heraklion, Greece
mouchtar@ics.forth.gr

ABSTRACT

Nonstationary oscillations are ubiquitous in music and speech, ranging from the fast transients in the attack of musical instruments and consonants to amplitude and frequency modulations in expressive variations present in vibrato and prosodic contours. Modeling nonstationary oscillations with sinusoids remains one of the most challenging problems in signal processing because the fit also depends on the nature of the underlying sinusoidal model. For example, frequency modulated sinusoids are more appropriate to model vibrato than fast transitions. In this paper, we propose to model nonstationary oscillations with adaptive sinusoids from the extended adaptive quasi-harmonic model (eaQHM). We generated synthetic nonstationary sinusoids with different amplitude and frequency modulations and compared the modeling performance of adaptive sinusoids estimated with eaQHM, exponentially damped sinusoids estimated with ESPRIT, and log-linear-amplitude quadratic-phase sinusoids estimated with frequency reassignment. The adaptive sinusoids from eaQHM outperformed frequency reassignment for all nonstationary sinusoids tested and presented performance comparable to exponentially damped sinusoids.

1. INTRODUCTION

Music and speech contain different types of nonstationary oscillations. The attack of many musical instruments presents transients due to nonstationarities [1]. Percussive sounds feature very sharp onsets with highly nonstationary oscillations [2]. Expressiveness in performance such as *tremolo*, *vibrato*, *glissando*, and *portamento* generally results in amplitude and frequency modulations [3, 4]. Similarly, speech sounds such as consonants contain transients [5]. Consonants known as *plosives* feature a sharp onset [6]. Expressivity in speech used to convey emotions, for example, results in prosodic contours [5] or modulations in frequency and amplitude, while vibrato can be said to characterize singing [7].

Sinusoidal modeling is a popular parametric representation for speech and music. Sinusoidal models are widely used in speech and music processing for coding [8, 9, 10], analysis and synthesis [11, 12, 13, 14, 15, 16, 17], enhancement [18, 19, 20, 21], modifications and transformations [12, 15, 22, 23, 24, 25, 26].

The general problem of fitting a sum of sinusoids to a signal is of great interest in many scientific areas. Thus, many algorithms have been developed for accurate estimation of the sinusoidal parameters. For speech and musical sounds, the algorithms can be separated into four categories, namely spectral peak-picking [11, 12], analysis-by-synthesis [15, 27, 28, 29], least squares [30, 31, 32], and subspace methods [33, 34, 35].

Polynomial phase signals [36] have been used to model nonstationary oscillations. McAulay and Quatieri [11] were possibly the first to propose to interpolate the phase values estimated at the center of the analysis window with cubic polynomials. Quadratic polynomials [37] were proposed as an alternative. Girin *et al.* [38] investigated the impact of the order of the polynomial used to represent the phase. They concluded that a polynomial of order 5 does not improve the modeling performance considerably to justify the increased complexity.

The time-frequency reassigned spectrogram [39] was developed to better represent nonstationary oscillations with the short-time Fourier transform. Reassignment is widely used [40, 41, 42] to estimate the parameters of the sinusoidal model. The derivative analysis method [43, 44] was later shown [45] to be theoretically equivalent to the reassignment method.

More recently, adaptive sinusoidal models [31, 32, 46] have gained attention due to their ability to adapt to the local characteristics of the signal via an iterative parameter re-estimation process. Previous works have modeled speech [16, 32] and monophonic musical instrument sounds [17, 47] as a sum of adaptive sinusoids. These studies focused on modeling speech and musical instrument sounds recorded under controlled conditions instead of expressive conversations or music performances. Consequently, the sounds do not feature the prosodic contours or embelishements that result in challenging nonstationary modulations.

[*] This work is financed by the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project "UID/EEA/50014/2013."

[†] This work is supported in part by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant 644283.

In this work, we investigate the ability of adaptive sinusoids from eaQHM [46] to model nonstationary oscillations. We generated synthetic nonstationary sinusoids with different amplitude and frequency modulations and compared the modeling performance of adaptive sinusoids estimated with eaQHM, exponentially damped sinusoids estimated with ESPRIT, and log-linear amplitude quadratic-phase sinusoids estimated with time-frequency reassignment. We designed nonstationary sinusoids with controlled amplitude and frequency modulations that mimic specific features of nonstationary oscillations found in expressive music and speech, such as *tremolo* and *vibrato*. In this article, we focus on modeling monocomponent signals composed of one of these nonstationary sinusoids to compare the ability of each model to capture that specific feature. We measure the modeling accuracy in the time domain with the signal-to-reconstruction-error ratio (SRER). The SRER is the ratio in dB between the energy in the original signal and in the modeling residual.

The next section briefly describes the underlying sinusoidal model for the exponentially damped sinusoidal model (EDSM), reassigned sinusoidal model (RSM), and extended adaptive quasi-harmonic model (eaQHM). Then we describe the synthetic nonstationary sinusoids used in this work. Next, we present the modeling performance of EDSM, RSM, and eaQHM for the nonstationary signals designed, followed by a discussion of the results. Finally, the conclusions and perspectives are presented.

2. SINUSOIDAL MODELS

Sinusoidal models implicitly assume that each partial (e.g., oscillatory mode) can be described by a time-varying sinusoid $s\left(t\right)$ as

$$s\left(t\right) = A\left(t\right)\cos\left[\Phi\left(t\right)\right], \tag{1}$$

where $A\left(t\right)$ is the time-varying amplitude and $\Phi\left(t\right)$ is the time-varying phase, jointly called the instantaneous components of the signal. $A\left(t\right)$ and $\Phi\left(t\right)$ describe respectively the long-term amplitude and frequency modulations of each partial along the total duration of the sound. Usually, these long-term variations are approximated by piece-wise functions inside short-term signal frames $x\left(t\right)$ typically lasting miliseconds obtained as

$$x_k\left(t\right) = s\left(t\right)w\left(t - k\tau\right), \;\; 0 \leq k \leq N-1, \tag{2}$$

where k is the frame number, τ is the time shift (hop size), and $w\left(t\right)$ is a window function with length L that is zero outside the support L. Typically, $\tau < L$ so that the frames overlap and $s\left(t\right)$ is modeled as N frames $x_k\left(t\right)$ viewed through a sliding window $w\left(t - k\tau\right)$ centered at τ as follows

$$s\left(t\right) = \sum_{k=0}^{N-1} x_k\left(t\right) = \sum_{k=0}^{N-1} s\left(t\right)w\left(t - k\tau\right). \tag{3}$$

Eq. (3) holds for windows $w\left(t\right)$ that satisfy the constant overlap-add (COLA) [48] constraint $\sum_{k=0}^{N} w\left(t - k\tau\right) = 1$, valid only for specific values of τ.

When $s\left(t\right)$ is assumed to be locally stationary, $x\left(t\right)$ becomes

$$x\left(t\right) = A\cos\left(\omega t + \theta\right) \tag{4}$$

modeled as a sinusoid with constant amplitude A, constant frequency $\omega = 2\pi f_0$ and constant phase shift θ. In this case, the long-term model for the partial $s\left(t\right)$ is composed of piece-wise stationary oscillations only capable of capturing relatively stable amplitude and frequency modulations. However, nonstationary oscillations commonly vary enough inside the frame to require a dedicated short-term model. In what follows, we describe the underlying short-term signal model $x\left(t\right)$ for the nonstationary sinusoidal models used in this work, namely exponentially damped sinusoidal model (EDSM), reassigned sinusoidal model (RSM), and extended adaptive quasi-harmonic model (eaQHM).

2.1. Exponentially Damped Sinusoidal Model (EDSM)

EDSM assumes that $x\left(t\right)$ can be approximated by the underlying signal model

$$x\left(t\right) = \exp\left(\lambda + \mu t\right)\cos\left(\omega t + \theta\right), \tag{5}$$

where $A\left(t\right) = \exp\left(\lambda + \mu t\right)$ is the temporal envelope and $\Phi\left(t\right) = \omega t + \theta$ is the time-varying phase. The short-term frame $x\left(t\right)$ in EDSM is simply modeled as a stationary sinusoid with constant frequency ω modulated in amplitude by an exponential envelope controlled by λ and μ. $A\left(t\right)$ grows exponentially when $\mu > 0$, decays when $\mu < 0$, and is constant if $\mu = 0$.

The literature has shown [33, 34, 35, 49] that subspace methods render accurate parameter estimation for EDSM. This work uses ESPRIT to fit the parameters of EDSM [35].

2.2. Reassigned Sinusoidal Model (RSM)

RSM can be shown to render good modeling performance [45] when $x\left(t\right)$ can be approximated by the underlying signal model

$$x\left(t\right) = \exp\left(\lambda + \mu t\right)\cos\left(\psi t^2 + \omega t + \theta\right), \tag{6}$$

where $A\left(t\right) = \exp\left(\lambda + \mu t\right)$ is the temporal envelope and $\Phi\left(t\right) = \psi t^2 + \omega t + \theta$ is the time-varying phase. The short-term frame $x\left(t\right)$ in RSM is approximated as a sinusoid with quadratic phase (quadratic frequency ψ, linear frequency ω, and phase shift θ) modulated in amplitude by an exponential envelope controlled by λ and μ similarly to EDSM.

The parameters of the model are estimated using the time-frequency reassignment method [40, 41, 42, 45]. This work uses the DESAM [50] toolbox to fit the parameters of RSM.

2.3. The extended adaptive Quasi-Harmonic Model (eaQHM)

The assumption behind eaQHM is that speech and musical sounds can be approximated by a sum of M quasi-harmonic, highly nonstationary, AM-FM modulated partials $s_m\left(t\right)$. Each partial is further modeled *inside* the analysis frame as a short-term $x\left(t\right)$ which can be approximated by the underlying signal model

$$x\left(t\right) = \left(\lambda + \mu t\right)\cos\left(\psi t^2 + \omega t + \theta\right), \tag{7}$$

where $A\left(t\right) = \left(\lambda + \mu t\right)$ is the temporal envelope and $\Phi\left(t\right) = \psi t^2 + \omega t + \theta$ is the time-varying phase. The short-term frame $x\left(t\right)$ in eaQHM is implicitly modeled as a sinusoid with quadratic phase (quadratic frequency ψ, linear frequency ω, and phase shift θ) modulated in amplitude by a *linear* envelope controlled by λ and μ. $A\left(t\right)$ grows linearly when $\mu > 0$, decays when $\mu < 0$, and is constant if $\mu = 0$.

The parameters λ, μ, ψ, ω, and θ are iteratively adapted from successive steps of parameter estimation using least squares [46]. Adaptation arises from a sequence of parameter re-estimation steps based on successive refinements of the model basis functions, which

come directly from (7). The complete parameter estimation algorithm is described elsewhere [46].

3. SYNTHETIC NONSTATIONARY SINUSOIDS

The algorithms will be tested on synthetic nonstationary sinusoids to show the properties of each model, along with their corresponding advantages and disadvantages. The following parameters were used to generate the synthetic nonstationary sinusoids, sampling frequency $F_s = 16$ kHz and total length $N = 1600$ samples, corresponding to 100 ms.

All the synthetic nonstationary sinusoids generated are a combination of an amplitude envelope (A) and phase (P). The amplitude envelopes are constant (C), exponential (E), linear (L), cubic (C3), sinusoidal (S), and exponential-sinusoidal (ES). The phases are linear (L), quadratic (Q), cubic (C3), or sinusoidal (S). The synthetic nonstationary sinusoids are described next.

3.1. Constant Amplitude Linear Phase (CA-LP)

This is simply a stationary sinusoid used as reference. All the methods are expected to perform very well for stationary sinusoids.

$$s(t) = A\cos(\omega t + \theta), \tag{8}$$

where A is the constant amplitude, $\omega = 2\pi f_0$ is the constant frequency, and θ is the phase shift. The parameter values were $A = 1$, $f_0 = 100$, and $\theta = \frac{-\pi}{2}$.

3.2. Exponential Amplitude Linear Phase (EA-LP)

This corresponds to the underlying model from EDSM, thus we expect EDSM to perform very well for this particular case.

$$s(t) = \exp(\lambda + \mu t)\cos(\omega t + \theta), \tag{9}$$

where λ and μ are respectively the constant and damping factors. Thus $A(t)$ grows linearly when $\mu > 0$, decays when $\mu < 0$, and is constant if $\mu = 0$. For the phase, $\omega = 2\pi f_0$ is the constant frequency, and θ is the phase shift. The parameter values were $\lambda = 0$, $\mu = -50$, $f_0 = 100$, and $\theta = \frac{-\pi}{2}$.

3.3. Exponential Amplitude Quadratic Phase (EA-QP)

This is the underlying model from RSM, thus we expect RSM to fit this signal very well.

$$s(t) = \exp(\lambda + \mu t)\cos\left(\psi t^2 + \omega t + \theta\right), \tag{10}$$

where λ and μ are respectively the constant and damping factors, ψ, is the quadratic frequency $\omega = 2\pi f_0$ is the linear frequency, and θ is the phase shift. The parameter values were $\lambda = -0.5$, $\mu = -5$, $\psi = (2\pi)^2 f_1$ with $f_1 = 100$, $\omega = 2\pi f_0$ with $f_0 = 440$, and $\theta = \frac{-\pi}{2}$.

3.4. Constant Amplitude Cubic Phase (CA-C3P)

This is a particularly challenging signal because the cubic phase has a large range of variation. Depending on the location of the roots, the phase can vary slowly at first and suddenly grow very fast. We expect the C3P to be challenging for all models mainly because it does not match the underlying signal used by any.

$$s(t) = A\cos\left(\phi t^3 + \psi t^2 + \omega t + \theta\right), \tag{11}$$

where A is the constant amplitude, ϕ is the cubic phase, ψ, is the quadratic frequency $\omega = 2\pi f_0$ is the linear frequency, and θ is the phase shift. The parameter values were $A = 1$, $\phi = (2\pi)^3 f_2$ with $f_2 = 4,597.7$, $\psi = (2\pi)^2 f_1$ with $f_1 = 1,661.1$, $\omega = 2\pi f_0$ with $f_0 = 156.8$, and $\theta = \left(\frac{-\pi}{2}\right)^3$. The phase parameters were chosen to place the roots of C3P at respectively 100, 240, and 580 samples from a total of $N = 1600$ samples.

3.5. Exponential Amplitude Cubic Phase (EA-C3P)

This signal is more challenging than before because the C3 phase is modulated in amplitude by an exponential envelope. We expect this signal to be challenging for all models.

$$s(t) = \exp(\lambda + \mu t)\cos\left(\phi t^3 + \psi t^2 + \omega t + \theta\right). \tag{12}$$

The parameter values were $\lambda = 0$ and $\mu = -50$. The C3P parameters are the same as used previously in 3.5.

3.6. Linear Amplitude Cubic Phase (LA-C3P)

$$s(t) = (\lambda + \mu t)\cos\left(\phi t^3 + \psi t^2 + \omega t + \theta\right), \tag{13}$$

where λ is the vertical shift and μ is the slope of the amplitude envelope. The parameter values were $\lambda = 1$, $\mu = -10$. The C3P parameters are the same as in 3.5.

3.7. Cubic Amplitude Cubic Phase (C3A-C3P)

$$s(t) = \left(\lambda + \mu t + \gamma t^2 + \beta t^3\right)\cos\left(\phi t^3 + \psi t^2 + \omega t + \theta\right), \tag{14}$$

where the parameters λ, μ, γ, and β control the time-varying behavior of $A(t)$. The C3 amplitude envelope can be designed to vary considerably in short time frames by placing all the roots of the polynomial inside the frame. The amplitude parameter values were $\lambda = 0.059$, $\mu = -16.68$, $\gamma = -1110$, $\beta = 19305$. The C3A is simply the C3P signal normalized between 0 and 1. The C3P parameter values are the same as in 3.5.

3.8. Sinusoidal Amplitude Sinusoidal Phase (SA-SP)

This example contains both classic AM and FM modulations. We expect this signal to pose a challenge for all models.

$$s(t) = [A + B\cos(\omega_A t)]\cos(\omega_0 t + \theta_0 + \alpha\cos(\omega t)), \tag{15}$$

where A is the constant gain, B is the amplitude of the sinusoid, and ω_A is the constant frequency of the amplitude envelope. The sinusoidal amplitude envelope mimics the classic amplitude modulation signal and it arises in cases when there is *tremolo* or *beating* frequencies. The frequency parameter ω controls the rate of temporal variation inside the frame. The parameter values are $A = 0.7143$, $B = 0.2857$, $\omega_A = 2\pi f_A$ with $f_A = 50$, $\omega_0 = 2\pi f_0$ with $f_0 = 1000$, $\theta_0 = \frac{-\pi}{2}$, $\alpha = 1$, $\omega = 2\pi f_P$ with $f_P = 130$.

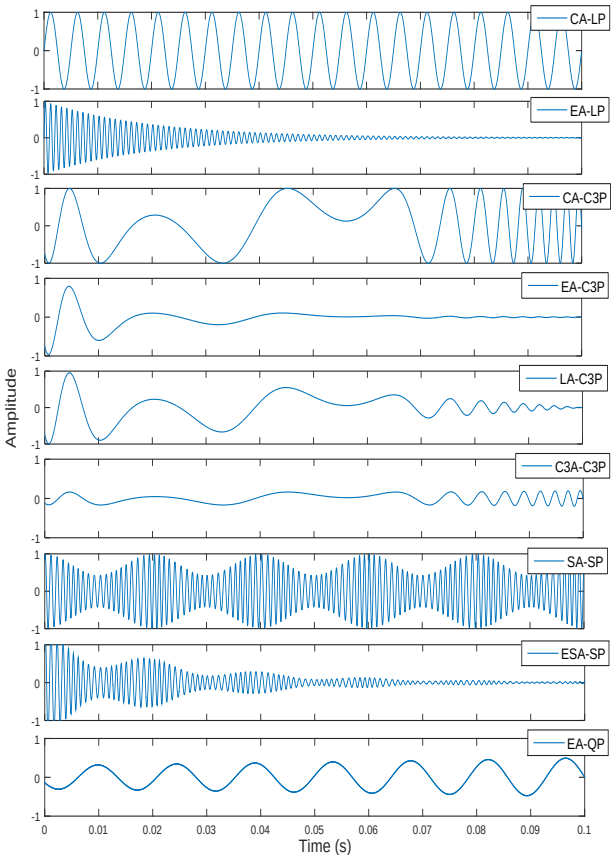


Figure 1: Illustration of the waveform for each synthetic nonstationary sinusoid from section 3.

3.9. Exponentially Damped Sinusoidal Amplitude Sinusoidal Phase (ESA-SP)

The ESA-SP sinusoid is $s(t) = A(t)\cos[\Phi(t)]$ with $A(t)$ and $\Phi(t)$ given below.

$$A(t) = \exp(\lambda + \mu t)\left[A + B\cos(\omega_A t)\right], \tag{16}$$

$$\Phi(t) = \omega_0 t + \theta_0 + \alpha\cos(\omega t), \tag{17}$$

where λ and μ are respectively the constant and damping factors and A is the constant gain, B is the amplitude of the sinusoid, and ω_A is the constant frequency. This amplitude envelope is simply the multiplication of the exponential (E) and the sinusoidal (S) envelopes, thus the result is a sinusoid modulated by the exponential. The parameter values are the same as in 3.8.

Figure 1 shows the synthetic nonstationary sinusoids described above to illustrate the resulting waveforms. Note that each signal presents very different characteristics, imposing different challenges for the models.

4. SYNTHETIC NONSTATIONARY SINUSOID MODELING ACCURACY

This section presents the experiment performed to compare the modeling accuracy of eaQHM, RSM, and EDSM for the nonstationary sinusoids described in section 3. We modeled each synthetic nonstationary sinusoid described earlier with eaQHM, RSM, and EDSM and measured the resulting modeling accuracy with the SRER as described next.

4.1. Measuring Modeling Accuracy

In what follows, we assume that the following relation holds

$$s(t) = y(t) + \hat{y}(t), \tag{18}$$

where $s(t)$ is the original synthetic signal, $y(t)$ is the model reconstruction after resynthesis, and $\hat{y}(t)$ is the modeling residual obtained by subtraction of $y(t)$ from $s(t)$ in the time domain. Then, the signal-to-reconstruction-error ratio (SRER) is defined as

$$\text{SRER} = 20\log_{10}\frac{\text{RMS}[s(t)]}{\text{RMS}[\hat{y}(t)]}\ \text{dB}. \tag{19}$$

Thus the SRER is the ratio in dB of the energy in the original synthetic signal $s(t)$ and the modeling residual $\hat{y}(t)$. Positive values indicate that $s(t)$ has more energy than $\hat{y}(t)$, while negative values indicate the opposite. Note that $\hat{y}(t)$ will only have low energy when the model $y(t)$ follows $s(t)$ very closely, which indicates good modeling performance. Consequently, higher SRER values indicate a better fit.

4.2. Analysis Parameters

All the synthetic nonstationary sinusoids were split into overlapping frames prior to analysis. The hop size was $H = 0.001F_s$ or 1 ms and the window size L varied between 10 ms and 70 ms as shown in Table 1.

EDSM uses a square window $w(t)$ for analysis and a Hamming window for overlap-add (OLA) resynthesis. RSM uses Hamming windows for both analysis and OLA resynthesis, while eaQHM uses a Hamming window for analysis and analytic resynthesis directly from (7).

4.3. Results

Table 1 shows the SRER value in dB for each synthetic signal from section 3 for each L indicated. RSM resulted in negative values for some values of L.

5. DISCUSSION

Table 1 shows that not all models performed as expected. While EDSM and eaQHM presented consistent performance for all the synthetic nonstationary sinusoids tested, RSM did not present robust performance. The SRER measure is very strict because it compares the waveforms directly. So small errors in only one parameter, such as the phase shift θ, for example, will lead to poor performance when measured with the SRER because the resulting waveform will be different. However, the instability in performance is likely due to the implementation used (the DESAM [50] toolbox) rather than the RSM method itself.

Table 1: SRER values in decibels (dB) for each synthetic signal when the window size L varies between 10 ms and 70 ms. C denotes *Constant*, E denotes *Exponential*, L denotes *Linear*, Q denotes *Quadratic*, C3 denotes *Cubic*, and S denotes *Sinusoidal* for either the amplitude envelope (A) or the phase (P). See section 3 for details.

		SRER (dB) for each window Size L(ms)						
sinusoid	algorithm	$L = 10$ ms	$L = 20$ ms	$L = 30$ ms	$L = 40$ ms	$L = 50$ ms	$L = 60$ ms	$L = 70$ ms
CA-LP	eaQHM	286.7	286.6	286.5	286.5	286.4	286.4	286.4
	RSM	−6.0	28.2	−6.0	25.1	−6.0	23.4	−6.0
	EDSM	282.1	278.1	274.0	272.8	264.6	263.4	268.7
EA-LP	eaQHM	66.6	53.4	46.6	42.7	40.0	38.7	34.9
	RSM	−6.0	−6.0	−6.0	−6.0	−6.0	−6.0	−6.0
	EDSM	280.3	269.8	267.7	265.7	262.2	271.0	270.2
EA-QP	eaQHM	51.5	10.0	3.4	2.1	1.5	0.6	0.7
	RSM	−2.3	−1.6	−0.7	−0.8	−0.5	−7.7	−0.1
	EDSM	41.1	6.7	3.5	2.3	1.9	1.4	1.2
CA-C3P	eaQHM	65.4	49.3	18.1	11.7	6.6	4.1	2.7
	RSM	−6.1	−5.2	−3.1	−5.4	−4.3	−21.5	−15.9
	EDSM	46.1	24.4	12.8	7.7	5.7	4.8	4.2
EA-C3P	eaQHM	57.5	45.7	5.2	2.7	2.4	2.0	1.5
	RSM	−3.6	−2.6	−1.1	−2.3	−1.4	−14.7	−1.8
	EDSM	49.3	24.9	11.1	5.0	3.6	3.4	3.1
LA-C3P	eaQHM	69.8	52.2	2.9	1.1	0.8	0.6	0.4
	RSM	−0.5	−0.3	−0.3	−0.3	−3.2	−3.9	−0.2
	EDSM	52.5	15.5	5.4	3.1	2.6	1.8	1.6
C3A-C3P	eaQHM	29.8	7.6	4.1	3.7	3.4	3.4	3.4
	RSM	2.6	3.7	3.1	3.2	3.2	3.2	3.2
	EDSM	11.2	4.4	3.8	2.9	3.6	3.5	2.5
SA-SP	eaQHM	24.2	7.2	4.1	4.0	3.6	3.3	3.6
	RSM	3.0	3.4	3.2	3.7	3.6	3.6	3.7
	EDSM	12.0	5.7	5.5	4.7	4.9	4.8	4.8
ESA-SP	eaQHM	107.4	33.4	33.6	13.1	15.4	17.5	17.5
	RSM	−6.0	−6.0	24.6	−6.0	−0.1	20.6	−6.0
	EDSM	165.9	139.8	123.6	112.0	102.9	95.5	89.1

Nonstationary sinusoids with time-varying frequency are more challenging to model with longer windows. The modeling performance of eaQHM and EDSM decreased when L increased for all the synthetic nonstationary sinusoids except when the phase was linear, namely CA-LP and EA-LP.

Both eaQHM and EDSM present very high SRER for stationary sinusoids (CA-LP). As expected, EDSM outperformed eaQHM for its underlying sinusoid (EA-LP). EDSM also outperformed eaQHM for exponentially damped sinusoidal amplitude modulation and sinusoidal phase (ESA-SP). RSM performed poorly for its underlying sinusoid (EA-QP), presenting negative values throughout.

In general, eaQHM presented the best performance of all models for most signals tested, indicating that adaptation is able to represent well even signals that are different from its underlying model. EDSM also presents better performance than RSM possibly due to the use of ESPRIT to estimate the parameter values.

6. CONCLUSIONS AND PERSPECTIVES

In this paper, we propose to model non-stationary oscillations with adaptive sinusoids from the extended adaptive quasi-harmonic model (eaQHM). We generated synthetic non-stationary sinusoids with different amplitude and frequency modulations and compared the modeling performance of adaptive sinusoids estimated with eaQHM, exponentially damped sinusoids (EDS) estimated with ESPRIT, and log-linear-amplitude quadratic-phase sinusoids estimated with time-frequency reassignment (RSM). Modeling performance is measured with the signal-to-reconstruction-error ratio (SRER), which uses the waveforms directly. The adaptive sinusoids from eaQHM outperformed RSM for all the signals tested and presented performance comparable to EDSM.

Future work should focus on applying eaQHM to modeling recordings of expressive speech and music performance. In previous works, eaQHM has been shown to perform well when modeling relatively stable speech utterances and musical instrument sounds. However, modeling the modulations from expressive speech and music performance would be challenging. Presently, eaQHM only handles monophonic sounds. Therefore, it would be also very interesting to investigate parameter estimation strategies for polyphonic music.

7. REFERENCES

[1] N. H. Fletcher, "The nonlinear physics of musical instruments," *Reports on Progress in Physics*, vol. 62, pp. 723–764, 1999.

[2] Rolf Bader and Uwe Hansen, "Modeling of musical instruments," in *Handbook of Signal Processing in Acoustics*, David Havelock, Sonoko Kuwano, and Michael Vorländer, Eds., pp. 419–446. Springer New York, 2008.

[3] P. Herrera and J Bonada, "Vibrato extraction and parameterization in the spectral modeling synthesis framework," in *Vibrato Extraction and Parameterization in the Spectral Modeling Synthesis framework*, 1998, pp. 107–110.

[4] Henrik Von Coler and Axel Roebel, "Vibrato Detection Using Cross Correlation Between Temporal Energy and Fundamental Frequency," in *131st AES Convention*, 2011.

[5] Nicolas Obin, Christophe Veaux, and Pierre Lanchantin, "Exploiting alternatives for text-to-speech synthesis: From machine to human," in *Speech Prosody in Speech Synthesis: Modeling and generation of prosody for high quality and flexible speech synthesis*, Keikichi Hirose and Jianhua Tao, Eds., Prosody, Phonology and Phonetics, pp. 189–202. Springer Berlin Heidelberg, 2015.

[6] G. P. Kafentzis, O. Rosec, and Y. Stylianou, "On the modeling of voiceless stop sounds of speech using adaptive quasi-harmonic models," in *Interspeech*, 2012.

[7] L. Regnier and G. Peeters, "Singing voice detection in music tracks using direct voice vibrato detection," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2009, pp. 1685–1688.

[8] E. B. George and M. Smith, "A new Speech Coding Model based on a Least-Squares Sinusoidal Representation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr 1987, pp. 1641–1644.

[9] R. J. Mcaulay and T. F. Quatieri, "Low-rate speech coding based on the sinusoidal model," in *Advances in Speech Signal Processing*, S. Furui and M. M. Sondhi, Eds. Marcel Dekker Inc., New York, 1992.

[10] S. Ahmadi and A. S. Spanias, "Low bit-rate speech coding based on an improved sinusoidal model," *Speech Communication*, vol. 34, no. 4, pp. 369 – 390, 2001.

[11] R. J. McAulay and T. F. Quatieri, "Speech Analysis/Synthesis based on a Sinusoidal Representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, pp. 744–754, 1986.

[12] X. Serra, *A System for Sound Analysis, Transformation, and Synthesis based on a Deterministic plus Stochastic Decomposition*, Ph.D. thesis, Stanford University, 1989.

[13] T.F. Quatieri and R.J. McAuley, "Audio signal processing based on sinusoidal analysis/synthesis," in *Applications of Digital Signal Processing to Audio and Acoustics*, Mark Kahrs and Karlheinz Brandenburg, Eds., chapter 9, pp. 343–416. Kluwer Academic Publishers, 2002.

[14] J. Laroche, Y. Stylianou, and E. Moulines, "HNM: A Simple, Effecient Harmonic plus Noise Model for Speech," in *Workshop on Appl. of Signal Proc. to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, Oct 1993, pp. 169–172.

[15] E. B. George and M. J. T. Smith, "Speech analysis/synthesis and modification using an analysis-by-synthesis/overlap-add sinusoidal model," *IEEE Transactions on Speech and Audio Processing*, vol. 5, no. 5, pp. 389–406, Sep 1997.

[16] Y. Pantazis, G. Tzedakis, O. Rosec, and Y. Stylianou, "Analysis/Synthesis of Speech based on an Adaptive Quasi-Harmonic plus Noise Model," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2010, pp. 4246–4249.

[17] M. Caetano, G. P. Kafentzis, A. Mouchtaris, and Y. Stylianou, "Adaptive sinusoidal modeling of percussive musical instrument sounds," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2013, pp. 1–5.

[18] Michael E. Deisher and Andreas S. Spanias, "Speech enhancement using state-based estimation and sinusoidal modeling," *The Journal of the Acoustical Society of America*, vol. 102, no. 2, pp. 1141–1148, 1997.

[19] J. Jensen and J.H.L. Hansen, "Speech enhancement using a constrained iterative sinusoidal model," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 7, pp. 731–740, 2001.

[20] E. Zavarehei, S. Vaseghi, and Qin Yan, "Noisy speech enhancement using harmonic-noise model and codebook-based post-processing," *IEEE Trans. on Audio, Speech, and Lang. Processing*, vol. 15, no. 4, pp. 1194–1203, 2007.

[21] Y. Stark and J. Tabrikian, "MMSE-based speech enhancement using the harmonic model," in *Electrical and Electronics Engineers in Israel, 2008. IEEEI 2008. IEEE 25th Convention of*, 2008, pp. 626–630.

[22] T.F. Quatieri and R.J. McAulay, "Shape-Invariant Time-Scale and Pitch Modifications of Speech," *IEEE Trans. on Acoust., Speech and Signal Processing*, vol. 40, pp. 497–510, 1992.

[23] Y. Stylianou, J. Laroche, and E. Moulines, "High-Quality Speech Modification based on a Harmonic + Noise Model," *Proc. European Conference on Speech Communication and Technology (EUROSPEECH)*, 1995.

[24] Naotoshi Osaka, "Timbre interpolation of sounds using a sinusoidal model," in *Proceedings of the International Computer Music Conference*, 1995.

[25] Riccardo Di Federico, "Waveform preserving time stretching and pitch shifting for sinusoidal models of sound," in *Proceedings of the COST-G6 Digital Audio Effects Workshop*, 1998, pp. 44–48.

[26] G. P. Kafentzis, G. Degottex, O. Rosec, and Y. Stylianou, "Time-scale Modifications based on an Adaptive Harmonic Model," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 8193–8197.

[27] M. Goodwin, "Matching pursuit with damped sinusoids," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1997, pp. 2037–2040.

[28] T. S. Verma and T. H. Y. Meng, "Sinusoidal modeling using frame-based perceptually weighted matching pursuits," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1999, pp. 981–984.

[29] R. Heusdeuns, R. Vafin, and W. B. Kleijn, "Sinusoidal modeling using psychoacoustic-adaptive matching pursuits," *IEEE Signal Processing Letters*, vol. 9, no. 8, 2002.

[30] Y. Stylianou, *Harmonic plus Noise Models for Speech, combined with Statistical Methods, for Speech and Speaker Modification*, Ph.D. thesis, E.N.S.T - Paris, 1996.

[31] Y. Pantazis, O. Rosec, and Y. Stylianou, "Adaptive AM-FM signal decomposition with application to speech analysis," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, pp. 290–300, 2011.

[32] G. Degottex and Y. Stylianou, "Analysis and synthesis of speech using an adaptive full-band harmonic model," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2085–2095, 2013.

[33] J. Nieuwenhuijse, R. Heusdens, and E. Deprettere, "Robust exponential modeling of audio signals," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1998, pp. 3581–3584.

[34] J. Jensen, R. Heusdens, , and S. Jensen, "A perceptual subspace approach for modeling of speech and audio signals with damped sinusoids," *IEEE Transaction on Speech and Audio Processing*, vol. 12, no. 2, pp. 121–132, March 2004.

[35] R. Badeau, B. David, and G. Richard, "A new perturbation analysis for signal enumeration in rotational invariance techniques," *IEEE Transaction on Signal Processing*, vol. 54, no. 2, pp. 492–504, February 2006.

[36] G. Zhou, G.B. Giannakis, and A. Swami, "On polynomial phase signals with time-varying amplitudes," *Signal Processing, IEEE Transactions on*, vol. 44, no. 4, pp. 848–861, 1996.

[37] Yinong Ding and Xiaoshu Qian, "Processing of musical tones using a combined quadratic polynomial-phase sinusoid and residual (quasar) signal model," *Journal of the Audio Engineering Society*, vol. 45, no. 7/8, pp. 571–584, 1997.

[38] L. Girin, S. Marchand, Joseph Di Martino, A. Robel, and G. Peeters, "Comparing the order of a polynomial phase model for the synthesis of quasi-harmonic audio signals," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2003, pp. 193–196.

[39] Sean A. Fulop and Kelly R. Fitz, "Algorithms for computing the time-corrected instantaneous frequency (reassigned) spectrogram, with applications," *Journal of the Acoustical Society of America*, vol. 119, no. 1, pp. 360–371, 2006.

[40] Jeremy J. Wells and Damian T. Murphy, "High-accuracy frame-by-frame non-stationary sinusoidal modelling," in *Proc. International Conference on Digital Audio Effects (DAFx)*, 2006, pp. 253–258.

[41] Saso Musevic and Jordi Bonada, "Generalized reassignment with an adaptive polynomial phase fourier kernel for the estimation of nonstationary sinusoidal parameters," in *Proc. International Conference on Digital Audio Effects (DAFx)*, 2011.

[42] Sylvain Marchand, "The Simplest Analysis Method for Non-stationary Sinusoidal Modeling," in *Proc. International Conference on Digital Audio Effects (DAFx)*, 2012, pp. 23–26.

[43] Sylvain Marchand and Philippe Depalle, "Generalization of the derivative analysis method to non-stationary sinusoidal modeling," in *Proc. International Conference on Digital Audio Effects (DAFx)*, 2008.

[44] B. Hamilton and P. Depalle, "A unified view of non-stationary sinusoidal parameter estimation methods using signal derivatives," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 369–372.

[45] S. Musevic and J Bonada, "Comparison of non-stationary sinusoid estimation methods using reassignment and derivatives," in *Sound and Music Computing Conference*, 2010.

[46] G. P. Kafentzis, Y. Pantazis, O. Rosec, and Y. Stylianou, "An extension of the adaptive quasi-harmonic model," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 4605–4608.

[47] M. Caetano, G. Kafentzis, G. Degottex, A. Mouchtaris, and Y. Stylianou, "Evaluating how well filtered white noise models the residual from sinusoidal modeling of musical instrument sounds," in *Proc. Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2013.

[48] C. Borss and R. Martin, "On the construction of window functions with constant-overlap-add constraint for arbitrary window shifts," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 337–340.

[49] K. Hermus, W. Verhelst, P. Lemmerling, P. Wambacq, and S. van Huffel, "Perceptual audio modeling with exponentially damped sinusoids," *Signal Processing*, vol. 85, no. 1, pp. 163–176, January 2005.

[50] Mathieu Lagrange, Roland Badeau, Bertrand David, Nancy Bertin, Jose Echeveste, Olivier Derrien, Sylvain Marchand, and Laurent Daudet, "The DESAM toolbox: spectral analysis of musical audio," in *Proc. International Conference on Digital Audio Effects (DAFx)*, 2010, pp. 254–261.

DISTRIBUTION DERIVATIVE METHOD FOR GENERALISED SINUSOID WITH COMPLEX AMPLITUDE MODULATION

Sašo Muševič
Music Technology Group,
Universitat Pompeu Fabra
Barcelona, Spain
musevic@gmail.com

Jordi Bonada
Music Technology Group,
Universitat Pompeu Fabra
Barcelona, Spain
jordi.bonada@upf.edu

ABSTRACT

The most common sinusoidal models for non-stationary analysis represent either complex amplitude modulated exponentials with exponential damping (cPACED) or log-amplitude/frequency modulated exponentials (generalised sinusoids), by far the most commonly used modulation function being polynomials for both signal families. Attempts to tackle a *hybrid* sinusoidal model, i.e. a generalised sinusoid with complex amplitude modulation were relying on approximations and iterative improvement due to absence of a tractable analytical expression for their Fourier Transform. In this work a simple, direct solution for the aforementioned model is presented.

1. INTRODUCTION

Sinusoidal analysis algorithms' vast area of application, ranging from sound and music analysis [1, 2], medical data analysis [3] and imaging [4], Doppler radar [5] and sonar applications, seismic signal analysis, hydrogen atom spectrum analysis [6], laser technology [7] and financial data analysis [8] had fuelled the research field for decades. Rapidly modulated sinusoids found in most of the aforementioned applications have sparked the interest in non-stationary sinusoidal analysis.

Recent development in this area, have provided efficient and accurate methods for either cPACED or generalised sinusoid model. A somehow hybrid model was attempted with real polynomial amplitude amplitude and frequency modulation, however to authors' knowledge only approximate, iterative-improvement type algorithm were developed to date [9, 10].

A very old idea of TF energy *reassignment* [11] has been a focus of much research lately [12, 13, 14, 15, 16]. The vast variety of modifications of the original reassignment, be it merely a generalisation for higher modulations [17] or redefined for an entirely different model [18], has called for a more general name for this family of algorithms. Recently a *reallocation* [19] of TF energy was proposed and will be used in this work.

The paper is organised as follows: in 2 the hybrid signal model is outlined, while section 3 derives the non-linear multivariate polynomial system and its solution, inspired by the distribution derivative method. Section 4 compares the accuracy and computational complexity of the proposed algorithm to the high-resolution method based on rotational invariance.

2. HYBRID SIGNAL MODEL

The hybrid sinusoidal model will be defined as follows:

$$s(t) = a(t)e^{r(t)}, \tag{1}$$

$$a(t) = \sum_{k=0} a_k m_k(t), a_k \in \mathbb{C} \tag{2}$$

$$r(t) = \sum_{l=0} r_l n_l(t), r_l \in \mathbb{C} \tag{3}$$

where m_k, n_k are the complex amplitude and log-AM/FM model functions respectively. To accommodate for the static amplitude and phase, $m_0 = n_0 = 1$ is assumed. A most common, but by no means mandatory selection for the model functions are polynomials: $m_k = n_k = t^k$.

The above model 1 is ambiguous with respect to parameters r_0 and a_0. To show this, the following derivation is considered:

$$s(t) = a(t)e^{r(t)} = a_0\tilde{a}(t)e^{r_0+\tilde{r}(t)} \tag{4}$$

$$= \tilde{a}(t)\exp(\overbrace{\log(|a_0|) + j\angle(a_0) + r_0}^{\tilde{r}_0} + \tilde{r}(t)), \tag{5}$$

Clearly a_0 and r_0 are in fact the same parameter in either Cartesian or polar coordinates. The decision seems irrelevant, however as will be shown in section 3, using the Cartesian form would result in a rank-deficient system, therefore the model will be constrained to $a_0 = 1$.

It is important to note that since modulation functions are complex they both contribute to overall AM/FM. If the same model functions are used ($m_k = n_k$) that can lead to some ambiguity, especially when the energy of m_k, n_k declines fast with k. Such ambiguity can be demonstrated when using polynomials for the modulation functions $m_k = n_k = t^k$:

$$a(t)e^{r(t)} = \exp(\log(a(t)) + r(t)) \tag{6}$$

$$\approx \exp(a_1 t + (2a_2 - {a_1}^2)t^2 + r(t)), \tag{7}$$

using the 2^{nd} degree truncated Taylor expansion. It is expected that an estimator for the model in 1 could be inaccurate when separate parameter estimates are considered, but generally much more flexible due to twin AM/FM functions. In practice however one is mostly concerned with algorithm's overall ability to fit to the signal under investigation, rather then individual per-parameter accuracy. It would be feasible to devise a disambiguation procedure but this is considered to be outside the scope of this document.

An example is shown in figure 1, where 2 hybrid model sinusoids with significantly different a_1, r_1 reach a signal-to-residual ratio (SRR) of $24dB$. To reach higher SRR the parameters would

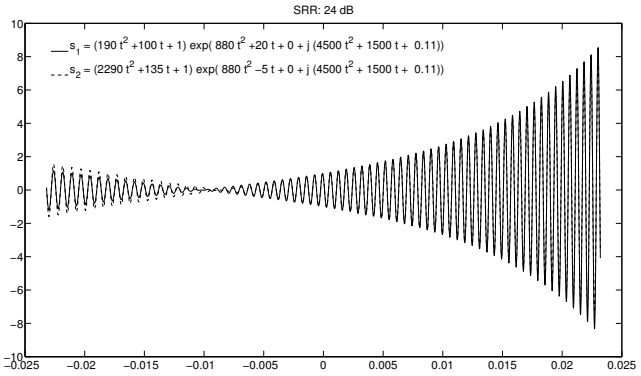


Figure 1: Two sinusoids with significantly different parameters obtain a similar SRR of 24dB.

have to eventually match exactly, however in noisy conditions a relatively high SRR is achievable with substantial error in per-parameter estimates. From figure 1 is also evident that the proposed model includes sinusoids with *negative* amplitude, suggesting good coding abilities for sinusoid pairs with close frequencies (i.e.: beating partials). The negative amplitude can occur when $\Im[a(t)] = 0$, since the overall amplitude corresponds to: $\sqrt{a(t)\bar{a}(t)}e^{\Re[r(t)]}$. The notion of negative amplitude is purely artificial ($\sqrt{a(t)\bar{a}(t)}e^{\Re[r(t)]}$ cannot be negative), as it does not have a natural physical meaning, however it comes handy as a mathematical generalisation. If negative amplitude is not allowed and $\Im[a(t)] = 0$, the derivative of $\sqrt{a(t)\bar{a}(t)}e^{\Re[r(t)]} = |a(t)|e^{\Re[r(t)]}$ is not continuous for all roots of $a(t)$ and thus the model function could not be considered a holomorphic function. In such cases the absolute value can be easily dropped and negative amplitude introduced, leading to a mathematically sound model in the context of holomorphic functions.

3. NON-LINEAR MULTIVARIATE POLYNOMIAL SYSTEM OF EQUATIONS

The non-linear system can be derived by considering the signal time derivative manipulated in the following way:

$$s'(t) = a'(t)e^{r(t)} + r'(t)a(t)e^{r(t)}, \Rightarrow \tag{8}$$

$$a(t)s'(t) = a'(t)s(t) + a(t)r'(t)s(t). \tag{9}$$

The last row can be rewritten in a more verbose form that reveals the non-linearity of the system:

$$m_0 s' + \sum_{k=1}^{K} a_k m_k s' = \sum_{k=1}^{K} a_k m'_k s + \left(m_0 + \sum_{k=1}^{K} a_k m_k\right) \sum_{l=1}^{L} r_l n'_l s, \tag{10}$$

where time variable t was omitted for compactness. The only non-linear terms arise from the last - double sum expression. Multiplying both sides with a window function $w(t)$ and taking a Fourier Transform (FT) at frequency ω yields:

$$S'_{wm_0}(\omega) + \sum_{k=1}^{K} a_k S'_{wm_k}(\omega) = \sum_{k=1}^{K} a_k S_{wm'_k}(\omega) + \sum_{l=1}^{L} r_l S_{wm_0 n'_l}(\omega) + \sum_{k=1}^{K}\sum_{l=1}^{L} a_k r_l S_{wm_k n'_l}(\omega), \tag{11}$$

where $S_f(\omega) = \langle s(t)f(t), e^{j\omega t}\rangle$ is the FT of the signal s multiplied by function f, and $S'_g(\omega) = \langle s'(t)g(t), e^{j\omega t}\rangle$ is the FT of the signal derivative multiplied by function g at frequency ω. Note that $n'_0 = m'_0 = 0$ and thus the sums on the right-hand side start at index 1 rather than 0. Above equation can be viewed as a (non-linear) multivariate polynomial with respect to parameters $a_k : k = 1 \dots K, r_l : l = 1 \dots L$. The expressions S'_f, S_g can be considered constants for any f, g as they can be directly computed from the signal. To calculate S'_f accurately, sample difference in time domain should be avoided [12]. A common approach is the use of distribution derivative rule $\langle x', y\rangle = -\langle x, y'\rangle$ and a real window function w as a part of the kernel y:

$$S'_{gw}(\omega) = \langle s'g, w\psi_\omega\rangle = \langle s', \bar{g}w\psi_\omega\rangle \tag{12}$$

$$= -\langle s, \bar{g}'w\psi_\omega\rangle - \langle s, \bar{g}w'\psi_\omega\rangle + j\omega\langle s, \bar{g}w\psi_\omega\rangle \tag{13}$$

$$= -\langle sg', w\psi_\omega\rangle - \langle sg, w'\psi_\omega\rangle + j\omega\langle sg, w\psi_\omega\rangle \tag{14}$$

$$= -S_{g'w}(\omega) - S_{gw'}(\omega) + j\omega S_{gw}(\omega), \tag{15}$$

where ψ_ω is generally a kernel function with FT centred around frequency ω. For the last equality to hold the kernel is set simply to the Fourier kernel: $\psi_\omega(t) = e^{j\omega t}$. Higher time derivatives can accurately be computed by chaining the above expression. Rearranging the equation and collecting together the model parameters yields:

$$S'_{wm_0}(\omega) = \sum_{k=1}^{K} a_k (S_{wm'_k}(\omega) - S'_{wm_k}(\omega)) + \sum_{l=1}^{L} r_l S_{wm_0 n'_l}(\omega) + \sum_{k=1}^{K}\sum_{l=1}^{L} a_k r_l S_{wm_k n'_l}(\omega). \tag{16}$$

Taking the FT at different frequencies close to the peak provides as many equations as necessary. Assuming polynomial modulation functions, the following system can be derived:

$$S'_w(\omega) + \sum_{k=1}^{K} a_k S'_{t^k w}(\omega) = \sum_{k=1}^{K-1} k a_k S_{t^{k-1} w}(\omega) + \sum_{k=1}^{K} a_K \sum_{l=0}^{L-1} (l+1) r_{l+1} S_{t^{k+l}}(\omega) + \sum_{l=0}^{L-1} (l+1) r_{l+1} S_{t^l}(\omega) \tag{17}$$

For a cPACED sinusoids with polynomial amplitude of degree 3 (ie: K=3, L=1) the following case can be deduced:

$$S'_w(\omega) + a_1 S'_{tw}(\omega) + a_2 S'_{t^2 w}(\omega) + a_3 S'_{t^3 w}(\omega) = a_1 S_w(\omega) + 2a_2 S_{tw}(\omega) + 3a_3 S_{t^2 w}(\omega) + a_1 r_1 S_{tw}(\omega) + a_2 r_1 S_{t^2 w}(\omega) + a_3 r_1 S_{t^3 w}(\omega) + r_1 S_w(\omega) \tag{18}$$

Grouping the linear and non-linear terms in respect to a_l, r_1:

$$S'_w(\omega) = a_1(S_w(\omega) - S'_{tw}(\omega)) + a_2(2S_{tw}(\omega) - S'_{t^2w}(\omega)) + a_3(3S_{t^2w}(\omega) - S'_{t^3w}(\omega)) + r_1 S_w(\omega) + a_1 r_1 S_{tw}(\omega) + a_2 r_1 S_{t^2w}(\omega) + a_3 r_1 S_{t^3w}(\omega). \quad (19)$$

The distribution derivative rule can be applied to the S' terms:

$$S'_{t^k w}(\omega) = -kS_{t^{k-1}w}(\omega) - S_{t^k w'}(\omega) + j\omega S_{t^k w}(\omega), \quad \text{for } k > 0 \quad (20)$$

$$S'_w(\omega) = -S_{w'}(\omega) + j\omega S_w(\omega), \quad \text{for } k = 0 \quad (21)$$

Following the approach of the distribution derivative method and considering the FT at 4 different frequencies, we obtain the following non-linear multivariate system:

$$Ax = b \quad (22)$$

$$A = \begin{pmatrix} S_w(\omega_1) - S'_{tw}(\omega_1) & \cdots & S_w(\omega_4) - S'_{tw}(\omega_4) \\ 2S_{tw}(\omega_1) - S'_{t^2w}(\omega_1) & \cdots & 2S_{tw}(\omega_4) - S'_{t^2w}(\omega_4) \\ 3S_{t^2w}(\omega_1) - S'_{t^3w}(\omega_1) & \cdots & 3S_{t^2w}(\omega_4) - S'_{t^3w}(\omega_4) \\ S_w(\omega_1) & \cdots & S_w(\omega_4) \\ S_{tw}(\omega_1) & \cdots & S_{tw}(\omega_4) \\ S_{t^2w}(\omega_1) & \cdots & S_{t^2w}(\omega_4) \\ S_{t^3w}(\omega_1) & \cdots & S_{t^3w}(\omega_4) \end{pmatrix}', \quad (23)$$

$$x = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ r_1 \\ r_1 a_1 \\ r_1 a_2 \\ r_1 a_3 \end{pmatrix}, b = \begin{pmatrix} S'_w(\omega_1) \\ S'_w(\omega_2) \\ S'_w(\omega_3) \\ S'_w(\omega_4) \end{pmatrix}.$$

Note that for high parameter values, the frequency spread of the signal might be large - a small number of frequency bins (in the above case 4) might not suffice to cover enough information in the Fourier domain. In such cases more frequency bins can be considered.

4. TESTS AND RESULTS

The proposed method was implemented in a Matlab script for an arbitrary degree of amplitude and exponential complex polynomials. The resulting multivariate non-linear systems were solved by the Matlab function $fsolve$. This function only accepts real variables and coefficients as parameters, although internally can use complex variables and solve the system if the complex equations are split into real and imaginary parts.

A polynomial amplitude of degree 3 was studied and the polynomial denoted as: $[a_3, a_2, a_1, 1] = [p_3 + jq_3, p_2 + jq_2, p_1 + jq_1, 1]$. The test values for p_3, p_2, p_1 were chosen so all the terms of the amplitude polynomial have equal impact on the final value:

$$p_3 \in \left[-\left(\frac{fs}{8T}\right)^3, \left(\frac{fs}{8T}\right)^3 \right] \quad (24)$$

$$p_2 \in \left[-\left(\frac{fs}{8T}\right)^2, \left(\frac{fs}{8T}\right)^2 \right] \quad (25)$$

$$p_1 \in \left[-\frac{fs}{8T}, \frac{fs}{8T} \right]. \quad (26)$$

The exact same value sets were used for the imaginary part of the polynomial $q(t)$. A Hann window function of length 511 samples was used for pole estimation and Hann window for the complex polynomial coefficients estimation. The damping factor was set to [-150,0,150] and only one frequency of 10000Hz was considered to match tests performed in [12]. r_0 was set to 0, since gain has theoretically no effect when the snr is fixed. For p_1, p_2, p_3, q_1, q_2 and q_3 parameters, only 5 linearly distributed values have been tested in order to keep the computational time reasonable. The comparison to a 3^{th} degree (i.e. 4 poles and amplitudes) simple high-resolution method (HRM) implementation from DESAM Toolbox [20] (section 5.1.2.) without whitening and the cPACED reassignment method (cPACED-RM) [18] was conducted. The signal tested is the real part of the complex cPACED signal, reflecting the real world scenario when analytical signal isn't available.

To measure accuracy, the commonly used Signal-to-Residual-Ratio (SRR) metric was used,

$$SRR = \frac{\langle s, ws \rangle}{\langle s - \hat{s}, w(s - \hat{s}) \rangle}, \quad (27)$$

where $s, \hat{s}$ are the original signal (without noise) and the estimated signal respectively, and w is the Hann window. The Signal-to-Noise-Ratio (SNR) range from -20 to 50dB with steps of 10dB was studied. The total computation times for both methods follow:

cPACED-RM	7 min
cPACED-DDM	300 min
High-resolution	880 min

(28)

Since HRM involves singular value decomposition of correlation matrix of size $N/2 \times N/2$ the computation cost is significantly the highest among the tested methods. cPACED-RM method requires $K - 1$ FFTs for the pole and K DTFTs for the complex polynomial estimates to build a linear system. In contrast, the proposed method requires only K DTFTs to build a non-linear multivariate system. However, solving such system by iterative methods requires a significant computation cost. cPACED-RM method performs much faster since it requires solving a linear system.

The classic Cramer-Rao bounds (CRBs) parameter-by-parameter comparison would total to 10 plots, overcomplicating the results and obscuring the overall accuracy. A more intuitive approach involves only one SRR/SNR plot, although a different upper accuracy bound is required. For each test case the CRBs for each parameter were computed. Denoting a CRB for parameter a_0 as ϵ_{a_0}, the minimum SRR for the specific CRB set can be defined:

$$\min SRR(\hat{s}(a_3 \pm \epsilon_{a_3}, a_2 \pm \epsilon_{a_2}, a_1 \pm \epsilon_{a_1}, r_0 \pm \epsilon_{r_0}, r_1 \pm \epsilon_{r_1}). \quad (29)$$

The mean and variance of the minimum SRR represents a good upper SRR bound. Figure 2 depicts the mean and variance of the upper SRR bound, the proposed method (cPACED-DDM), cPACED-RM and HRM. At low SNR, cPACED-DDM and cPACED-RM

perform roughly the same, up to ˜5dB below the upper bound, while cPACED-RM performs ˜3dB better. For mid-high SNRs, all methods perform roughly the same, about ˜5dB below the upper bound. In general cPACED-DDM performs ˜1dB below cPACED-RM.

The main advantage of the proposed algorithm is that it offers the flexibility to analyze generalised sinusoids with complex amplitude of any polynomial degree combination. With the purpose of having an initial exploration of its general performance, we have computed the mean SRR obtained for several combinations of amplitude and exponential complex polynomial degrees. In particular, amplitude polynomial degrees were set to [0,1,2,3] and exponential polynomial degrees to [1,2,3]. Table 1 shows the results obtained. In this experiment we used the same polynomial coefficient value sets as for the previous cPaced case. For each SNR, the results show a general tendency to decrease the SRR as we increase the complexity of the signal by increasing the degrees of the polynomials. Figure 3 compares the mean and variance of the SRR obtained for all degree combinations with the SNR. It shows a general trend of reaching an SRR ˜24dB above the SNR value, although the difference decreases significantly to ˜13dB for SNR=-20dB, and to ˜14dB for SNR=50dB. This seems to indicate that the proposed method reaches a plateau for high SNRs above 50dB.

5. DISCUSSION AND FUTURE WORK

In this work the currently most flexible sinusoidal method for TF energy reassignment analysis has been described. The concept used in the distribution derivative method is used to generate a nonlinear multivariate system of polynomials obtained by the first signal derivative. It is important to note that higher signal derivatives would provide enough equations for a solution to exist, however a significantly more complex system would be obtained. Even if solution could eventually be obtained, it is desirable to avoid higher signal derivatives due to ill conditioning.

The method showed a similar performance than the high resolution and the reassignment methods for the cPACED signal model, however in theory the proposed method is much more flexible since it can be applied to generalised sinusoids with complex amplitude of any polynomial degree combination. The initial exploration of the performance obtained for several degree combinations was promising, although a more in depth evaluation was left for future work. Further, the proposed sinusoidal model seems promising for the analysis of overlapping partials, as the beating function corresponds to real value amplitude/frequency modulated sinusoids - a subfamily of signals described by the proposed model.

On the other hand high-resolution methods' intrinsic frequency resolution of 1 frequency bin [21, 22] for damped sinusoids has not been surpassed, as common window function mainlobe width (several bins) and significant sidelobe amplitude both reduce the frequency resolution.

6. REFERENCES

[1] J. Bonada, O. Celma, A. Loscos, J. Ortolà, and X. Serra, "Singing voice synthesis combining excitation plus resonance and sinusoidal plus residual models," La Habana, Cuba, Sep. 2001. [Online]. Available: http://quod.lib.umich.edu/cgi/p/pod/dod-idx?c=icmc;idno=bbp2372.2001.093

[2] X. Serra, "A system for sound Analysis/Transformation/Synthesis based on a deterministic plus stochastic decomposition," Ph.D. dissertation, Stanford University, Stanford, CA, 1989. [Online]. Available: https://ccrma.stanford.edu/files/papers/stanm58.pdf

[3] S. M. I. Daubechies, *Wavelets in Medicine and Biology*. CRC Press, 1996, ch. A nonlinear squeezing of the continuous wavelet transform based on auditory nerve models, pp. 527–546.

[4] M. Pattichis, C. Pattichis, M. Avraam, A. Bovik, and K. Kyriacou, "Am-fm texture segmentation in electron microscopic muscle imaging," *Medical Imaging, IEEE Transactions on*, vol. 19, no. 12, pp. 1253–1257, 2000.

[5] A. Sabatini, "Correlation receivers using laguerre filter banks for modelling narrowband ultrasonic echoes and estimating their time-of-flights," *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, vol. 44, no. 6, pp. 1253–1263, 1997.

[6] T. W. Hänsch, A. L. Schawlow, and G. W. Series, "The spectrum of atomic hydrogen," *Sci. Amer.*, vol. 240, no. 94, pp. 531–544, Mar. 1979.

[7] P. W. Milonni and J. H. Eberly, *Lasers*, ser. Wiley Series in Pure and Applied Optics. New York: Wiley-Interscience, 1988.

[8] D. Filipovic, "Exponential-polynomial families and the term structure of interest rates," *Bernoulli*, vol. 6, no. 6, pp. 1081–1107, 2000.

[9] B. Friedlander and J. Francos, "Estimation of amplitude and phase of non-stationary signals," in *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, 1993, pp. 431–435 vol.1.

[10] Y. Pantazis, O. Rosec, and Y. Stylianou, "Adaptive am/fm signal decomposition with application to speech analysis," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 2, pp. 290–300, 2011.

[11] F. Auger and P. Flandrin, "Improving the readability of time-frequency and time-scale representations by the reassignment method," *Signal Processing, IEEE Transactions on*, vol. 43, no. 5, pp. 1068–1089, 1995.

[12] S. Marchand and P. Depalle, "Generalization of the derivative analysis method to Non-Stationary sinusoidal modeling," *Digital Audio Effects, 2008. Proceedings of the 11th Int. Conference on (DAFx-08)*, 2008. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00351950/en/

[13] M. Betser, "Sinusoidal polynomial parameter estimation using the distribution derivative," *Signal Processing, IEEE Transactions on*, vol. 57, no. 12, pp. 4633 –4645, dec. 2009.

[14] S. Muševič and J. Bonada, "Comparison of non-stationary sinsoid estimation methods using reassignment and derivatives," Barcelona, Spain, Jul. 2010. [Online]. Available: http://smcnetwork.org/files/proceedings/2010/14.pdf

[15] B. Hamilton and P. Depalle, "A unified view of non-stationary sinusoidal parameter estimation methods using signal derivatives," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, march 2012, pp. 369 –372.

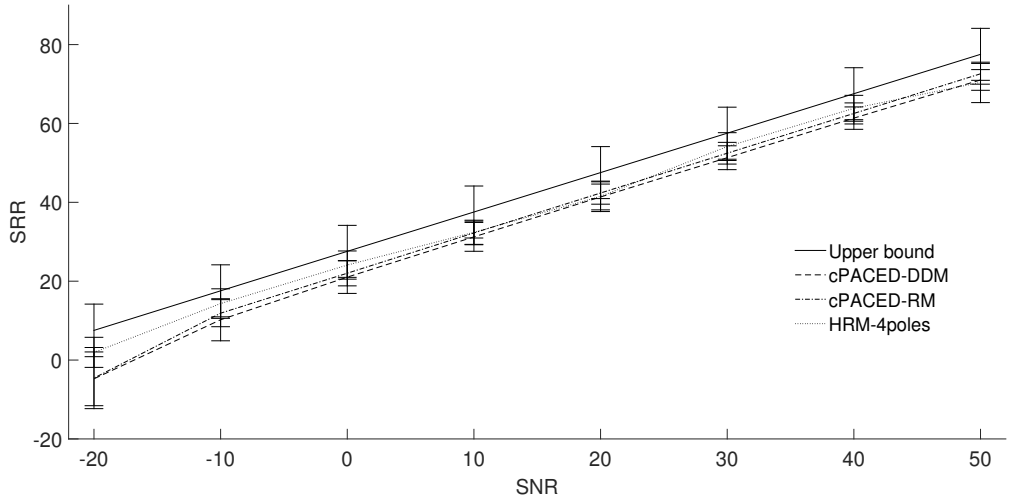


Figure 2: SRR mean and variance for cPaced signals.

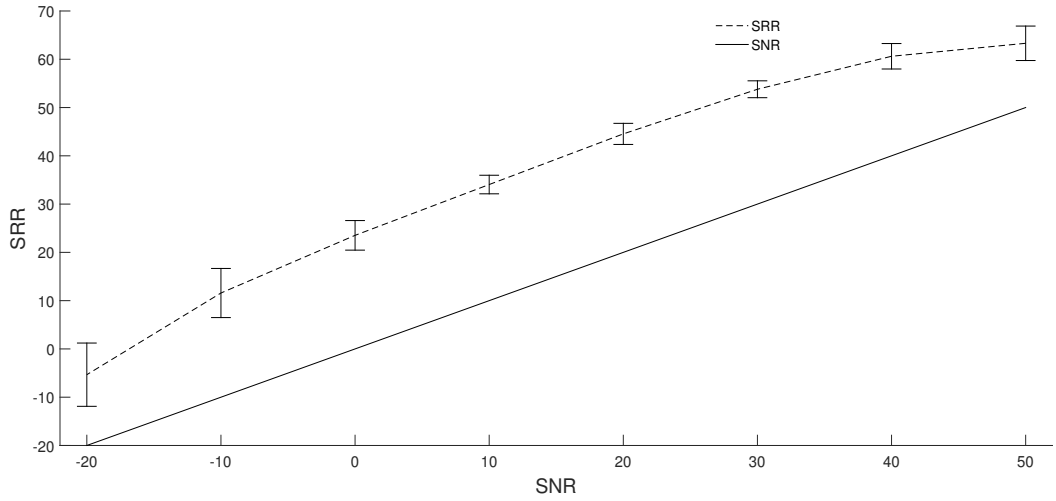


Figure 3: mean and variance of SRR means for all combinations of [0,1,2,3] amplitude polynomial degrees and [1,2,3] exponential polynomial degrees sets. Details are given in Table 1.

SNR	a0r1	a0r2	a0r3	a1r1	a1r2	a1r3	a2r1	a2r2	a2r3	a3r1	a3r2	a3r3
50	72.2	66.7	64.6	62.8	63.9	62.3	64.2	61.2	60.5	62.3	60.8	58.2
40	67.0	63.1	61.7	59.2	61.3	60.0	61.7	59.2	58.7	60.1	58.9	56.6
30	56.6	56.5	54.2	54.3	54.6	53.2	54.5	53.0	52.4	53.3	52.4	50.5
20	50.0	46.4	44.7	44.9	45.2	43.8	44.8	43.6	42.9	44.0	43.0	41.1
10	36.2	35.2	34.7	36.8	34.8	33.6	35.0	33.5	32.8	33.9	32.7	29.4
0	25.9	25.7	24.9	26.4	25.2	23.2	25.2	23.1	22.1	23.8	21.7	15.1
-10	13.7	16.8	14.9	15.4	14.6	10.3	14.7	10.9	8.4	13.3	7.8	-1.8
-20	3.2	1.4	-3.0	-1.5	-2.4	-7.4	-1.6	-6.9	-13.1	-2.5	-10.4	-19.8

Table 1: SRR mean for several polynomial degree combinations. Column numbers in top row indicate amplitude and exponential complex polynomial degrees respectively (e.g. a1r3 means amplitude polynomial degree 1 and exponential polynomial degree 3).

[16] E. Chassande-Mottin, F. Auger, and P. Flandrin, *Reassignment*. ISTE, 2010, pp. 249–277. [Online]. Available: http://dx.doi.org/10.1002/9780470611203.ch9

[17] M. S. Xue Wen, "Notes on model-based non-stationary sinusoid estimation methods using derivative," in *Digital Audio Effects, 2009. Proceedings of the 12th Int. Conference on (DAFx-09)*, 2009.

[18] J. B. Sašo Muševič, "Derivative analysis of complex polynomial amplitude, complex exponential with exponential damping," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013.

[19] I. Daubechies, J. Lu, and H.-T. Wu, "Synchrosqueezed wavelet transforms: An empirical mode decomposition-like tool," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 243 – 261, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1063520310001016

[20] R. Badeau, B. David, N. Bertin, J. Echeveste, M. Lagrange, O. Derrien, and S. Marchand, "Desam toolbox v1.1." [Online]. Available: http://www.tsi.telecom-paristech.fr/aao/en/2010/03/29/desam-toolbox-2/

[21] B. Porat and B. Friedlander, "On the accuracy of the kumaresan-tufts method for estimating complex damped exponentials," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 35, no. 2, pp. 231–235, 1987.

[22] Y. Hua and Y. Zhang, "Perturbation analysis of tk method for harmonic retrieval problems," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 36, no. 2, pp. 228–240, 1988.

[23] Sage mathematical software. [Online]. Available: http://www.sagemath.org

[24] S. Scholler and P. Purwins, "Sparse approximations for drum sound classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, pp. 933 – 940, 2011 2011.

[25] M. Betser, P. Collen, G. Richard, and B. David, "Estimation of frequency for AM/FM models using the phase vocoder framework," *Signal Processing, IEEE Transactions on*, vol. 56, no. 2, pp. 505–517, 2008.

[26] R. Badeau, "Méthodes à haute résolution pour l'estimation et le suivi de sinusoïdes modulées. application aux signaux de musique," Ph.D. dissertation, École Doctorale d'Informatique, Télécommunications et Électronique de Paris, 2005.

[27] M. Bartkowiak, "A complex envelope ssinusoidal model for audio codig," 2007.

[28] M. Christensen and S. van de Par, "Efficient parametric coding of transients," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 4, pp. 1340 –1351, july 2006.

[29] R. McAulay and T. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 34, no. 4, pp. 744 – 754, aug 1986.

[30] G. Kafentzis, Y. Pantazis, O. Rosec, and Y. Stylianou, "An extension of the adaptive quasi-harmonic model," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, march 2012, pp. 4605 –4608.

[31] K. Hermus, W. Verhelst, P. Lemmerling, P. Wambacq, and S. V. Huffel, "Perceptual audio modeling with exponentially damped sinusoids," *Signal Processing*, vol. 85, no. 1, pp. 163 – 176, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165168404002361

[32] J. Jensen, R. Heusdens, and S. Jensen, "A perceptual subspace approach for modeling of speech and audio signals with damped sinusoids," *Speech and Audio Processing, IEEE Transactions on*, vol. 12, no. 2, pp. 121 – 132, march 2004.

[33] J. Nieuwenhuijse, R. Heusens, and E. Deprettere, "Robust exponential modeling of audio signals," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 6, may 1998, pp. 3581 – 3584 vol.6.

[34] R. Badeau, B. David, and G. Richard, "High-resolution spectral analysis of mixtures of complex exponentials modulated by polynomials," *Signal Processing, IEEE Transactions on*, vol. 54, no. 4, pp. 1341 – 1350, april 2006.

[35] M. Desainte-Catherine and S. Marchand, "High precision fourier analysis of sounds using signal derivatives," *Audio Engineering Society, 1953 Journal of*, 1998. [Online]. Available: http://citeseer.ist.psu.edu/desainte-catherine98high.html

[36] P. Djuric and S. Kay, "Parameter estimation of chirp signals," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 38, no. 12, pp. 2118–2126, 1990.

DESIGN PRINCIPLES FOR LUMPED MODEL DISCRETISATION USING MÖBIUS TRANSFORMS

François G. Germain, Kurt J. Werner

Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University, Stanford, CA, USA
fgermain@stanford.edu, kwerner2@stanford.edu

ABSTRACT

Computational modelling of audio systems commonly involves discretising lumped models. The properties of common discretisation schemes are typically derived through analysis of how the imaginary axis on the Laplace-transform s-plane maps onto the Z-transform z-plane and the implied stability regions. This analysis ignores some important considerations regarding the mapping of individual poles, in particular the case of highly-damped poles. In this paper, we analyse the properties of an extended class of discretisations based on Möbius transforms, both as mappings and discretisation schemes. We analyse and extend the concept of frequency warping, well-known in the context of the bilinear transform, and we characterise the relationship between the damping and frequencies of poles in the s- and z-planes. We present and analyse several design criteria (damping monotonicity, stability) corresponding to desirable properties of the discretised system. Satisfying these criteria involves selecting appropriate transforms based on the pole structure of the system on the s-plane. These theoretical developments are finally illustrated on a diode clipper nonlinear model.

1. INTRODUCTION

Computational modelling of audio systems is a major topic of interest, including emulation of existing electronic and acoustic systems such as vintage audio effects or acoustic instruments. When the dynamics of a physical system are known through its transfer function, a common procedure is to discretise it using the bilinear transform or the forward or backward Euler method.

The general properties of those discretisations are often derived from analysing the mapping of the imaginary axis in the Laplace transform s-plane onto the Z-transform z-plane [1]. Since these mappings fall into the category of conformal mappings, the imaginary axis always maps to a circle or a line [2]. Additional analysis of these transforms can be found in [3], where a method to design digital constant-Q filters is presented using hybrid transforms between the bilinear transform and the backward Euler method. More generally, in the case where a state-space representation of the system is available, all those methods correspond to different numerical schemes [1, 4], which are well-studied methods in the field of numerical analysis [5]. In particular, those methods are well-studied in terms of numerical accuracy and stability as a function of the system dynamics and the chosen sampling rate for linear time-invariant (LTI) systems.

In the audio literature, little attention has been given to the simulation distortion introduced by those transforms due to the distortions of pole properties. However, the analysis of the mapping of the imaginary axis can offer only guarantees regarding the response of LTI systems in a steady-state context. For example, the magnitude response of an LTI system after discretisation with the bilinear transform can be shown to correspond exactly to the magnitude response of the original system up to a contraction and distortion of the imaginary axis [4, 6]. In cases where that frequency distortion is problematic, frequency warping methods can be used with the bilinear transform to compensate for it [4, 7]. However, warping breaks the equivalence between the bilinear transform and its numerical equivalent, the trapezoidal method, adding additional errors terms to the transform to achieve frequency matching. Additionally, warping requires previous knowledge of the system to know the needed level of warping. A method was also proposed in order to derive digital parametric equalizer designs with a non-zero Nyquist-frequency gain and better match the non-warped response of analog equalizers [8]. Those techniques do not consider the distortion introduced in the pole locations, which can produce noticeable differences in the system response for non-steady state conditions (e.g. transients, modulated input). Similar observations have been made in the context of artificial reverberations regarding the noticeable influence of all-pass filters on the short-term audio colouration of those effects despite their flat magnitude response [9]. Additionally, one must remain careful when considering the properties of those methods for discretising nonlinear and/or time-varying systems. Other typical filter designs (e.g. impulse-invariant methods [10], matched Z-transform [11], Prony's method [7]) based on exact pole placement can improve the transient behaviour of LTI systems, but they are typically impractical in the context of nonlinear and/or time-varying systems.

More advanced systematic discretisation schemes (e.g. high-order methods, fractional bilinear transform) are described in the numerical analysis literature [5], or in the audio context, in studies on physical-modelling-based synthesis [12, 13] or filter design [3, 14]. Those methods can often handle systems where the bilinear transform and the Euler methods are limited (e.g. strongly nonlinear systems). However, the bilinear transform and the Euler methods have been historically preferred as they preserve the system order, allowing for compact system representation and efficient computation. Backward Euler and the bilinear transform also have the property of unconditionally preserving stability and minimum-phase properties [1, 13]. This proves useful as digital audio effects work with a prescribed sampling frequency. Another aspect to consider is the limiting of undesirable transient behaviour when several systems are cascaded, more so if those systems are nonlinear and/or time-varying. In this case, the bilinear transform and the Euler methods are generally preferred to filter design methods based on exact pole placement, which can be sensitive to the way the system is implemented, even for LTI systems [1].

One case where the bilinear transform and the Euler methods are limited is systems with highly-damped poles. Numerical analysis literature shows that discretisation of such systems using the trapezoidal rule shows high-frequency oscillations at its output in transient mode [15, 16]. Indeed, the mapping of pole contours with identical damping in the s-plane, while still circles, are not centred around the origin, meaning their damping in the z-plane is not frequency-independent [3]. As a result, the properties of a mapped pole in the z-plane depend on the pole damping in a way that can noticeably impact the dynamics of the discretised system transient response. Methods such as frequency warping cannot compensate efficiently for this despite their use of a priori information of the system pole locations in the s-plane. This information can be available as, for example, we can often know where all the possible poles of a parametrised equalizer, or all the instantaneous poles of a nonlinear analog circuit will lie across all the operating conditions of those systems. However, with such information, we can imagine a generalisation of the bilinear transform and the Euler methods that could enforce a wider range of desirable properties for a discretised system (e.g. stability, pole damping and frequency) under all its known operating conditions.

In Sec. 2, we present a theoretical extension and generalised analysis of typical discretisations (e.g. bilinear transform) in the context of the Möbius transforms. In Sec. 3, we generalise the analysis of s-plane distortion introduced by discretisation and propose criteria in order to design discretisation schemes alleviating typical unwanted behaviour. In Sec. 4, we apply those principles to the discretisation of a typical audio system.

2. DISCRETISATIONS AS MÖBIUS TRANSFORMS

2.1. Lumped models

Audio systems can often be represented as single-input single-output (SISO) LTI lumped models (e.g. analog filters) with input $u(t)$ and output $y(t)$ (with Laplace transforms $U(s)$ and $Y(s)$) can be described by their transfer function in the Laplace domain [1]:

$$H(s) = \frac{Y(s)}{U(s)} = \frac{\sum_{m=0}^{M} b_m s^m}{\sum_{n=0}^{N} a_n s^n}. \tag{1}$$

This generalises to multiple-input multiple-output (MIMO) systems with the state-space representation

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{aligned} \tag{2}$$

and state variables $\mathbf{x}$. The generalised system transfer function

$$\mathbf{H}(s) = \mathbf{C}\left(s\mathbf{I} - \mathbf{A}\right)^{-1}\mathbf{B} + \mathbf{D} \tag{3}$$

has poles corresponding to the eigenvalues of $\mathbf{A}$ [6].

Some nonlinear time-varying systems can also be described in a state-space form as

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)) \tag{4a}$$

$$\mathbf{y}(t) = \mathbf{g}(t, \mathbf{x}(t), \mathbf{u}(t)). \tag{4b}$$

For a given operating point $(t_0, \mathbf{x}_0, \mathbf{u}_0)$, we linearise the system around that point using a small perturbation approximation as the state-space model

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}_0\mathbf{x}(t) + \mathbf{B}_0\mathbf{v}_0(t) \\ \mathbf{y}(t) &= \mathbf{C}_0\mathbf{x}(t) + \mathbf{D}_0\mathbf{v}_0(t) \end{aligned} \tag{5}$$

with $\mathbf{v}_0(t) = \left[(\mathbf{u}(t) - \mathbf{u}_0)^T, \quad (t - t_0), \quad \mathbf{x}_0^T\right]^T$, in which case we interpret the eigenvalues of $\mathbf{A}_0$ as "instantaneous" poles [13].

2.2. Transfer function discretisation and Möbius transforms

Computational simulation of LTI systems typically requires digitising their transfer function $\mathbf{H}(s)$ (Eq. (3)) at a sampling interval T, and several methods have been proposed in the literature [1]. Methods such as the bilinear transform (BT), backward Euler (BE), and forward Euler (FE) have the advantage of being simple, order-preserving, aliasing-free and independent of the transfer function form. Those methods correspond to the substitution of s by $\mathcal{T}(z)$ in the transfer function. They can be interpreted as mappings $z \mapsto s = \mathcal{T}(z)$ (and its inverse $s \mapsto z = \mathcal{T}^{-1}(s)$) between the Laplace transform s-plane and the Z-transform z-plane, with

$$\mathcal{T}_{\mathrm{BT}}(z) = \tfrac{2}{T}\tfrac{1-z^{-1}}{1+z^{-1}}, \mathcal{T}_{\mathrm{BE}}(z) = \tfrac{1-z^{-1}}{T}, \mathcal{T}_{\mathrm{FE}}(z) = \tfrac{1-z^{-1}}{Tz^{-1}}. \tag{6}$$

A point in the s-plane is decomposed as $s = \sigma + j\Omega$ with σ defined as the *damping* and Ω as the *frequency* of that point. A point in the z-plane is decomposed as $z = re^{j\omega}$ ($r \geq 0$ and $\omega \in]-\pi, \pi]$) with $\log(r)/T$ defined as the *damping* and ω/T as the *frequency* of that point.

These mappings belong to the class of *conformal mappings* (i.e. angle-preserving transforms) [2]. They also belong to the subclass of *Möbius transforms* which contains all rational order-preserving mappings. Möbius transforms have the form [17]

$$\mathcal{T}(z) = \frac{a + bz^{-1}}{c + dz^{-1}} \text{ and } \mathcal{T}^{-1}(s) = -\frac{ds - b}{cs - a} \tag{7}$$

with $ad - bc \neq 0$. They map circles (including lines as circles of infinite radius) in the origin plane to circles (including lines) in the target plane [2]. Möbius transform coefficients $(a, b, c, d) \in \mathbb{C}^4$ are defined up to common multiplying factor $\gamma \in \mathbb{C}$ since $(\gamma a, \gamma b, \gamma c, \gamma d)$ corresponds to the same transform. A transform is uniquely defined (up to factor γ) by setting the mapping of 3 separate points.

In digital filter design, filters with real coefficients (or equivalently with a conjugate symmetric Fourier transform with respect to DC) are typically preferred. This corresponds to the use of symmetric mappings with respect to the real axis (or equivalently which maps the real axis onto itself). That property is guaranteed if and only if the transform coefficients (a, b, c, d) are all real up to a common multiplying factor γ (possibly complex).

Also, we can note that, in both the s-plane and the z-plane, the upper part of the plane (i.e. points with positive imaginary part $\Im(\cdot)$) corresponds to points with positive frequency ($\Omega \geq 0$ and $\omega \geq 0$). For this reason, we limit the discussion to transformations mapping the upper plane of the s-plane to the upper plane to the z-plane. For real coefficients (a, b, c, d), we get:

$$\Im(z) = \frac{\Im(s)(ad - bc)}{\Im(s)^2 c^2 + (a - \Re(s)c)^2} \tag{8}$$

so that $\Im(s) \geq 0 \Leftrightarrow \Im(z) \geq 0$ is true if and only if $ad - bc > 0$.

Another possible condition of interest is that the origin $s = 0$ maps to a DC point in the z-plane (i.e. with $\omega = 0$); this property holds if and only if a and b have opposite signs. We may also want to consider only mappings such that $|\Omega| \to \infty$ maps to $\omega \to \pm\pi$; this property holds if and only if c and d have same signs.

In the rest of the paper, we consider only Möbius transforms with real coefficients verifying $ad - bc > 0$. By construction, these transforms are all order-preserving and aliasing-free.

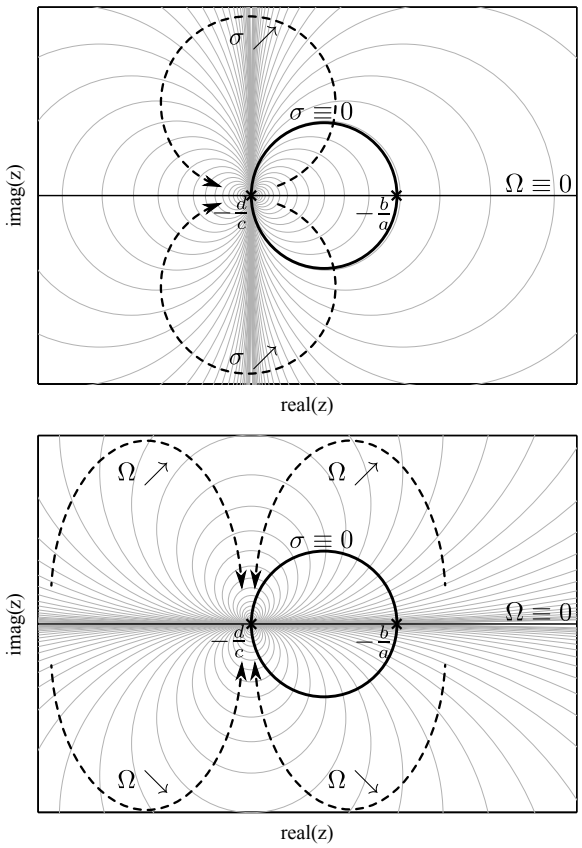


Figure 1: *Circles (in grey) corresponding to constant σ (top) and constant Ω (bottom) in the z-plane. The black horizontal axis corresponds to the real axis in the z-plane, mapped from the real axis $\Omega \equiv 0$. The black circle corresponds to the region mapped from the imaginary axis $\sigma \equiv 0$. In the case of the bilinear transform, the circle correspond to the unit circle, with $d/c = 1$ and $b/a = -1$. When following the dashed arrows, $\nearrow$ indicates increasing quantities and $\searrow$ decreasing ones.*

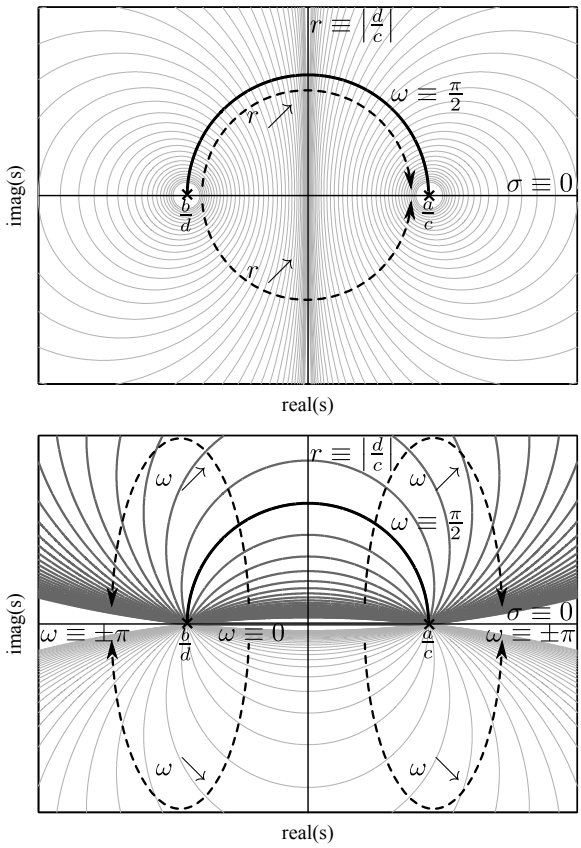


Figure 2: *Circles (in grey) corresponding to constant r (top) and arcs corresponding to constant ω (bottom) in the s-plane. Darker arcs correspond to $\omega \in [0, \pi]$. The black axis corresponds to the real axis in the s-plane, mapped from $\omega \equiv 0$ and $\omega \equiv \pm\pi$. The black arc corresponds to the $\omega \equiv \pi/2$. The black vertical line corresponds to $r \equiv |d/c|$. In the case of the bilinear transform, the arc belongs to the unit circle and the vertical line corresponds to the imaginary axis, with $b/d = -2/T$ and $a/c = 2/T$.*

2.3. Finite difference methods

Computational simulation of a system defined by Eq. (4) is a typical problem in numerical analysis [1, 5]. To digitise the system at a sampling interval T, we can compute the next sample $\mathbf{x}_n$ at $t_n = nT$ by applying common numerical methods. One-step methods correspond to methods which only use quantities evaluated at t_{n-1} and t_n, including $\mathbf{f}_n = \mathbf{f}(t_n, \mathbf{x}_n, \mathbf{u}_n)$ and $\mathbf{f}_{n-1} = \mathbf{f}(t_{n-1}, \mathbf{x}_{n-1}, \mathbf{u}_{n-1})$. Such methods include:

- forward Euler: $\mathbf{x}_n = \mathbf{x}_{n-1} + T\,\mathbf{f}_{n-1}$,
- backward Euler: $\mathbf{x}_n = \mathbf{x}_{n-1} + T\,\mathbf{f}_n$, and
- trapezoidal rule: $\mathbf{x}_n = \mathbf{x}_{n-1} + T\,(\mathbf{f}_n + \mathbf{f}_{n-1})\,/2$.

In the case of LTI systems, these methods are equivalent to order-preserving mappings of the transfer function, with forward and backward Euler being equivalent to the mappings in Eq. (6), and the trapezoidal rule being equivalent to the bilinear transform.

Like the Möbius transforms, these methods can be expressed as

$$a\mathbf{x}_n + b\mathbf{x}_{n-1} = c\,\mathbf{f}_n + d\,\mathbf{f}_{n-1} \tag{9}$$

where the parameters (a, b, c, d) are defined up to a constant factor.

A common way to classify these methods is accuracy, i.e. the order in T of the leading error term of the Taylor series expansion

$$\begin{aligned} a\mathbf{x}_n + b\mathbf{x}_{n-1} - c\,\mathbf{f}_n - d\,\mathbf{f}_{n-1} &= (a+b)\mathbf{x}_{n-1} + aT\dot{\mathbf{x}}_{n-1} \\ &- (c+d)\dot{\mathbf{x}}_{n-1} + (\tfrac{aT^2}{2} - cT)\ddot{\mathbf{x}}_{n-1} + \mathcal{O}(cT^2 + aT^3). \end{aligned} \tag{10}$$

By cancelling some of those terms, we obtain methods with various orders of accuracy. The only 2nd-order method that cancels the terms in $\mathbf{x}$, $\dot{\mathbf{x}}$, and $\ddot{\mathbf{x}}$ corresponds to the trapezoidal rule. 1st-order methods cancel the terms in $\mathbf{x}$ and $\dot{\mathbf{x}}$ with $a = -b$ and $c + d = aT$, and include, among others, backward Euler ($c = aT$ and $d = 0$) and forward Euler ($c = 0$ and $d = aT$). All other methods are 0th-order, even if some of the terms in Eq. (10) can still be cancelled (e.g. the term in $\mathbf{x}$ if we have $a = -b$).

It can seem counter-intuitive to consider 0th-order methods, meaning that error does not unconditionally vanish as we decrease the sampling interval. However, we can observe that:

- in audio, the sampling period is usually fixed so that higher-order error in T does not always translate in less error for the discretised system,
- the true solution for typical conditions are often such that $||\mathbf{x}(t)|| \rightarrow_{t\to\infty} 0$ and $||\dot{\mathbf{x}}(t)|| \rightarrow_{t\to\infty} 0$ (e.g. impulse response) so that the numerical solution error vanishes at $t \to \infty$ for all those methods,
- for other conditions when $||\dot{\mathbf{x}}(t)|| \rightarrow_{t\to\infty} 0$ (e.g. step response) but $||\mathbf{x}(t)|| \nrightarrow_{t\to\infty} 0$, the error for methods with $a = -b$ also vanishes at $t \to \infty$.

It can also seem counter-intuitive to consider 1st-order methods, when a 2nd-order method is available. However, the usage of the backward Euler method shows it is already standard practice to trade numerical accuracy for other types of desirable properties.

Here, it seems logical to restrict to methods such that $a \neq 0$ to ensure that the term in $\mathbf{x}_n$ is present. We can set it to $1/T$ as the coefficients are defined up to a multiplying factor.

3. POLE MAPPING AND WARPING

3.1. Bilinear transform frequency warping interpretation

In digital filter design, the bilinear transform is typically parameterised by a gain factor η as [7]

$$\mathcal{T}(z) = \eta \frac{1 - z^{-1}}{1 + z^{-1}}\,. \tag{11}$$

We interpret η as equal to $2/T'$ with T' a parameter potentially different from the sampling period T such that:

$$\mathcal{T}(z) = \frac{2}{T'} \frac{1 - z^{-1}}{1 + z^{-1}}\,. \tag{12}$$

For the canonical bilinear transform ($T = T'$, see Eq. (6)), we know that the imaginary axis $\sigma \equiv 0$ is mapped to the unit circle $r \equiv 1$ with the frequency distortion [4]

$$\omega = 2\tan^{-1}\left(\Omega T/2\right)/T\,. \tag{13}$$

For any T', those transforms map the imaginary axis to the unit circle, the DC point $s = 0$ to $z = 1$, and $|s| \to \infty$ to $z = -1$, with different distortion of the frequencies. One additional mapping condition between a point s on the imaginary axis and a point z on the unit circle can be used to uniquely define the desired frequency distortion. The frequency warping method aim at selecting T' so that the point $s_0 = j\Omega_0$ (only for $\Omega_0 \in \,]-\pi, \pi[$) maps to $z_0 = e^{j\Omega_0}$ (i.e. $\Omega_0 = \omega_0 T$), meaning that the frequencies of s_0 and z_0 match [7]. This is achieved with

$$T' = 2\tan(\Omega_0 T/2)/\Omega_0. \tag{14}$$

Warping can be interpreted in the context of numerical methods by looking at the equation

$$\dot{x}(t) = j\Omega_0 x(t),\ x(0) = 1 \tag{15}$$

and discretising it with the trapezoidal rule

$$x_n = \xi x_{n-1},\ x_0 = 1 \quad \text{with} \quad \xi = \frac{1 + j\Omega_0 T/2}{1 - j\Omega_0 T/2}\,. \tag{16}$$

The solutions to Eq. (15) and Eq. (16) are $x(t) = e^{j\Omega_0 t}$ ($t \geq 0$) and $x_n = \xi^n$ ($n \geq 0$). Since $\xi = e^{j\omega_0 T}$ with

$$\omega_0 = 2\tan^{-1}\left(\Omega_0 T/2\right)/T\,, \tag{17}$$

the solution to the discretised system presents a frequency-dependent phase lag due to eigenvalue distortion introduced by the discretisation [5]. That distortion matches the frequency distortion from the bilinear transform, meaning that the frequency warping can be interpreted in the numerical analysis framework as:

- Modifying Eq. (4a) so that the eigenvalues of the system shift by a multiplicative factor of T'/T. In filter design, that is the case where the filter coefficients are *pre-warped* and the canonical bilinear transform is used, or equivalently
- Changing the time step for the discretisation of Eq. (4a) to T' (q'_n is quantity q evaluated at nT' instead of nT) as

$$\begin{aligned}&(\mathbf{x}_n - \mathbf{x}_{n-1})/T = (\mathbf{f}_n + \mathbf{f}_{n-1})/2\\ &\mapsto (\mathbf{x}'_n - \mathbf{x}'_{n-1})/T = (\mathbf{f}'_n + \mathbf{f}'_{n-1})/2\end{aligned} \tag{18}$$

  with $\mathbf{f}_n = \mathbf{f}(t_n, \mathbf{x}_n, \mathbf{u}_n)$ and $\mathbf{f}'_n = \mathbf{f}(t'_n, \mathbf{x}'_n, \mathbf{u}'_n)$, and modifying the discretisation of Eq. (4b)

$$\mathbf{y}_n = \mathbf{g}(t_n, \mathbf{x}_n, \mathbf{u}_n) \mapsto \mathbf{y}_n = \mathbf{g}(t'_n, \mathbf{x}'_n, \mathbf{u}'_n)\,, \tag{19}$$

  effectively creating a mismatch between the time steps of $\mathbf{y}$ and that of $(t, \mathbf{x}, \mathbf{u})$. In filter design, that corresponds to changing T in T' in the bilinear transform.

Effectively, frequency warping in the bilinear transform is equivalent to *compensating for the phase lag* introduced by the trapezoidal rule in the numerical solution of Eq. (15). As the phase lag is frequency-dependent, it can only be cancelled for a single frequency. This process can be thought to be similar to the issue of minimising numerical dispersion when modelling of the 2D/3D wave equation using finite-difference meshes [12, 18].

3.2. Mapping equations

The analysis presented in the previous section focuses on the mapping of the imaginary axis. However, additional attention can be given to the mapping of all the points of the s-plane in order to further generalise the concept of warping. For any point $s = \sigma + j\Omega$, the transform defined by (a, b, c, d) maps it to $z = re^{j\omega}$ with

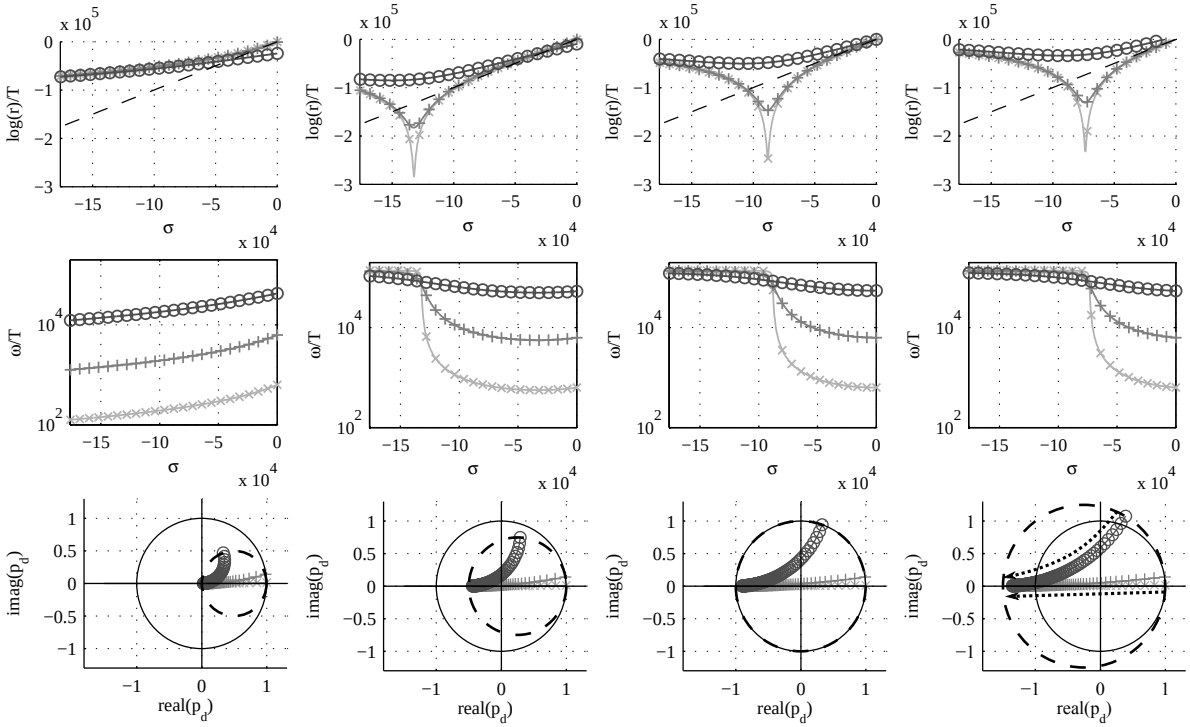
$$\begin{aligned}r &= \sqrt{\frac{\Omega^2 d^2 + (b - \sigma d)^2}{\Omega^2 c^2 + (a - \sigma c)^2}}\\ \tan\frac{\omega}{2} &= \frac{\Omega(ad - bc)}{(a - c\sigma)^2(b - d\sigma)^2 + \Omega^2 cd - r((a - c\sigma)^2 + \Omega^2 c^2)}\end{aligned} \tag{20}$$

We can derive the contours resulting from the mapping of regions of interests, as shown in [3] for the bilinear transform, backward Euler and forward Euler. For constant damping σ, s is mapped onto a circle $\mathcal{C}_\sigma(\zeta_\sigma, R_\sigma)$ (Fig. 1) such that

$$\zeta_\sigma = \frac{1}{2}\left(\frac{d\sigma - b}{a - c\sigma} - \frac{d}{c}\right) \quad \text{and} \quad R_\sigma = \frac{ad - bc}{2|c(c\sigma - a)|}\,. \tag{21}$$

For constant frequency Ω, s is mapped onto a circle (Fig. 1)

$$\zeta_\Omega = -\frac{d}{c} + j\frac{ad - bc}{2c^2\Omega} \quad \text{and} \quad R_\Omega = \frac{ad - bc}{2c^2|\Omega|} \tag{22}$$



(a) $\alpha = 0$ (BE) (b) $\alpha = 0.5$ (c) $\alpha = 1$ (BT) (d) $\alpha = 1.5$

Figure 3: *Trajectories of pole p in the z-plane for 3 different constant frequencies Ω as a function of σ (0.1 kHz: light grey $\times$, 1 kHz: medium grey $+$, 10 kHz: dark grey $\circ$). Top: $\log(r)/T$ as a function of σ (The dashed line indicates $\log(r)/T \equiv \sigma$); middle: ω/T as a function of σ; bottom: location of p in the z-plane as a function of σ. The imaginary axis $\sigma \equiv 0$ maps to the dashed circle; p starts on the right part of the dashed circle for $\sigma = 0$ and travels towards the left and the real axis as σ increases, as shown by the dotted arrows.*

such that the centre ζ moves along a vertical line going through the mapped point for infinity, $z = -d/c$.

For both σ and Ω, we see that all the circles contain the mapped point for infinity $-d/c$ and asymptotically shrink towards it, with their centre ζ moving along a line (vertical for constant Ω and horizontal for constant σ). We can also analyse the inverse mapping in order to analyse regions generating digital filters with constant parameters. For constant damping $\log(r)/T$ (or equivalently constant r), z is mapped onto an arc (Fig. 2) such that

$$\zeta_r = \frac{acr^2 - bd}{c^2r^2 - d^2} \text{ and } R_r = \frac{r(ad - bc)}{|c^2r^2 - d^2|}\,. \tag{23}$$

For constant frequency ω/T and $\omega \geq 0$, z is mapped onto the part with non-negative imaginary part ($\sigma \geq 0$) of a circle (Fig. 2) such that

$$\zeta_\omega = \frac{ad + bc}{2cd} - j\frac{ad - bc}{2cd\tan\omega} \text{ and } R_\omega = \frac{ad - bc}{2|cd\sin\omega|} \tag{24}$$

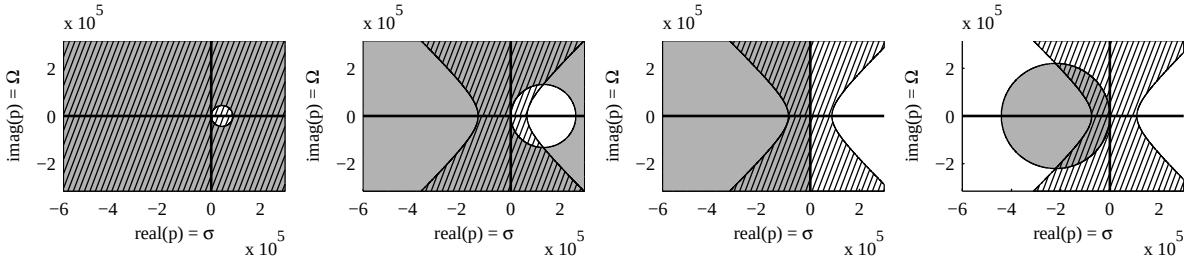
and for $\omega < 0$, z is mapped onto the part with non-positive imaginary part ($\sigma \leq 0$) of that same circle (Fig. 2).

Similarly to the forward mapping, for both r and ω, we see that all the circles have their centre organised around a single line (vertical for constant ω and horizontal for constant r).

3.3. Generalised warping

For the general class of Möbius transforms, three different mapping conditions between s- and z-planes define a unique Möbius transform (up to a multiplicative factor). In typical transforms (BT, BE, FE), we set mapping conditions for the origin ($s = 0$) and infinity ($|s| \to \infty$), so that one mapping degree of freedom remains. However, as mentioned in Sec. 2.2, we wish to limit ourselves to Möbius transforms with real coefficients, which means that not all third mapping conditions can be fulfilled.

While we cannot control the mapping of any additional point, other warping processes can be considered. For example, if a transform with real coefficients and $ad-bc > 0$ maps $s_1 \mapsto z_1$ and $s_2 \mapsto z_2$, then there is a unique circle $\mathcal{C}_s$ centred on the real axis that maps to $\mathcal{C}_z$ such that $\{s1, s2\} \in \mathcal{C}_s$ and $\{z_1, z_2\} \in \mathcal{C}_z$. Those two circles intersect the real axis in two pair of points $(\tilde{s}_1, \tilde{z}_1)$ and $(\tilde{s}_2, \tilde{z}_2)$ mapped to each other. Recall that we can force the mapping of an additional point on the positive imaginary axis $s = j\Omega$ ($\Omega > 0$) onto any point of the upper unit circle $z = e^{j\omega}$ ($\omega \in [0, \pi[$) for the bilinear transform using frequency warping. In the same way, we can show it is possible to find a Möbius transform with real coefficients and $ad - bc > 0$ that will map an additional point p of the upper arc of $\mathcal{C}_s$ to any point p_d on the upper arc of $\mathcal{C}_z$. The warping can be done by altering a single parameter χ such that the warped transform corresponds to $(\chi a, \chi b, c, d)$. This



(a) $\alpha = 0$ (BE) (b) $\alpha = 0.5$ (c) $\alpha = 1$ (BT) (d) $\alpha = 1.5$

Figure 4: *Acceptable pole locations for a given α according to Eq.* (29) *(hatched) and Eq.* (30) *(grey).*

approach can then allow for the design of transforms using similar principles to Sec. 3.1, interpreting warping as changing the phase lag introduced by a given Möbius transform in the numerical solution of $\dot{x}(t) = px(t)$. Phase lag cancellation ($\Omega_p = \omega_{p_d}/T$) can be achieved for only a subset of applicable frequencies Ω_p.

Another option would be to remove the condition either for the origin or infinity, in which case we can exactly specify the mapping of another point s. In this case, the transform coefficients are still ill-defined. An additional condition is needed, for example by cancelling one or several of the terms in Eq. (10), if it does not conflict with the mapping conditions. Otherwise, we can also find sets of transforms that would satisfy an alternative set of criteria such as the one presented in the next two sections.

3.4. Damping monotonicity conditions

In the rest of the paper, we consider only transforms of the form $(a, b, c, d) = (\frac{1+\alpha}{T}, -\frac{1+\alpha}{T}, 1, \alpha)$, even though all the discussed properties extend readily to all previous Möbius transforms. This subclass of transforms includes the bilinear transform ($\alpha = 1$), the forward ($\alpha \to \infty$), and the backward ($\alpha = 0$) Euler methods. As numerical schemes, they are all 1st- or 2nd-order schemes and can be viewed as forward Euler with an added dissipative term [16]:

$$\begin{aligned}\mathbf{x}_n &= \mathbf{x}_{n-1} + T(\mathbf{f}_n + \alpha\mathbf{f}_{n-1})/(1+\alpha) \\ &= \underbrace{\mathbf{x}_{n-1} + T\mathbf{f}_{n-1}}_{\text{FE}} + \underbrace{T^2\ddot{\mathbf{x}}_{n-1}/(1+\alpha)}_{\text{dissipative term}} + \mathcal{O}(T^3)\,. \end{aligned} \quad (25)$$

We also assume we have knowledge of a region in the s-plane enclosing all the possible poles of the studied system (or all the possible instantaneous poles for systems modelled as Eq. (4)).

Through such transform, a pole p is mapped to p_d as

$$p = \sigma + j\Omega \mapsto p_d = re^{j\omega} = \frac{1+\alpha+\alpha Tp}{1+\alpha-Tp}\,. \quad (26)$$

The mapping also generates poles at $z = -\alpha$ from the mapping of the zeroes of the continuous systems, but these poles are all cancelled by the zeroes at $z = -\alpha$ from the mapping of the poles if we only consider proper systems [7]. By adapting Eq. (20) to the class of transforms considered here, the mapping between quantities (σ, Ω) and r simplifies to:

$$r^2 = \frac{(1+\alpha+\alpha T\sigma)^2 + (\alpha T\Omega)^2}{(1+\alpha-T\sigma)^2 + (T\Omega)^2} \quad (27)$$

From Eqs. (20) and (27), we can observe in more details the behaviour of the trajectories (Fig. 3) of the poles for different conditions such as $\log(r)/T$ and ω/T at constant Ω as σ decreases from zero into the region of stable poles $\sigma < 0$.

In the plots of $\log(r)/T$ at a constant σ we observe that while the continuous-time and discrete-time damping have a monotonic relationship for lower Ω, we ultimately reach an inflection point for which an increase in the continuous-time damping results in a decrease in discrete-time damping. Passed the inflection point, the discrete-time frequency moves quickly towards $\pm\pi$, so that p_d becomes a resonant (and possibly unstable) pole at the Nyquist frequency. For a given Ω, it is possible to obtain an analytical expression of that inflection point σ_m by solving the equation $\frac{\partial r}{\partial \sigma} = \frac{1}{2r}\frac{\partial r^2}{\partial \sigma} = 0$. From Eq. (27), we get two solutions:

$$\sigma_\pm = \frac{(\alpha^2-1) \pm \sqrt{(\alpha+1)^4 + (2\alpha T\Omega)^2}}{2\alpha T}\,. \quad (28)$$

We can see that $\sigma_+ > 0$ and $\sigma_- < 0$. Also, σ and r have a monotonic relationship for any $\sigma \in [\sigma_-, \sigma_+]$. In the s-plane, this monotonicity condition is verified inside a rectangular hyperbola of semi major axis $\frac{(\alpha+1)^2}{2\alpha T}$ and centre $\left(\frac{\alpha^2-1}{2\alpha T}, 0\right)$:

$$\left(\sigma - \frac{\alpha^2-1}{2\alpha T}\right)^2 - \Omega^2 \leq \left(\frac{(\alpha+1)^2}{2\alpha T}\right)^2 \quad \text{for } \alpha > 0\,, \quad (29)$$

which becomes the half-plane $\sigma > -\frac{1}{T}$ for $\alpha \to \infty$ (i.e. forward Euler) and $\sigma \in \mathbb{R}$ for $\alpha = 0$ (i.e. backward Euler).

If we have prior knowledge of the pole possible locations (e.g. from the physics of the system), we can select α so that all poles lie in that region, and consequently ensure the monotonic relationship between σ and r at constant Ω. From Eq. (25), we can interpret this process as adding enough dissipation to the discretised system in order to move its poles away from the Nyquist frequency.

3.5. Stability conditions

Another condition to verify, in particular if we also want to allow for values $\alpha > 1$ (i.e. discretisation schemes that can produce unstable discrete-time poles from stable continuous-time poles such as forward Euler), is that all the potential poles will be stable, i.e. that $r < 1$. While complete stability analysis of general time-varying system cannot be guaranteed through the sole analysis of its instantaneous poles [19], a preliminary analysis is generally done by verifying the condition $r < 1$ for those poles [13]. In

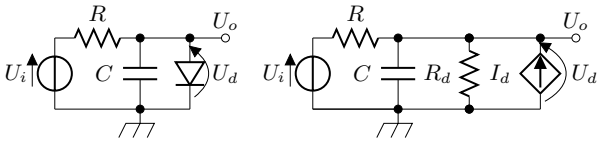


Figure 5: *Diode clipper circuit (left: original; right: linearised).*

the s-plane, the condition is verified over the region described by:

$$\begin{aligned} \left(\sigma - \tfrac{1}{T}\tfrac{1+\alpha}{1-\alpha}\right)^2 + \Omega^2 &> \left(\tfrac{1}{T}\tfrac{1+\alpha}{1-\alpha}\right)^2 \quad \text{for } \alpha \in [0,1[\\ \left(\sigma - \tfrac{1}{T}\tfrac{1+\alpha}{1-\alpha}\right)^2 + \Omega^2 &< \left(\tfrac{1}{T}\tfrac{\alpha+1}{\alpha-1}\right)^2 \quad \text{for } \alpha > 1\,, \end{aligned} \tag{30}$$

which degenerates in the half-plane $\sigma < 0$ for $\alpha = 1$.

Again, with prior knowledge of the pole possible locations, it is possible to ensure that all the poles will lie in those two regions by selecting α appropriately, so that the system stability is guaranteed for all potential poles of the system. The intersection of this condition with the monotonic damping condition in the s-plane can be seen in Fig. 4 for different values of α.

4. SIMULATIONS

We study the diode clipper in Fig. 5 [20] to illustrate those concepts. Following the Shockley diode law, the current I through a diode is modelled as a function of the voltage U across a diode as

$$I = f(U) = I_s \left(e^{U/V_t} - 1\right). \tag{31}$$

If we measure the output voltage U_o around the diode as a function of the driving voltage U_i, the system is described through the state-space representation with state variable U_d:

$$\begin{aligned} \dot{U}_d &= (U_i - U_d)/(RC) - f(U_d)/C \\ U_o &= U_d\,. \end{aligned} \tag{32}$$

At a given time instant t_0, the diode nonlinear characteristic can be linearised into a small perturbation model around U_{d0} with a resistor R_d in parallel with a constant current source I_d as

$$\begin{aligned} f(U_d) &\approx U_d/R_d - I_d \\ R_d &= U_t e^{-U_{d0}/V_t}/I_s \\ I_d &= I_s\left(1 + (U_{d0}/V_t - 1)e^{U_{d0}/V_t}\right) \end{aligned} \tag{33}$$

and the linearised state-space representation becomes

$$\begin{aligned} \dot{U}_d &= -\frac{U_d}{C(R||R_d)} + \begin{bmatrix} \frac{1}{RC}, & \frac{1}{C} \end{bmatrix} \begin{bmatrix} U_i \\ I_d \end{bmatrix} \\ U_o &= U_d\,. \end{aligned} \tag{34}$$

The linearised system has one real pole at

$$p = -\frac{1}{C}\left(\frac{1}{R} + \frac{1}{R_d}\right) = -\frac{1}{C}\left(\frac{1}{R} + \frac{I_s}{V_t}e^{U_{d0}/V_t}\right). \tag{35}$$

In general, it is not possible to know analytically the range of output voltages of this system, but we can estimate it by looking at

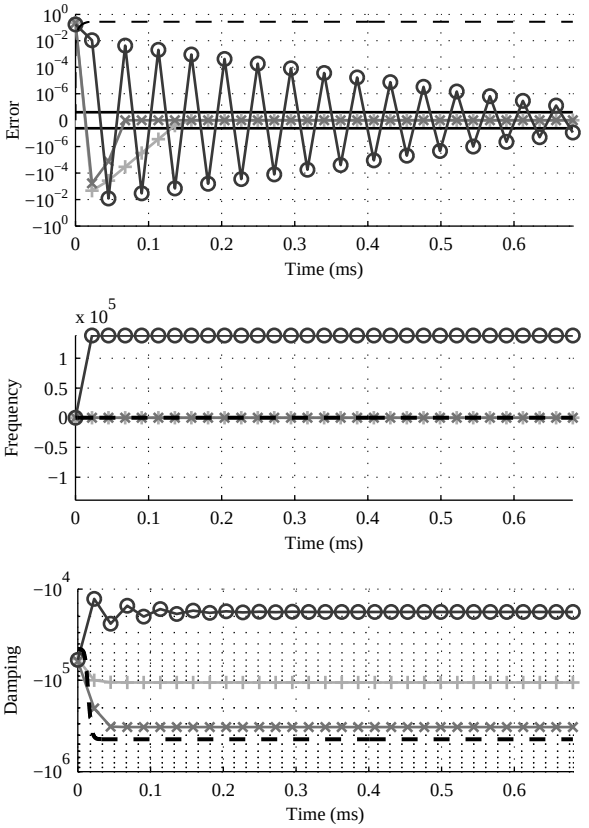


Figure 6: *System response to a $U_i(t) = 0.5u(t)$ for $\alpha = 0$ (light grey +), 0.11 (medium grey ×) and 1 (dark grey ○). Top: reference U_o^{ref} (dashed) and signed error $\hat{U}_o - U_o^{\text{ref}}$ for each α in log-amplitude; middle: instantaneous pole frequencies Ω^{ref} (dashed) and $\hat{\omega}/T$ for each α; bottom: instantaneous pole dampings σ^{ref} (dashed) and $\log(\hat{r})/T$ for each α.*

the steady-state response of the system at maximum and minimum input voltage U_i as the solution of

$$0 = (U_i - U_d)/(RC) - f(U_d)/C \tag{36}$$

that can be found empirically. For a N914 switching diode ($I_s = 2.52\,\text{nA}$, $V_t = 25.85\text{mV}$), $R = 2.2\,\text{k}\Omega$ and $C = 0.01\,\mu\text{F}$, having an input voltage in $[-0.5\,\text{V}, 0.5\,\text{V}]$ produces steady-state output voltages in $[-0.5\,\text{V}, 0.275\,\text{V}]$. As a result the pole p has a continuous-time damping σ in $[-4.42 \times 10^5, -4.55 \times 10^4]$.

To ensure that we pick a Möbius transform of the form $(a, b, c, d) = (\frac{1+\alpha}{T}, -\frac{1+\alpha}{T}, 1, \alpha)$, with α such that all possible continuous-time pole locations $p = \sigma$ ($\sigma < 0$) fall inside the hyperbola described in Eq. (29), we have the condition:

$$\begin{aligned} \forall\sigma,\ -(\alpha+1)/(\alpha T) \le \sigma &\Leftrightarrow -(\alpha+1)/(\alpha T) \le \sigma_{\min} \\ &\Leftrightarrow \alpha(T\sigma_{\min} + 1) \ge -1\,. \end{aligned} \tag{37}$$

For $\sigma \le -1/T$, this condition is always satisfied. Hence, one way to avoid issues with the mapping of the poles is to reduce the sampling interval T (e.g. oversample). For $\sigma < -1/T$, the condition is $\alpha \le -1/(1+T\sigma_{\min})$. For our system sampled at 44.1 kHz

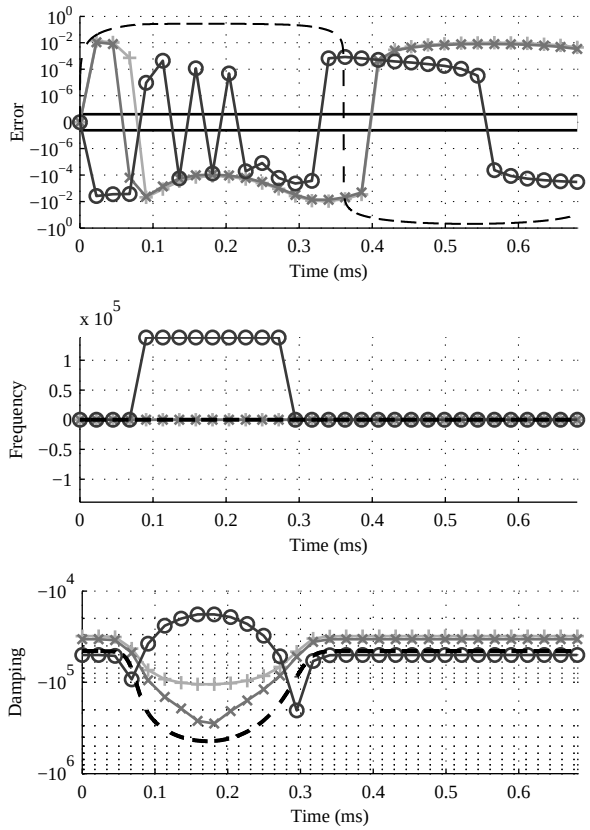


Figure 7: *System response to* $U_i(t) = 0.5\sin(2940\pi t)u(t)$ *(first cycle) for* $\alpha = 0$ *(light grey +),* 0.11 *(medium grey ×) and* 1 *(dark grey ∘). Top: reference* U_o^{ref} *(dashed) and signed error* $\hat{U}_o - U_o^{\mathrm{ref}}$ *for each* α *in log-amplitude; middle: instantaneous pole frequencies* Ω^{ref} *(dashed) and* $\hat{\omega}/T$ *for each* α*; bottom: instantaneous pole dampings* σ^{ref} *(dashed) and* $\log(\hat{r})/T$ *for each* α*.*

($T \approx 22.7\mu$s), the condition to have pole damping monotonicity is $\alpha \lesssim 0.11$. The stability condition translates into a less tight bound $\alpha < (T\sigma_{\min} + 2)/(T\sigma_{\min} - 2)$, i.e. $\alpha \lesssim 1.5$.

We study the impact of instantaneous pole locations by simulating the system response to a step function $U_i(t) = 0.5u(t)$ (Fig. 6) and the onset of a sinusoid $U_i(t) = 0.5\sin(2940\pi t)u(t)$ (Fig. 7), with unit step function $u(t) = \mathbf{1}\{t \geq 0\}$. As reference U_o^{ref} for the system response, we compute the solution to Eq. (32) using the variable-step Runge–Kutta solver `ode45` from MATLAB with absolute and relative accuracies set to machine precision and a maximum time step of $T/8$ (resampled using piecewise cubic interpolation). Then, we compute the response $\hat{U}_o$ of Eq. (32) after discretising it using $\alpha = 0$ (BE), $\alpha = 1$ (BT) and $\alpha = 0.11$ (nearly critical damping monotonicity condition). We solve all implicit update equations using Newton's method [1] to machine precision. For each simulation, we compute the (signed) error $\hat{U}_o - U_o^{\mathrm{ref}}$, and the coordinates (damping and frequency) of the reference instantaneous poles in the s-plane (σ^{ref} and Ω^{ref}, computed using U_o^{ref} as approximation to $U_o(t)$), and of the discretised systems in the z-plane ($\log(\hat{r})/T$ and $\hat{\omega}/T$). The simulations show how the bilinear transform instantaneous pole frequency shifts to π/T when the voltage across the diode becomes high, which results in spurious high-frequency oscillations. On the other hand, backward Euler and $\alpha = 0.11$ instantaneous pole frequencies are always 0 so that no oscillations are triggered.

5. CONCLUSION

In this paper, we presented an generalisation of common methods for discretising transfer functions using Möbius transforms. We study the properties of these transforms when used as discretisation methods in the context of mapping functions from the s-plane to the z-plane and in the parallel context of single-step numerical methods for state-space representations of systems. We introduce general considerations regarding the distortion of pole properties (damping and frequency) as a function of the transform coefficients when mapping from one plane to the other. Finally, we present some criteria based on desirable properties of the pole locations. This allows us to design Möbius transforms that yield discretised systems with those properties as a function of the location of all possible continuous-time system poles. These concepts are illustrated through the simulation of a typical diode clipper circuit, using small-perturbation analysis to predict the transient behaviour of the system as a function of the pole location.

6. REFERENCES

[1] J. O. Smith, *Physical Audio Signal Processing*, W3K Pub., 2010.
[2] Z. Nehari, *Conformal mapping*, Dover Publications, 2011.
[3] T. Stilson, *Efficiently-Variable Non-Oversampled Algorithms in Virtual-Analog Music Synthesis*, Ph.D. thesis, Stanford Univ., 2006.
[4] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, 3rd ed., 2009.
[5] P. Moin, *Fundamentals of engineering numerical analysis*, Cambridge University Press, 2010.
[6] G. F. Franklin, D. J. Powell, and M. L. Workman, *Digital Control of Dynamic Systems*, Prentice Hall, 3rd ed., 1997.
[7] J. O. Smith, *Introduction to Digital Filters with Audio Applications*, W3K Pub., 2007.
[8] S. J. Orfanidis, "Digital parametric equalizer design with prescribed nyquist-frequency gain," *J. Audio Eng. Soc*, vol. 45, no. 6, 1997.
[9] J. A. Moorer, "About this reverberation business," *Comput. Music J.*, vol. 3, pp. 13–28, 1979.
[10] T. W. Parks and C. S. Burrus, *Digital filter design*, Wiley, 1987.
[11] R. M. Golden, "Digital filter synthesis by sampled-data transformation," *IEEE Trans. Audio Electroacoust.*, vol. 16, no. 3, 1968.
[12] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, Wiley, 1st ed., 2009.
[13] D. T. Yeh, J. S. Abel, A. Vladimirescu, and J. O. Smith, "Numerical methods for simulation of guitar distortion circuits," *Comput. Music J.*, vol. 32, no. 2, pp. 23–42, 2008.
[14] S.-C. Pei and H.-J. Hsu, "Fractional bilinear transform for analog-to-digital conversion," *IEEE Trans. Signal Process.*, vol. 56, no. 5, pp. 2122–2127, 2008.
[15] B. Lindberg, "On smoothing and extrapolation for the trapezoidal rule," *BIT Numerical Mathematics*, vol. 11, no. 1, pp. 29–52, 1971.
[16] W. Gao, E. Solodovnik, R. Dougal, G. J. Cokkinides, and A. P. S. Meliopoulos, "Elimination of numerical oscillations in power system dynamic simulation," in *Proc. 18th IEEE Appl. Power Electron. Conf. and Expo.*, 2003, vol. 2, pp. 790–4.
[17] T. Needham, *Visual complex analysis*, Oxford Univ. Press, 1998.
[18] B. Hamilton and A. Torin, "Finite difference schemes on hexagonal grids for thin linear plates with finite volume boundaries," in *Proc. Int. Conf. Digital Audio Effects*, 2014, pp. 592–599.
[19] J. Laroche, "On the stability of time-varying recursive filters," *J. Audio Eng. Soc*, vol. 55, no. 6, pp. 460–471, 2007.
[20] J. Macak and J. Schimmel, "Nonlinear circuit simulation using time-variant filter," in *Proc. Int. Conf. Digital Audio Effects*, 2009.

WAVE DIGITAL FILTER ADAPTORS FOR ARBITRARY TOPOLOGIES AND MULTIPORT LINEAR ELEMENTS

Kurt James Werner, Julius O. Smith III, Jonathan S. Abel

Center for Computer Research in Music and Acoustics (CCRMA), Stanford University
660 Lomita Drive, Stanford, CA 94305, USA
[kwerner|jos|abel]@ccrma.stanford.edu

ABSTRACT

We present a Modified-Nodal-Analysis-derived method for developing Wave Digital Filter (WDF) adaptors corresponding to complicated (non-series/parallel) topologies that may include multiport linear elements (e.g. controlled sources and transformers). A second method resolves noncomputable (non-tree-like) arrangements of series/parallel adaptors. As with the familiar 3-port series and parallel adaptors, one port of each derived adaptor may be rendered reflection-free, making it acceptable for inclusion in a standard WDF tree. With these techniques, the class of acceptable reference circuits for WDF modeling is greatly expanded. This is demonstrated by case studies on circuits which were previously intractable with WDF methods: the Bassman tone stack and Tube Screamer tone/volume stage.

1. INTRODUCTION

The Wave Digital Filter (WDF) concept [1] provides an elegant framework for creating digital models of analog reference circuits (or any lumped reference system). However, the class of reference circuits which are tractable with WDF techniques is very small. Specifically, reference circuits with complicated topologies (those that can't be decomposed entirely into series and parallel connections) and/or multiport linear elements (transformers, controlled sources, operational amplifiers, etc.) are *not* accommodated *in general* by known techniques.

In this work, we focus on expanding the class of tractable *linear* reference circuits to include these problematic cases. Although these elements may be accomodated by known techniques in specific configurations, the situations where they are problematic are *not* rare edge cases. For instance, bridged-T networks [2] are commonly present in guitar tone stack circuits [3, 4] and analog drum machine circuits [5–7]. Operational amplifiers are often used in complicated feedback arrangements which thwart WDF modeling except under specific circumstances [8]. This work addresses a need for simple adaptor derivation procedures suitable for *any* topology which may arise in a reference circuit. To that end, we'll emphasize methodical and even automatable "stamp" techniques, well-known in Modified Nodal Analysis (MNA) [9–12].

Researchers have primarily targeted the well-known restriction of WDFs to reference circuits with a single nonlinearity [13]. Since musical circuits in general may contain *many* nonlinearities, physical modeling / virtual analog researchers have focused on trying to extend this result to the case of multiple and multiport nonlinearities. However, we argue that limitations on WDF reference circuit topologies are just as problematic. In a companion paper [14], we'll review treatments of nonlinearities in WDFs and show how the resolution of topological issues presented in this paper enable novel treatments of reference circuits with multiple/multiport nonlinear elements.

In §2, we review relevant previous work. Two novel methods for deriving adaptor structures are given in §§3–4. Case studies on the Fender Bassman tone stack and Tube Screamer tone/volume stage are given in §5. Both circuits have problematic topologies which render them intractable with classical WDF techniques—their simulation demonstrates the success of our methods.

2. PREVIOUS WORK

In the early 1970s, Alfred Fettweis formulated the WDF framework as a technique for designing digital filter structures that mimic the properties of analog reference circuits, which had well-studied behavior and well-established design principles [15, 16]. The analog reference circuits of interest commonly had ladder [17] or lattice [18] structure. Hence, it is not surprising that when Fettweis and Meerkötter formalized the concept of a WDF adaptor, they focused on series and parallel connections [19].

When Fettweis published his omnibus 1986 article "Wave Digital Filters: Theory and Practice" [1], the WDF formalism had reached a high level of maturity. This included extensions to the multidimensional case which led to uses of WDFs as solvers of partial differential equations [20]. Today, in physical modeling [21] and virtual analog, WDFs are an active research area. Researchers in these fields aim to mimic the behavior of musical circuitry, such as guitar amplifiers and effect pedals. WDFs are attractive for this application, but the range of reference circuits that the framework can be applied to is significantly restricted.

Classical WDF methods are applicable *only* to circuits whose topologies can be decomposed into series and parallel connections [19]. Although the significance of this limitation is only rarely acknowledged, it was noted by Martens and Meerkötter as early as 1976 [22]. Fränken *et al.* [23, 24] used formal graph-theoretic methods to generate adaptor structures from circuits with complicated topologies. Their work yields insights about the existence and implications of complicated topologies and ideal transformers [23, 24], though without presenting a method for deriving the actual scattering behavior of the resulting $\mathcal{R}$*-type* adaptors.[1]

Recently, Paiva *et al.* studied operational amplifiers (op-amps) in a WDF context [8], showing how circuits with an op-amp differential amplifier topology can be treated as a feedforward cascade of controlled sources—we can consider this an instance of the "leaf–leaf connection" discussed by De Sanctis and Sarti [26].

[1] Meerkötter and Fränken furthermore propose techniques for factorizing scattering matrices to reduce computational costs [25], but again this cannot be applied in a WDF context without knowledge of the scattering behavior of these adaptors.

Although it is clear that a generalization of their technique would be applicable to some other op-amp topologies, there are cases where the relationships among controlled sources will create inadmissible feedback loops.

Some researchers have shown examples of circuits with complicated topologies which, if we insist on only using classical 3-port series and parallel adaptors, result in non-treelike (and hence noncomputable) *ring structures* [2, 27]. Schwerdtfeger and Kummert studied methods for iteratively resolving such ring structures at runtime, based on contractivity properties of WDF elements [2].

3. METHOD ONE: MNA

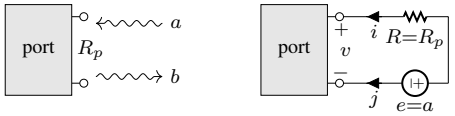
Here we present a novel method, based on graph-theoretic views of WDF adaptor structures, for finding the scattering matrix of a WDF adaptor with arbitrary topology that may include absorbed linear multiport elements. A special case without any linear multiport elements was briefly presented in [28]. Although this may seem a violation of the modularity goal of WDFs, absorbing WDF elements into adaptors to ease realizability issues is as old as WDFs themselves—Fettweis' resistive voltage (current) source [16] can be considered merely a voltage (current) source and a resistor absorbed into a 3-port series (parallel) adaptor.

Given a reference circuit, which may contain linear multiport elements and complicated topologies, the first step of our method is to find a suitable WDF adaptor structure. It is possible to accomplish this by inspection, but more convenient to apply the method of Fränken *et al.* [23, 24]. According to this method, a connected graph representing the reference circuit is formed, where graph nodes correspond to circuit nodes and graph edges correspond to ports in the circuits, i.e., these graph edges correspond to bipole circuit elements (resistors, capacitors, inductors, etc.) or ports of a linear multiport element (transformer, controlled source, etc.).

Then, standard graph separation algorithms which find "split components" are used to decompose the connected graph into a tree structure: the *SPQR tree*. For realizability reasons, we must ensure that these graph separation algorithms will not separate edges that correspond to a multiport circuit element. This can be avoided by pre-processing the graph structure with so-called *replacement graphs* before applying a separation algorithm [24].[2] A minimal suitable replacement graph includes the addition of 3 fictitious nodes to each multiport element, and fictitious edges connecting each one of them to every original node in the multiport element; this is sufficient to ensure that graph separation algorithms will not break apart the multiport linear element.

In the SPQR tree, nodes represent topological entities in the graph with detected split components, and edges represent virtual edges in the split components. $\mathcal{S}$, $\mathcal{P}$, and $\mathcal{R}$ nodes correspond directly to familiar **S**eries adaptors, **P**arallel adaptors, and the less well-known family of **R**igid or "strongly connected" adaptors. $\mathcal{Q}$ nodes correspond to single component ports. This process highlights the fact that series and parallel adaptors are *not* sufficient to represent all linear circuit topologies, not even those without multiport elements. A result of using replacement graphs, multiport linear elements will commonly "clump up" inside of $\mathcal{R}$-type nodes. The idea of sources absorbed into adaptors is not necessarily new [26], but generalizing the concept in light of formal graph decomposition methods is crucial to our approach.

[2] An example is given in §5.2.



(a) *Wave domain.* (b) *Kirchhoff domain.*

Figure 1: *Instantaneous Thévenin port equivalent.*

A suitable WDF adaptor structure follows from the derived SPQR tree. The second step of our method is to find the scattering behavior of each adaptor in the SPQR tree. The scattering behavior of series and parallel adaptors is well-known [19], but the actual scattering behavior of $\mathcal{R}$-type adaptors is not. We require an easy-to-apply technique that yields this scattering behavior for any $\mathcal{R}$-type adaptor, even ones including absorbed multiport linear elements, and that allows us to make one port *reflection-free* to ensure computability of the WDF structure.

Our novel method accomplishes these goals within the framework of the Modified Nodal Analysis (MNA) formalism [9, 10]. Since MNA works in the Kirchhoff domain, we need to form an equivalent circuit that corresponds to adaptor port definitions. We call the circuit which produces a certain incident wave upon a port an *instantaneous Thévenin port equivalent*.[3] It is easily derived by considering the relationships among a voltage source value e, source current j, electrical resistance R, port voltage v, port current i, port resistance R_p, incident wave a, and reflected wave b.

A single WDF port and its instantaneous Thévenin equivalent are shown in Fig. 1. If we set $R = R_p$, the voltage source value must be equal to the incident wave; $e = a$. In every case, we can see that $i = -j$. This is identical to the classic derivation of adaptation criteria for a resistive voltage source [1], but framing it "backwards" shows us how to create a Kirchhoff-domain equivalent to any topological entity including $\mathcal{R}$-type adaptors. Such an equivalent circuit is simply formed by, each port of an adaptor, attaching an instantaneous Thévenin port equivalent, setting the electrical resistance equal to the port resistance, and setting the voltage source value to the incident wave value.

From this equivalent circuit, we need to assemble a MNA system. In general, a MNA system is set up as

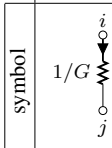
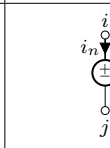
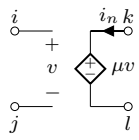
$$\underbrace{\begin{bmatrix}\mathbf{Y} & \mathbf{A}\\ \mathbf{B} & \mathbf{D}\end{bmatrix}}_{\text{MNA matrix } \mathbf{X}} \begin{bmatrix}\boldsymbol{v}_n\\ \boldsymbol{j}\end{bmatrix} = \begin{bmatrix}\boldsymbol{i}_s\\ \boldsymbol{e}\end{bmatrix}, \tag{1}$$

where $\mathbf{X}$ partitions $\mathbf{Y}$, $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{D}$ define the relationship among node voltages $\boldsymbol{v}_n$, voltage source branch currents $\boldsymbol{j}$, current source values $\boldsymbol{i}_s$ and voltage source values $\boldsymbol{e}$ [9, 10]. For $\mathcal{R}$-type adaptors with no linear multiport elements, we will only employ resistor and voltage source stamps, yielding a version of (1) where $\mathbf{B} = \mathbf{A}^{\mathsf{T}}$ and $\mathbf{D} = \mathbf{0}$. In our context, we will also always have $\boldsymbol{i}_s = 0$.

Finding $\mathbf{X}$ by inspection using, e.g., Kirchhoff's Current Law (KCL) is possible, but can be tedious—the use of *element stamps* (sometimes called "MNA templates" [11, 12] or MNA "by inspection" [10]) greatly simplifies this process, even to the point of being automatable. Using element stamps is simple. Every node in

[3] This is related to the *augmented network* described by Belevitch [29] and has also been called the *Wave Equivalent Thevenin Source* in microstrip engineering [30].

Table 1: *Modified Nodal Analysis element stamps.*

	resistor	voltage source	VCVS
symbol			
stamp	$\begin{matrix} & i & j \\ i & G & -G \\ j & -G & G \end{matrix}$	$\begin{matrix} & i & j & n & \\ i & & & 1 & \\ j & & & -1 & \\ \text{next} & 1 & -1 & & \end{matrix}$ source $\begin{matrix} \\ \\ \overline{E} \\ \\ \end{matrix}$	$\begin{matrix} & i & j & k & l & n & \\ k & & & & & 1 & \\ l & & & & & -1 & \\ \text{next} & -\mu & \mu & 1 & -1 & & \end{matrix}$

the equivalent circuit is assigned an index, and then the contribution of each element is added into $\mathbf{X}$ one by one according to the element stamps. A fine point of this process is that one node in the equivalent circuit is chosen as the "datum" node, and neither its row nor its column appear in $\mathbf{X}$. Since the number of independent KCL equations in a circuit is always one less than the number of nodes [31], $\mathbf{X}$ would always be singular without this step.

Element stamps corresponding to a resistor, a voltage source, and a voltage-controlled voltage source are given in Table 1. Other stamps, corresponding to, e.g., transformers and other controlled sources, are given in the MNA literature [9–11].

A populated MNA system can be used to derive the scattering behavior of an $\mathcal{R}$-type adaptor. Recall the standard WDF voltage wave definition [1], applied to a vector of ports,

$$\boldsymbol{a} = \boldsymbol{v} + \mathbf{R}_p \boldsymbol{i} \quad \text{and} \quad \boldsymbol{b} = \boldsymbol{v} - \mathbf{R}_p \boldsymbol{i}\,, \tag{2}$$

with vectors of incident and reflected waves $\boldsymbol{a}$ and $\boldsymbol{b}$, and a diagonal matrix of port resistances $\mathbf{R}_p$. Combining (2) yields

$$\boldsymbol{b} = \boldsymbol{a} - 2\mathbf{R}_p \boldsymbol{i}\,. \tag{3}$$

Recall that in forming our equivalent circuit we imposed

$$\boldsymbol{e} = \boldsymbol{a}\,, \quad \mathbf{R} = \mathbf{R}_p\,, \quad \text{and} \quad \boldsymbol{i} = -\boldsymbol{j}\,. \tag{4}$$

Inverting $\mathbf{X}$ from (1) allows us to solve for $\boldsymbol{j}$ in terms of $\boldsymbol{e}$:

$$\boldsymbol{j} = \begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix} \mathbf{X}^{-1} \begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}^{\mathsf{T}} \boldsymbol{e}\,. \tag{5}$$

Combining (3), (4), and (5), yields

$$\boldsymbol{b} = \mathbf{S}\boldsymbol{a}\,, \quad \text{with} \quad \mathbf{S} = \mathbf{I} + 2\begin{bmatrix}\mathbf{0} & \mathbf{R}\end{bmatrix} \mathbf{X}^{-1} \begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}^{\mathsf{T}}\,. \tag{6}$$

This method is simple to apply and yields a scattering matrix even for adaptors which have absorbed multiport linear elements. In a WDF, we always need to be able to ensure that the port facing towards the root of the tree is reflection free, i.e., the reflected wave at that port does not depend instantaneously on the incident wave at that port. This is accomplished just as in the traditional series and parallel cases. For a port n that we must adapt, we simply solve for the value of R_n which accomplishes $s_{nn} = 0$, where s_{nn} is the diagonal entry of $\mathbf{S}$ corresponding to the contribution of incident wave a_n to reflected wave b_n.

We stress that the adaptors resulting from this process are fully compatible with a standard WDF *tree* framework. It is well-known that N-port series (parallel) adaptors can always be decomposed into $N-2$ cascaded 3-port series (parallel) adaptors [26]. This property means that each adaptor in a Binary Connection Tree (BCT) [26, 32, 33] always has one parent and two children. However, $\mathcal{R}$-type adaptors cannot be decomposed into smaller adaptors and have $N \geq 6$ ports. Hence they have $N-1 \geq 5$ children and a connection tree including them can no longer be assumed binary. To avoid a loss of generality for circuits with $\mathcal{R}$-type adaptors, we drop the "Binary" from the BCT concept, calling it rather the "Connection Tree" (CT)—this does not require any further alteration to standard WDF theory or terminology.

Table 2: $\mathbf{S}_{rings}$ *stamps for series/parallel adaptors.*

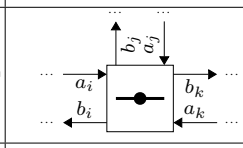
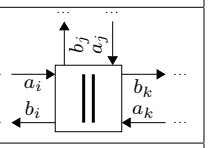
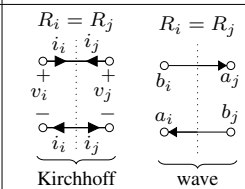
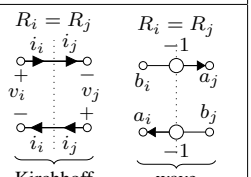
	series connection	parallel connection
adaptor		
stamp	$\begin{matrix} & i & j & k \\ i & 1-\gamma_i & -\gamma_i & -\gamma_i \\ j & -\gamma_j & 1-\gamma_j & -\gamma_j \\ k & -\gamma_k & -\gamma_k & 1-\gamma_k \end{matrix}$ $\gamma_m = \frac{2R_m}{(R_i+R_j+R_k)}$	$\begin{matrix} & i & j & k \\ i & \delta_i-1 & \delta_j & \delta_k \\ j & \delta_i & \delta_j-1 & \delta_k \\ k & \delta_i & \delta_j & \delta_k-1 \end{matrix}$ $\delta_m = \frac{2G_m}{(G_i+G_j+G_k)}$

Table 3: $\mathbf{C}$ *stamps for port compatibility.*

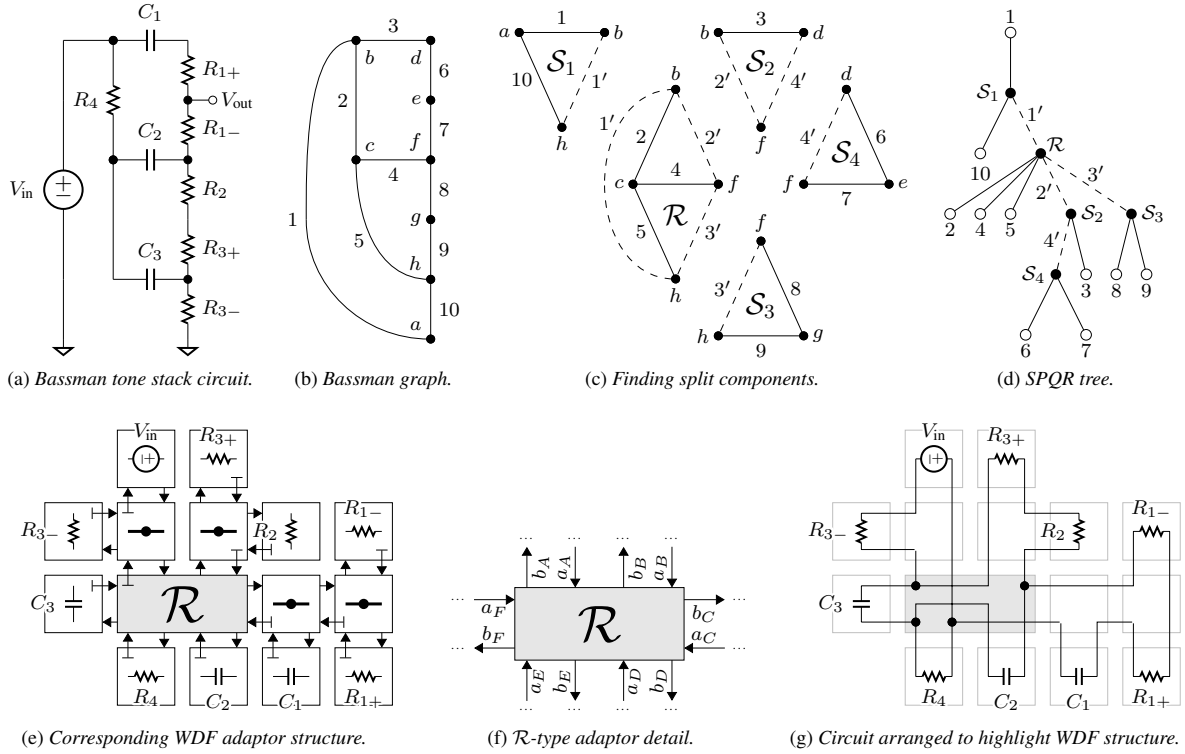
	direct connection	inverse connection
ports		
stamp	$\begin{matrix} & i & j \\ i & 0 & 1 \\ j & 1 & 0 \end{matrix}$	$\begin{matrix} & i & j \\ i & 0 & -1 \\ j & -1 & 0 \end{matrix}$

4. METHOD TWO: "RING" RESOLUTION

The method presented in §3 is simple and systematic and should be applicable to deriving the scattering behavior of *any* WDF adaptor, including hitherto intractable $\mathcal{R}$-type adaptors with or without absorbed multiport linear elements.

In this section, we present an alternate derivation which combines non-tree-like arrangements of standard 3-port series and parallel adaptors which are occasionally seen in the WDF literature [2, 27] into one larger $\mathcal{R}$-type adaptor. These are usually noncomputable since they violate the assumption of a tree structure and hence contain delay-free loops, though recent work achieves guaranteed convergence under runtime iteration [2]. As before, we can render one of the ports in this adaptor reflection-free, making it suitable for inclusion in a standard WDF tree.

We denote the incident and reflected waves at ports that are *internal* to the noncomputable network as $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ and those that are *external* as our normal wave variables $\boldsymbol{a}$ and $\boldsymbol{b}$ as before. Internal ports face other adaptors within the noncomputable network and external ports face the rest of the WDF structure. The whole



(a) *Bassman tone stack circuit.* (b) *Bassman graph.* (c) *Finding split components.* (d) *SPQR tree.*

(e) *Corresponding WDF adaptor structure.* (f) *$\mathcal{R}$-type adaptor detail.* (g) *Circuit arranged to highlight WDF structure.*

Figure 2: *Deriving a WDF adaptor structure for the Fender Bassman tone stack, as in §3.*

noncomputable structure is described by the scattering description

$$\begin{bmatrix} \boldsymbol{b}_i \\ \boldsymbol{b} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix}}_{\mathbf{S}_{\text{rings}}} \begin{bmatrix} \boldsymbol{a}_i \\ \boldsymbol{a} \end{bmatrix} . \tag{7}$$

From this description, we need to find $\mathbf{S}$ as in (6). It is not difficult to populate the $\mathbf{S}_{\text{rings}}$ partitions $\mathbf{S}_{11}$, $\mathbf{S}_{12}$, $\mathbf{S}_{21}$, and $\mathbf{S}_{22}$. Since (7) describes the behavior of an interconnected network of *standard* 3-port series and parallel adaptors (albeit a noncomputable one), the *local* scattering behavior is known. Recall the scattering matrices for unconstrained 3-port series and parallel adaptors, with three ports i, j, and k, incident waves a_i, a_j, and a_k, and reflected waves b_i, b_j, and b_k. The scattering at these adaptors is described by resistance and conductance ratios γ_m and δ_m [26, 34].[4] Hence $\mathbf{S}_{11}$, $\mathbf{S}_{12}$, $\mathbf{S}_{21}$, and $\mathbf{S}_{22}$ can be populated by inspection. Taking inspiration from the *Wave Tableau* (WT) technique [33], and to match the simplicity of the MNA element stamp method presented in §3, we introduce a two-part stamp method for populating these matrices. We note that the WT technique is applied globally, while this method leverages graph-theoretic perspectives [24] to restrict the tableau to a minimal subtree. After assigning each port a numerical index, the stamps shown in Table 2 are used to populate (7).

At the internal ports, a port *C*ompatibility matrix $\mathbf{C}$ describes the relationship between $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$:

$$\boldsymbol{a}_i = \mathbf{C}\boldsymbol{b}_i . \tag{8}$$

[4] Recall that conductance is the reciprocal of resistance: $G_m = 1/R_m$.

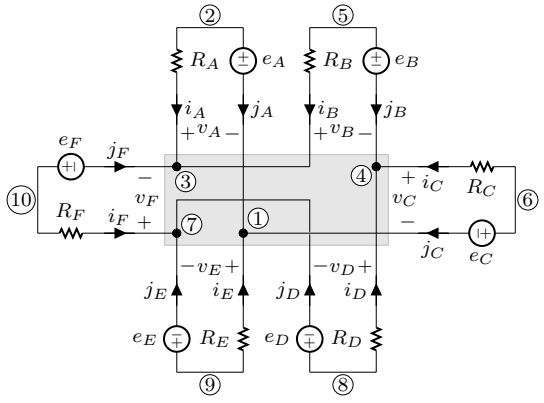


Figure 3: *Instantaneous Thévenin port equivalent to Fig. 2f.*

$\mathbf{C}$ can also be populated by inspection; every internal incident wave is equal to the reflected wave at the port it is connected to, sometimes with a sign inversion if the polarities at that port don't match. $\mathbf{C} = \mathbf{C}^{\mathsf{T}}$ on account of the reciprocity of port connections. We abstract this process into another stamp procedure. Stamps for direct connections and inverse connections[5] are shown in Table 3.

[5] Sometimes known as "null" connections [35], this can also be considered the case of an adapted ($R_i = R_j$) 2-port series adaptor [1].

$$\mathbf{X} = \begin{bmatrix} \mathbf{Y} & \mathbf{A} \\ \mathbf{B} & \mathbf{D} \end{bmatrix} =$$

	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	A	B	C	D	E	F
②	G_A	$-G_A$	0	0	0	0	0	0	0	1	0	0	0	0	0
③	$-G_A$	G_A+G_B	0	$-G_B$	0	0	0	0	0	0	0	0	0	0	-1
④	0	0	G_C+G_D	0	$-G_C$	0	$-G_D$	0	0	0	-1	0	0	0	0
⑤	0	$-G_B$	0	G_B	0	0	0	0	0	0	1	0	0	0	0
⑥	0	0	$-G_C$	0	G_C	0	0	0	0	0	0	1	0	0	0
⑦	0	0	0	0	0	G_F	0	0	$-G_F$	0	0	0	-1	-1	0
⑧	0	0	$-G_D$	0	0	0	G_D	0	0	0	0	0	1	0	0
⑨	0	0	0	0	0	0	0	G_E	0	0	0	0	0	1	0
⑩	0	0	0	0	0	$-G_F$	0	0	G_F	0	0	0	0	0	1
A	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0
E	0	0	0	0	0	-1	0	1	0	0	0	0	0	0	0
F	0	-1	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 4: *Bassman MNA system matrix—example resistor stamp in light shading, example voltage source stamp in dark shading.*

Rearranging (8) as $\boldsymbol{b}_i = \mathbf{C}^{-1}\boldsymbol{a}_i$ and substituting into the top set of equations in (7) yields

$$\mathbf{C}^{-1}\boldsymbol{a}_i = \mathbf{S}_{11}\boldsymbol{a}_i + \mathbf{S}_{12}\boldsymbol{a}\,. \tag{9}$$

Solving for $\boldsymbol{a}_i$ yields

$$\boldsymbol{a}_i = (\mathbf{C}^{-1} - \mathbf{S}_{11})^{-1}\mathbf{S}_{12}\boldsymbol{a}\,. \tag{10}$$

Substituting into the bottom set of equations of (7) yields

$$\mathbf{S} = \mathbf{S}_{21}(\mathbf{C}^{-1} - \mathbf{S}_{11})^{-1}\mathbf{S}_{12} + \mathbf{S}_{22}\,. \tag{11}$$

When it can be applied, this method gives identical results to §3. An advantage is that it does not require any recourse to graph theory. A disadvantage is that noncomputable adaptor structures only arise from inspection—there is no known systematic procedure for generating them. Unlike §3, this technique does not support cases involving multiport linear circuit elements.

5. CASE STUDIES

The results of this paper enable us to model linear circuits which would previously have been off-limits as reference circuits for a WDF. We present two detailed tutorial examples demonstrating our techniques: the tone stack from the Fender Bassman amp and the tone/volume stage from the Tube Screamer distortion pedal.

5.1. Fender Bassman Tone Stack

As a first example, we'll study the tone stack from the Fender Bassman amp[6] (Fig. 2a), using each of the two methods derived in §§3–4. Yeh and Smith studied the Bassman tone stack by deriving its transfer function [4]. Although they suggested that it could potentially be implemented as a WDF, that would have been a daunting task at the time, since the Bassman tone stack circuit (Fig. 2a) can't be decomposed into a tree of series and parallel adaptors. Until now simulation of this circuit as a WDF would require the use of component consolidation [3] or topological transformations such as the Y–Δ ("wye–delta") transformation [36].

What follows is a step by step walkthrough of how §3 can be applied to model this circuit. First a graph representing this

[6] R_1, R_2, and R_3 are high, low, and mid tone control potentiometers.

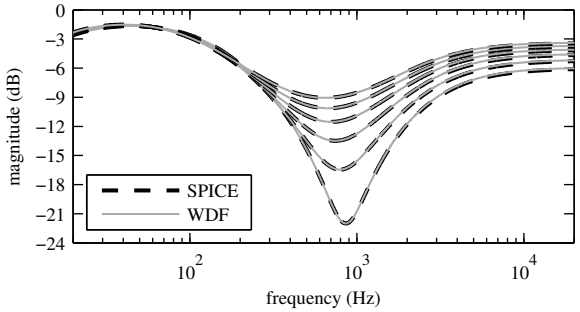
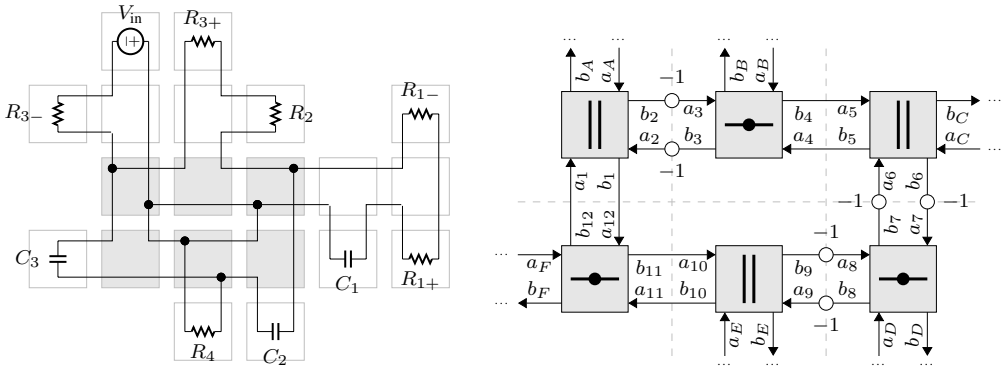


Figure 5: *Bassman magnitude responses, low and high potentiometers at* 50%*, mid potentiometer at* 20% *increments.*

circuit is formed (Fig. 2b). In this graph, nodes correspond to circuit nodes and are each assigned a lowercase letter. Graph edges correspond to ports in the circuit and are each assigned an arabic numeral. For this graph representation, we follow the procedure of Fränken *et al.* [24] to find split components (Fig. 2c). This yields four series connections and a 6-port $\mathcal{R}$-type connection. We designate the voltage source (edge 1) as the root of the tree for realizability reasons—an ideal voltage source cannot be adapted and must be a root element. From here, an SPQR tree can be formed (Fig. 2d). A WDF adaptor structure follows by identity from this SPQR tree (Fig. 2e). This adaptor structure contains one $\mathcal{R}$-type adaptor, corresponding to the $\mathcal{R}$-type connection in Fig. 2c. Incident and reflected wave labeling details (each port is assigned a lowercase letter) of this problematic adaptor are shown in detail in Fig. 2f, and a rearranged version of Fig. 2a which highlights the derived adaptor structure is shown in Fig. 2g.

We form an equivalent circuit to Fig. 2f by attaching instantaneous Thévenin port equivalents (Fig. 3). Each node is assigned a circled arabic numeral. Using an MNA stamp (Table 1) for each voltage source and resistor yields the system matrix $\mathbf{X}$ (Fig. 4).

Each stamp contributes to certain entries in the system matrix and source vector. That is, multiple stamps can contribute to the same system matrix entry. For instance, we can see this at play at matrix entries $(③, ③)$ and $(④, ④)$ in Fig. 4.



(a) *Circuit arranged using only series/parallel connections.* (b) *Detail on noncomputable system of series/parallel adaptors.*

Figure 6: *Considering Bassman tone stack with only series/parallel adaptors, as in §4.*

Plugging Fig. 4 into (6) yields the scattering matrix $\mathbf{S}$ of this adaptor. Since port A is facing towards the root of the WDF tree, it must be rendered reflection-free. This is accomplished by solving for the value of R_A which sets $s_{11} = 0$.

To verify this model, we compare a family of magnitude response curves to "ground truth" SPICE simulations (Fig. 5). We find the WDF magnitude responses by taking the FFT of an impulse response that has sufficiently decayed to zero. The WDF magnitude response shows a close correspondence to SPICE, except for the expected frequency warping at high frequencies, a known property of the bilinear transform (BLT) [21].[7]

We can also use the Bassman tone stack as an example of §4. Rather than using the SPQR technique of Fränken *et al.*, we create (by inspection) an adaptor structure that only uses series and parallel adaptors (Fig. 6a). This is similar to Fig. 2e, but the $\mathcal{R}$ node of Fig. 2e has been replaced by a noncomputable network of series and parallel adaptors, shown in detail in Fig. 6b.

Denoting the length-6 vectors of external incident and reflected waves as $\boldsymbol{a} = [a_A, \ldots, a_F]^\mathsf{T}$ and $\boldsymbol{b} = [b_A, \ldots, b_F]^\mathsf{T}$ and the length-12 vectors of internal incident and reflected waves as $\boldsymbol{a}_i = [a_1, \ldots, a_{12}]^\mathsf{T}$ and $\boldsymbol{b}_i = [b_1, \ldots, b_{12}]^\mathsf{T}$, we apply the stamp procedures of §4. The matrix resulting from the stamp procedure (Table 2) is given in Fig. 7. The compatibility matrix (8) relating $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ is found according to the next stamp procedure (Table 3):

$$\mathbf{C} = \begin{array}{c} \begin{array}{cccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{array} \\ \left[\begin{array}{cccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array}. \quad (12)$$

Here light shading indicates one example of an inverse connection and dark shading indicates one example of a direct connection. Plugging Fig. 7 and (12) into (11) yields an identical $\mathbf{S}$ to that obtained by the previous method.

[7] `https://ccrma.stanford.edu/~jos/pasp/Bilinear_Transformation.html`

5.2. Tube Screamer Tone/Volume Stage

As a second case study, we derive a WDF from the tone/volume stage of the Tube Screamer distortion pedal.[8] This derivation is similar to §5.1, although somewhat complicated by the op-amp. In addition to showing a case where an adaptor "absorbs" a linear multiport element, this demonstrates how these techniques can be used for advanced and general op-amp modeling. Unlike [8], this technique is not limited to differential amplifier arrangements.

In the circuit (Fig. 8a), the op-amp is treated as ideal, i.e., as a voltage-controlled voltage source that relates the output voltage to the differential input voltage $v_+ - v_-$ by a large open-loop gain A_{OL}. A graph (Fig. 8b) is formed by the *replacement graph* method of Fränken *et al.* [24]; nameless replacement graph nodes and edges are indicated in gray. Detail on the resulting $\mathcal{R}$-type topology is shown in Fig. 8c; remaining standard series and parallel structures are not shown. As before, the split component search yields an SPQR tree (Fig. 8d) and corresponding WDF adaptor structure (Fig. 8e). Notice in Fig. 8f that the controlled source has been absorbed into the $\mathcal{R}$-type adaptor. The $\mathcal{R}$-type adaptor scattering matrix derivation is done according to §3.

To verify this model, we compare a family of magnitude response curves to "ground truth" SPICE simulations (Fig. 9). The WDF magnitude response shows an excellent correspondence to SPICE, except for BLT frequency warping as before.

6. CONCLUSION

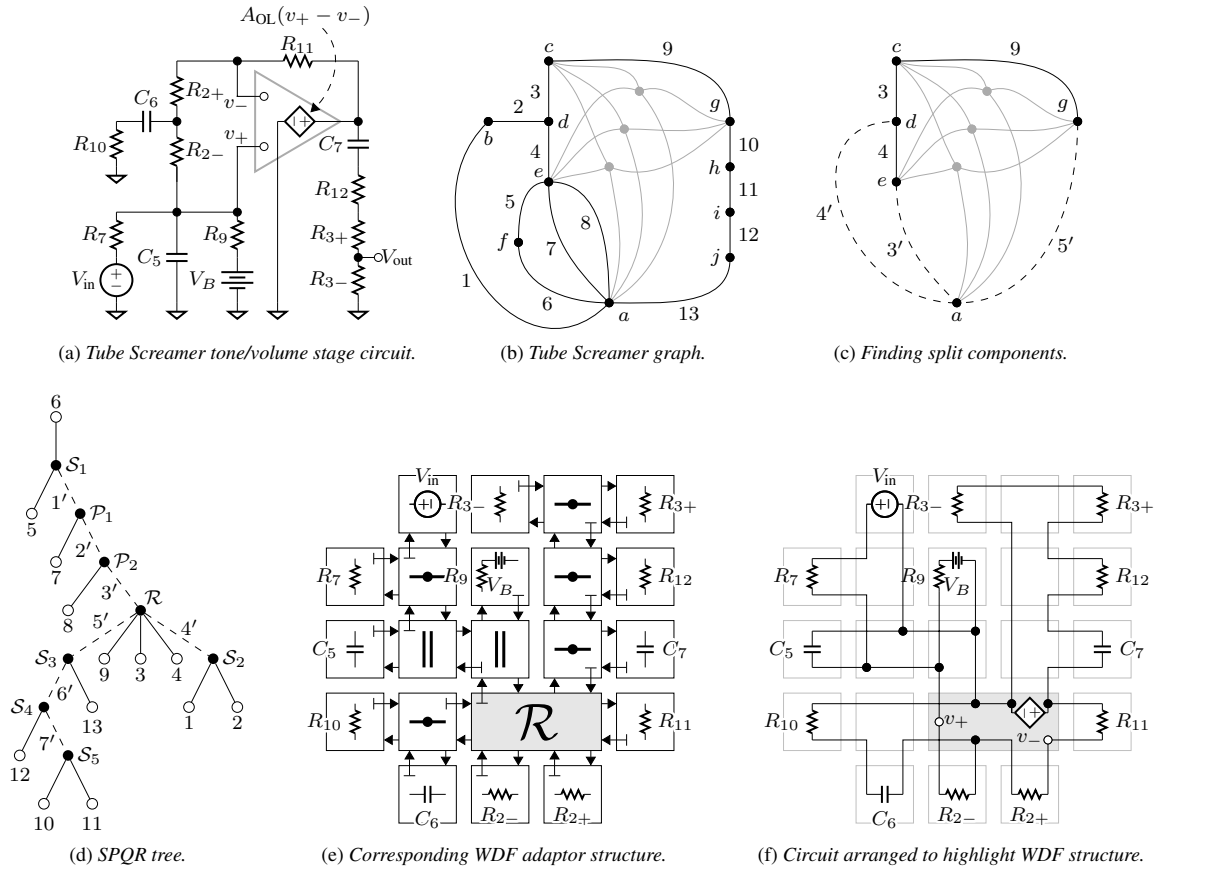
We presented a method (§3) that leverages Modified Nodal Analysis to derive the scattering behavior of WDF adaptors with complicated topologies and even absorbed multiport linear elements. This method is applicable to reference circuits involving any multi-port linear elements with MNA descriptions, e.g., voltage- or current-controlled voltage or current sources and transformers. A second method (§4), applicable to non-tree-like combinations of series and parallel adaptors, involves algebraic wave domain manipulations. Both are based on simple stamp philosophies and yield adaptors which are suitable for inclusion in standard WDF trees. Combining these methods with the graph-theoretic insights of Fränken *et al.* brings a high degree of generality to linear WDFs.

[8] R_2 and R_3 are tone and volume control potentiometers.

$$\mathbf{S}_{\text{rings}} = \left[\begin{array}{c|c} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \hline \mathbf{S}_{21} & \mathbf{S}_{22} \end{array}\right] = \ldots$$

	1	2	3	4	5	6	7	8	9	10	11	12	A	B	C	D	E	F
1	δ_1-1	δ_2	0	0	0	0	0	0	0	0	0	0	δ_A	0	0	0	0	0
2	δ_1	δ_2-1	0	0	0	0	0	0	0	0	0	0	δ_A	0	0	0	0	0
3	0	0	$1-\gamma_3$	$-\gamma_3$	0	0	0	0	0	0	0	0	0	$-\gamma_3$	0	0	0	0
4	0	0	$-\gamma_4$	$1-\gamma_4$	0	0	0	0	0	0	0	0	0	$-\gamma_4$	0	0	0	0
5	0	0	0	0	δ_5-1	δ_6	0	0	0	0	0	0	0	0	δ_C	0	0	0
6	0	0	0	0	δ_5	δ_6-1	0	0	0	0	0	0	0	0	δ_C	0	0	0
7	0	0	0	0	0	0	$1-\gamma_7$	$-\gamma_7$	0	0	0	0	0	0	0	$-\gamma_7$	0	0
8	0	0	0	0	0	0	$-\gamma_8$	$1-\gamma_8$	0	0	0	0	0	0	0	$-\gamma_8$	0	0
9	0	0	0	0	0	0	0	0	δ_9-1	δ_{10}	0	0	0	0	0	0	δ_E	0
10	0	0	0	0	0	0	0	0	δ_9	$\delta_{10}-1$	0	0	0	0	0	0	δ_E	0
11	0	0	0	0	0	0	0	0	0	0	$1-\gamma_{11}$	$-\gamma_{11}$	0	0	0	0	0	$-\gamma_{11}$
12	0	0	0	0	0	0	0	0	0	0	$-\gamma_{12}$	$1-\gamma_{12}$	0	0	0	0	0	$-\gamma_{12}$
A	δ_1	δ_2	0	0	0	0	0	0	0	0	0	0	δ_A-1	0	0	0	0	0
B	0	0	$-\gamma_B$	$-\gamma_B$	0	0	0	0	0	0	0	0	0	$1-\gamma_B$	0	0	0	0
C	0	0	0	0	δ_5	δ_6	0	0	0	0	0	0	0	0	δ_C-1	0	0	0
D	0	0	0	0	0	0	$-\gamma_D$	$-\gamma_D$	0	0	0	0	0	0	0	$1-\gamma_D$	0	0
E	0	0	0	0	0	0	0	0	δ_9	δ_{10}	0	0	0	0	0	0	δ_E-1	0
F	0	0	0	0	0	0	0	0	0	0	$-\gamma_F$	$-\gamma_F$	0	0	0	0	0	$1-\gamma_F$

Figure 7: *Bassman loop resolution matrix—example parallel stamp in light shading, example series stamp in dark shading.*



(a) *Tube Screamer tone/volume stage circuit.* (b) *Tube Screamer graph.* (c) *Finding split components.*

(d) *SPQR tree.* (e) *Corresponding WDF adaptor structure.* (f) *Circuit arranged to highlight WDF structure.*

Figure 8: *Deriving a WDF adaptor structure for the Tube Screamer tone/volume stage, as in §3.*

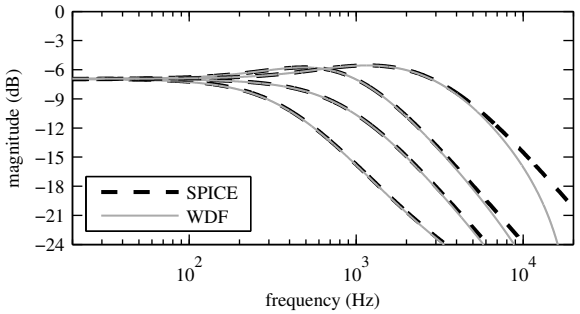


Figure 9: *Tube Screamer magnitude responses, volume potentiometer at* 50%*, tone potentiometer at* 0%*,* 50%*,* 95%*, and* 100%*.*

In general, adapted n-port scattering matrices involve $n^2 - 1$ multiplies. One-multiplier realizations of 2- and 3-port adaptors are well known [1, 3]. Even without exploiting any structural knowledge of scattering matrices, it may be possible to reduce realization cost with matrix factorization [25]. Future work should seek canonic realizations of the adaptors presented in this paper.

We've focused on lingering issues of topology in WDF models of *linear* reference circuits. The presented methods are applicable to the same situations in nonlinear circuits. Computability concerns greatly heighten topological issues for WDF models of reference circuits with multiple/multiport nonlinearities. A companion paper [14] expands on the initial perspective of [28] and shows how the methods in this paper can yield a novel framework for considering WDFs with multiple/multiport nonlinear elements.

7. REFERENCES

[1] A. Fettweis, "Wave digital filters: Theory and practice," *Proc. IEEE*, vol. 74, no. 2, pp. 270–327, 1986.

[2] T. Schwerdtfeger and A. Kummert, "A multidimensional signal processing approach to wave digital filters with topology-related delay-free loops," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Florence, Italy, May 4–9 2014, pp. 389–393.

[3] M. Karjalainen, "Efficient realization of wave digital components for physical modeling and sound synthesis," *IEEE Trans. Audio, Speech, Language Process.*, vol. 16, no. 5, pp. 947–956, 2008.

[4] D. T.-M. Yeh and J. O. Smith III, "Discretization of the '59 Fender Bassman tone stack," in *Proc. Int. Conf. Digital Audio Effects (DAFx-06)*, Montréal, Canada, Sept. 18–20 2006.

[5] K. J. Werner et al., "A physically-informed, circuit-bendable, digital model of the Roland TR-808 bass drum circuit," in *Proc. Int. Conf. Digital Audio Effects*, Erlangen, Germany, Sept. 1–5 2014, vol. 17.

[6] K. J. Werner et al., "The TR-808 cymbal: a physically-informed, circuit-bendable, digital model," in *Proc. Int. Comput. Music / Sound Music Comput. Conf.*, Athens, Greece, Sept. 14–20 2014, vol. 40/11.

[7] K. J. Werner et al., "More cowbell: a physically-informed, circuit-bendable, digital model of the TR-808 cowbell," in *Proc. Int. Audio Eng. Soc. (AES) Conv.*, Los Angeles, CA, Oct. 9–12 2014, vol. 137.

[8] R. C. D. Paiva et al., "Emulation of operational amplifiers and diodes in audio distortion circuits," *IEEE Trans. Circuits Syst. II: Expr. Briefs*, vol. 59, no. 10, pp. 688–692, 2012.

[9] C.-W. Ho et al., "The modified nodal approach to network analysis," *IEEE Trans. Circuits Syst*, vol. 22, no. 6, pp. 504–509, 1975.

[10] J. Vlach and K. Singhal, *Computer methods for circuit analysis and design*, Springer, 1983.

[11] D. T.-M. Yeh et al., "Automated physical modeling of nonlinear audio circuits for real-time audio effects—part I: Theoretical development," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 4, pp. 728–737, 2010.

[12] D. T.-M. Yeh, "Automated physical modeling of nonlinear audio circuits for real-time audio effects—part II: BJT and vacuum tube examples," *IEEE Trans. Audio, Speech, Language Process.*, vol. 20, no. 4, pp. 1207–1216, 2012.

[13] K. Meerkötter and R. Scholz, "Digital simulation of nonlinear circuits by wave digital filter principles," in *Proc. IEEE Int. Symp. Circuits Syst.*, June 1989, vol. 1, pp. 720–723.

[14] K. J. Werner et al., "Resolving wave digital filters with multiple/multiport nonlinearities," in *Proc. Int. Conf. Digital Audio Effects (DAFx-15)*, Trondheim, Norway, Nov. 30 – Dec. 3 2015.

[15] A. Fettweis, "Some principles of designing digital filters imitating classical filter structures," *IEEE Trans. Circuit Theory*, vol. 18, no. 2, pp. 314–316, 1971.

[16] A. Fettweis, "Digital filters structures related to classical filter networks," *Archiv Elektronik Übertragungstechnik (AEÜ)*, vol. 25, pp. 79–89, 1971.

[17] A. Sedlmeyer and A. Fettweis, "Digital filters with true ladder configuration," *Int. J. Circuit Theory Appl.*, vol. 1, pp. 5–10, 1973.

[18] A. Fettweis et al., "Wave digital lattice filters," *Int. J. Circuit Theory Appl.*, vol. 2, pp. 203–211, 1974.

[19] A. Fettweis and K. Meerkötter, "On adaptors for wave digital filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 23, no. 6, 1975.

[20] S. Bilbao, *Wave and Scattering Methods for Numerical Simulation*, John Wiley and Sons, New York, July 2004.

[21] J. O. Smith III, *Physical Audio Signal Processing for Virtual Musical Instruments and Audio Effects*, online book, 2010 edition.

[22] G. O. Martens and K. Meerkötter, "On N-port adaptors for wave digital filters with application to a bridged-tee filter," in *Proc IEEE Int. Symp. Circuits Syst.*, Munich, Germany, Apr. 1976, pp. 514–517.

[23] D. Fränken et al., "Generation of wave digital structures for connection networks containing ideal transformers," in *Proc. Int. Symp. Circuits Syst. (ISCAS '03)*, May 25–28 2003, vol. 3, pp. 240–243.

[24] D. Fränken et al., "Generation of wave digital structures for networks containing multiport elements," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 52, no. 3, pp. 586–596, 2005.

[25] K. Meerkötter and D. Fränken, "Digital realization of connection networks by voltage-wave two-port adaptors," *Archiv Elektronik Übertragungstechnik (AEÜ)*, vol. 50, no. 6, pp. 362–367, 1996.

[26] G. De Sanctis and A. Sarti, "Virtual analog modeling in the wave-digital domain," *IEEE Tran. Audio, Speech, Language Process.*, vol. 18, no. 4, pp. 715–727, 2010.

[27] G. De Sanctis, "Una nuova metodologia per l'implementazione automatica di strutture ad onda numerica orientata alla modellazione ad oggetti di interazioni acustiche," M.S. thesis, Politecnico di Milano, Italy, 2002.

[28] K. J. Werner et al., "A general and explicit formulation for wave digital filters with multiple/multiport nonlinearities and complicated topologies," in *Proc. IEEE Workshop Appl. Sig. Process. Audio Acoust. (WASPAA)*, New Paltz, NY, Oct. 18–21 2015.

[29] V. Belevitch, *Classical network theory*, San Francisco, CA, 1968.

[30] B. P. Stošić and M. V. Gmitrović, "Equivalent Thevenin source method as tool for response calculation of wave digital structures," in *Proc. Int. Conf. Telecommun. Modern Satellite, Cable, Broadcast. Services*, Niš, Serbia, Sept. 26–28 2007, vol. 8, pp. 203–206.

[31] S. Seshu and M. B. Reed, *Linear graphs and electrical networks*, Addison–Wesley, Reading, MA, 1961.

[32] G. De Sanctis et al., "Automatic synthesis strategies for object-based dynamical physical models in musical acoustics," in *Proc. Int. Conf. Digital Audio Effects*, London, UK, Sept. 8–11 2003, vol. 6.

[33] A. Sarti and G. De Sanctis, "Systematic methods for the implementation of nonlinear wave-digital structures," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 56, no. 2, pp. 460–472, 2009.

[34] S. D'Angelo, *Virtual Analog Modeling of Nonlinear Musical Circuits*, Ph.D. diss., Aalto University, Espoo, Finland, Sept. 2014.

[35] S. D'Angelo and V. Välimäki, "Wave-digital polarity and current inverters and their application to virtual analog audio processing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Kyoto, Japan, Mar. 2012, pp. 469–472.

[36] D. T.-M. Yeh, "Tutorial on wave digital filters," `https://ccrma.stanford.edu/~dtyeh/papers/wdftutorial.pdf`, Jan. 25 2008.

RESOLVING WAVE DIGITAL FILTERS WITH MULTIPLE/MULTIPORT NONLINEARITIES

Kurt James Werner, Vaibhav Nangia, Julius O. Smith III, Jonathan S. Abel

Center for Computer Research in Music and Acoustics (CCRMA), Stanford University
660 Lomita Drive, Stanford, CA 94305, USA
[kwerner|vnangia|jos|abel]@ccrma.stanford.edu

ABSTRACT

We present a novel framework for developing Wave Digital Filter (WDF) models from reference circuits with multiple/multiport nonlinearities. Collecting all nonlinearities into a vector at the root of a WDF tree bypasses the traditional WDF limitation to a single nonlinearity. The resulting system has a complicated scattering relationship between the nonlinearity ports and the ports of the rest of the (linear) circuit, which can be solved by a Modified-Nodal-Analysis-derived method. For computability reasons, the scattering and vector nonlinearity must be solved jointly; we suggest a derivative of the K-method. This novel framework significantly expands the class of appropriate WDF reference circuits. A case study on a clipping stage from the Big Muff Pi distortion pedal involves both a transistor and a diode pair. Since it is intractable with standard WDF methods, its successful simulation demonstrates the usefulness of the novel framework.

1. INTRODUCTION

The Wave Digital Filter (WDF) concept [1] has been used extensively in physical modeling [2] and virtual analog [3–5]. Researchers in these fields aim to create digital simulations that mimic the physics of reference systems such as guitar amplifiers and effect pedals. Musical circuits of interest may have *many* nonlinearities (diodes, transistors, triodes, etc.) to which desirable sonic qualities are commonly ascribed [6]. Since WDFs natively support only one nonlinear (NL) circuit element [7], the class of reference circuits which can be modeled by WDFs is very limited.

Though nonlinearity handling in WDFs is an active research area [7–47], creating a WDF from any reference circuit is not a solved problem. Known WDF techniques do not accommodate circuits with multiple/multiport nonlinearities in general.

In this work, we focus on expanding the range of tractable *nonlinear* reference circuits to include circuits with multiple/multiport nonlinearities. We present a novel framework which is general enough to accommodate circuits with any topology and any number of nonlinearities with any number of ports each.[1] Building on the work of Fränken *et al.* [48, 49], we emphasize *topological aspects* of the problem and our solution. Rather than treating special cases, this framework treats the common topological issues surrounding multiple and multiport nonlinearities as *the norm*.

In §2, previous work on WDF nonlinearities is discussed. Our framework is presented in §§3–4; a review of the procedure for deriving system matrices is given in §5. A case study is given in §6. §7 concludes, discusses limitations of our treatment, and speculates about future research directions.

[1] This perspective is given in abbreviated form in [46].

2. PREVIOUS WORK

The first generation of WDF research [1, 50–54] was concerned primarily with creating digital filter structures from analog prototypes, hence leveraging existing analog filter design principles. In 1989, Meerkötter and Scholz noticed that the remaining degree of freedom in a linear WDF could be applied towards including a single algebraic nonlinearity, presenting case studies involving an ideal diode and a piecewise linear resistor [7]. Felderhoff expanded this treatment to reference circuits with a single nonlinear capacitor or inductor by considering WDF counterparts to *mutators* from classical circuit theory [8, 9]. This thread was continued by Sarti and De Poli, who formalized nonlinear reactance handling and studied adaptors "with memory" [10, 11]. At the same time, WDFs were being used in physical models of *mechanical* systems—De Sanctis *et al.* studied nonlinear frictions and stiffnesses [18] while Pedersini *et al.* [12, 13] and Bilbao *et al.* [15, 16] studied nonlinear musical applications such as piano hammers and reeds. Under the umbrella of *block-based modeling*, WDFs found continued application in modeling parts of mechanical systems [17, 18, 20, 24–26].

Until this point, WDFs were limited to a *single one-port* nonlinearity. When modeling *distributed* systems in the closely-related digital waveguide context [2], some may consider this to only be a mild restriction, since in that context multiple nonlinearities may sometimes be considered decoupled by propagation delays [30]. However, for lumped systems, the restriction to a single nonlinearity is a *significant* limitation for WDFs. Attempts to circumvent this limitation have generated a large body of research.

The simplest technique combines multiple nonlinear elements into a single nonlinear one-port. Yeh and Smith derived an implicit diode clipper pair wave domain characteristic via numerical methods [28, 29]. Paiva *et al.* derived a simplified *explicit* diode pair wave-domain description using the Lambert $\mathcal{W}$ function, considering also semi-explicit versions [34, 35]. Werner *et al.* refined aspects of this model and generalized it to diode clippers with an arbitrary number of diodes in each direction, an arrangement common in "modded" and stock guitar distortion pedals [45].

Multiport nonlinear elements can sometimes be simplified to *cross-control* models [21, 22, 31, 33]. Karjalainen and Pakarinen's WDF tube amplifier stage assumed zero grid current. This physically reasonable simplification allows the nonlinearity to be modeled as a plate–cathode *one-port* with the grid voltage as a cross control [21]. They later refined their model by relaxing this assumption [23]. In these circuits and especially those with more than one multiport nonlinear element, the judicious use of ad hoc unit delays aids realizability at the cost of accuracy [21–23, 31–33]. This mirrors how unit delays were used to separate stages [50, 51]

before the development of reflection-free ports [54].

Iterative schemes also hold promise. D'Angelo *et al.* built on [21, 23, 36] by modeling the triode as a single memoryless nonlinear 3-port WDF element—they designed a secant-method-derived solver, customized to one set of triode equations: the Cardarilli *et al.* tube model [37, 38]. Building on their previous work on topology-related delay-free loops [39], Schwerdtfeger and Kummert proposed a "multidimensional" approach to global iterative methods which leverages WDF contractivity properties to guarantee convergence in circuits with multiple nonlinearities [40].

Nonlinear elements in circuits with only small perturbations around an operating point may be approximated by linearized models. De Sanctis and Sarti suggest a hybrid-π bipolar junction transistor (BJT) model for modeling a class-A amplifier [31]. A companion paper to the present work discusses how linear controlled sources may be incorporated into WDFs and should be able to accommodate linearization techniques in general [47]. Bernardini *et al.* expanded *piecewise* linear (PWL) models [7] to cases with multiple one-port nonlinearities, yielding realizable structures from algebraic manipulation of a PWL description [42–44].

Reference circuits with multiple/multiport nonlinearities tend to have complicated topologies which make them intractable with known WDF techniques. In this research thread, the interaction between multiport nonlinearities and topological issues has been understudied, though special cases have been noticed. In 2004, Petrausch and Rabenstein proposed a technique for modeling reference circuits having multiple nonlinearities [19, 24]; however its application is limited to circuits with a vector parallel connection between the nonlinearities and a linear subcircuit. Karjalainen and Pakarinen mention computability issues that could arise from trying to add an external plate–grid Miller capacitance to their model [21,23]. Similar issues in feedback circuits have been noted by D'Angelo *et al.* [37] and De Sanctis and Sarti [31].

Significant insights into topological aspects of *linear* WDFs were provided by Fränken *et al.*, who adopted a graph-theoretic approach using connected graphs and SPQR trees [48, 49]. Their method systematizes decomposition of *linear* circuits containing multiport elements into a WDF adaptor structure. In a companion paper, we showed how to derive scattering matrices for the relatively unknown but very common $\mathcal{R}$-type adaptors that arise in these structures, which may also involve absorbed multiport linear elements [47].

3. DERIVING ADAPTOR STRUCTURE

Deriving WDF adaptor structures by hand can be difficult, especially for reference circuits with complicated topologies and multiple/multiport nonlinearities. In fact, systematic procedures for deriving them seem unknown. Although complicated topological issues may arise even in linear circuits, they are particularly common in nonlinear circuits with multiple/multiport elements. In the WDF literature, multiport nonlinearities seem to always appear in configurations which consider a port between each terminal of the nonlinear device and ground [21–23, 31–33]. In these cases, the number of ports in the nonlinear element is equal to the number of terminals. This perspective greatly limits the class of appropriate reference circuits—it has no accommodations for arbitrary networks with embedded nonlinearities, or specifically circuits with "feedback" between terminals of nonlinear elements.

To expand the range of appropriate WDF reference circuits, we extend the method of Fränken *et al.* [48,49][2] by making two observations. First, their method can be applied to circuits with multiport *nonlinearities*—to avoid splitting them up during the "split components" search, we apply their *replacement graph* technique to multiport nonlinear elements.

Second, we note that designating *all* nonlinear elements (one- and multi-port) in a reference circuit as a collection which must not be separated, replacing it with a single replacement graph, and searching for split components yields an *SPQR tree* with all nonlinear elements embedded in a root element. Typically it is an $\mathcal{R}$-type topology: a topology which cannot be decomposed further into series and parallel topologies. Rarely, it could have series or parallel topology [19]. In those cases, our method remains just as applicable—the root node should just be called $\mathcal{S}$ or $\mathcal{P}$ as appropriate. In all cases, the scattering at this topology can be found using a method derived from Modified Nodal Analysis (MNA) [55, 56], which is described in detail in a companion paper [47].

A WDF adaptor structure follows from the derived SPQR tree. This structure contains all of the nonlinear circuit elements embedded inside of a complicated $\mathcal{R}$-type topology. All of the WDF subtrees that hang below this root collection may be handled with standard techniques. The behavior of the root element itself, however, is not entirely obvious at first. To decompose this problem into pieces that are more approachable, we propose a modification to the SPQR tree of Fränken *et al.*—this modification involves pulling all of the nonlinear circuit elements *out* of the root $\mathcal{R}$-type adaptor and placing them *above* the $\mathcal{R}$-type adaptor.

Hence the vector nonlinearity is separated from the scattering. With both scattering and the nonlinear elements described in the wave domain, this problem framework is summarized by

$$\text{wave nonlinearity} \quad \{\boldsymbol{a}_I = f(\boldsymbol{b}_I) \tag{1}$$

$$\text{scattering} \quad \begin{cases} \boldsymbol{b}_I = \mathbf{S}_{11}\boldsymbol{a}_I + \mathbf{S}_{12}\boldsymbol{a}_E & (2) \\ \boldsymbol{b}_E = \mathbf{S}_{21}\boldsymbol{a}_I + \mathbf{S}_{22}\boldsymbol{a}_E\,, & (3) \end{cases}$$

with external incident and reflected wave vectors $\boldsymbol{a}_E$ and $\boldsymbol{b}_E$, internal incident and reflected wave vectors $\boldsymbol{a}_I$ and $\boldsymbol{b}_I$, scattering matrix $\mathbf{S} = \left[\begin{smallmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{smallmatrix}\right]$, and vector nonlinear function $f(\cdot)$. This is shown as a vector signal flow graph in Fig. 1a.

4. LOOP RESOLUTION

There are two issues with (1)–(3) / Fig. 1a. First, they typically form an implicit and coupled set of transcendental equations—something that will not be convenient to solve. Second, there is a noncomputable delay-free loop through $f(\cdot)$ and $\mathbf{S}_{11}$. This combination of an $\mathcal{R}$-type scattering matrix and a vector of nonlinearities at the root can potentially be handled in various ways. In this paper, we propose a special case of the *K-method*, a technique for resolving implicit loops in nonlinear state space (NLSS) systems which is well-known in physical modeling [57–59], as a solution.

Our main framework separates (1)–(3) / Fig. 1a into three phenomenon: *scattering* ($\mathbf{S}$ and its partitions) between wave variables at external ($\boldsymbol{a}_E$, $\boldsymbol{b}_E$) and internal ($\boldsymbol{a}_I$, $\boldsymbol{b}_I$) ports, *conversion* ($\mathbf{C}$ and its partitions) between internal wave ($\boldsymbol{a}_C$, $\boldsymbol{b}_C$) and Kirchhoff ($\boldsymbol{v}_C$, $\boldsymbol{i}_C$) variables, and a *vector nonlinear* Kirchhoff domain relationship $h(\cdot)$ between $\boldsymbol{v}_C$ and $\boldsymbol{i}_C$. This problem framework is

[2] A review of their method is given in [47].

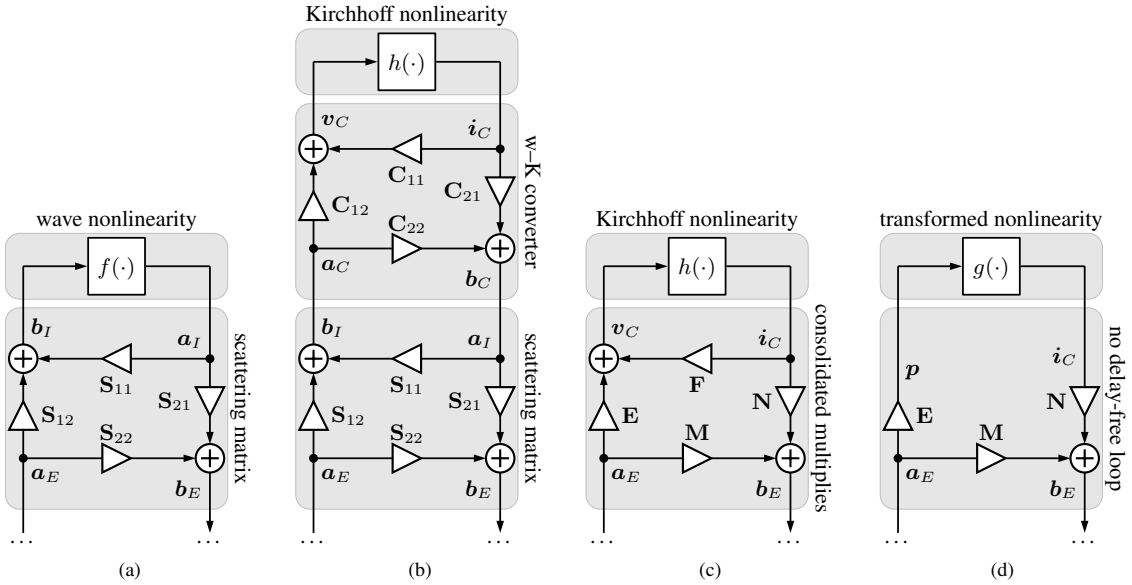


Figure 1: *Framework signal flow graphs: (a) A framework (1)–(3) entirely in the wave domain; (b) A framework with a Kirchhoff-domain vector nonlinearity (4)–(10) involving Kirchhoff nonlinearity $h(\cdot)$, w–K converter $\mathbf{C}$, and scattering matrix $\mathbf{S}$; (c) $\mathbf{S}$ and $\mathbf{C}$ partitions consolidated into NLSS (14) matrices $\mathbf{E}$, $\mathbf{F}$, $\mathbf{M}$, and $\mathbf{N}$; and (d) resolving the delay-free loop by transforming $\boldsymbol{i}_C = h(\boldsymbol{v}_c)$ into $\boldsymbol{i}_C = g(\boldsymbol{p})$ according to (15)–(16). In (b)–(d), $\boldsymbol{a}_E$ is supplied by and $\boldsymbol{b}_E$ delivered to classical WDF subtrees below.*

summarized by

$$\text{Kirchhoff nonlinearity} \quad \{\boldsymbol{i}_C = h(\boldsymbol{v_C}) \tag{4}$$

$$w\text{–}K \text{ converter} \quad \begin{cases} \boldsymbol{v}_C = \mathbf{C}_{11}\boldsymbol{i}_C + \mathbf{C}_{12}\boldsymbol{a}_C & (5) \\ \boldsymbol{b}_C = \mathbf{C}_{21}\boldsymbol{i}_C + \mathbf{C}_{22}\boldsymbol{a}_C & (6) \end{cases}$$

$$\text{compatibility} \quad \begin{cases} \boldsymbol{a}_C = \boldsymbol{b}_I & (7) \\ \boldsymbol{a}_I = \boldsymbol{b}_C & (8) \end{cases}$$

$$\text{scattering} \quad \begin{cases} \boldsymbol{b}_I = \mathbf{S}_{11}\boldsymbol{a}_I + \mathbf{S}_{12}\boldsymbol{a}_E & (9) \\ \boldsymbol{b}_E = \mathbf{S}_{21}\boldsymbol{a}_I + \mathbf{S}_{22}\boldsymbol{a}_E & (10) \end{cases}$$

and shown as a vector signal flow graph in Fig. 1b.

In this form, the structure is still *noncomputable* due to the presence of multiple delay-free loops. To ameliorate this, we consolidate the eight $\mathbf{S}$ and $\mathbf{C}$ partitions into four matrices $\mathbf{E}$, $\mathbf{F}$, $\mathbf{M}$, and $\mathbf{N}$. First consider (5) and (10) in matrix form,

$$\begin{bmatrix} \boldsymbol{v}_C \\ \boldsymbol{b}_E \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{12} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{21} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_C \\ \boldsymbol{a}_I \end{bmatrix} + \begin{bmatrix} \mathbf{C}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{i}_C \\ \boldsymbol{a}_E \end{bmatrix}, \tag{11}$$

and (6)–(9) in matrix form (eliminating $\boldsymbol{b}_I$ and $\boldsymbol{b}_C$),

$$\begin{bmatrix} \mathbf{I} & -\mathbf{S}_{11} \\ -\mathbf{C}_{22} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_C \\ \boldsymbol{a}_I \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{S}_{12} \\ \mathbf{C}_{21} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{i}_C \\ \boldsymbol{a}_E \end{bmatrix}. \tag{12}$$

Solving (12) for internal port wave vectors $[\boldsymbol{a}_C^\mathsf{T}, \boldsymbol{a}_I^\mathsf{T}]^\mathsf{T}$ yields

$$\begin{bmatrix} \boldsymbol{a}_C \\ \boldsymbol{a}_I \end{bmatrix} = \begin{bmatrix} \mathbf{S}_{11}\mathbf{H}\mathbf{C}_{21} & \mathbf{S}_{12} + \mathbf{S}_{11}\mathbf{H}\mathbf{C}_{22}\mathbf{S}_{12} \\ \mathbf{H}\mathbf{C}_{21} & \mathbf{H}\mathbf{C}_{22}\mathbf{S}_{12} \end{bmatrix} \begin{bmatrix} \boldsymbol{i}_C \\ \boldsymbol{a}_E \end{bmatrix}, \tag{13}$$

where $\mathbf{H} = (\mathbf{I} - \mathbf{C}_{22}\mathbf{S}_{11})^{-1}$. Plugging (13) into (11) and recalling (4) yields a consolidated version of (4)–(10),

$$\begin{cases} \boldsymbol{i}_C = h(\boldsymbol{v}_C) \\ \boldsymbol{v}_C = \mathbf{E}\boldsymbol{a}_E + \mathbf{F}\boldsymbol{i}_C \\ \boldsymbol{b}_E = \mathbf{M}\boldsymbol{a}_E + \mathbf{N}\boldsymbol{i}_C \end{cases} \text{with} \quad \begin{aligned} \mathbf{E} &= \mathbf{C}_{12}(\mathbf{I} + \mathbf{S}_{11}\mathbf{H}\mathbf{C}_{22})\mathbf{S}_{12} \\ \mathbf{F} &= \mathbf{C}_{12}\mathbf{S}_{11}\mathbf{H}\mathbf{C}_{21} + \mathbf{C}_{11} \\ \mathbf{M} &= \mathbf{S}_{21}\mathbf{H}\mathbf{C}_{22}\mathbf{S}_{12} + \mathbf{S}_{22} \\ \mathbf{N} &= \mathbf{S}_{21}\mathbf{H}\mathbf{C}_{21}, \end{aligned} \tag{14}$$

which is shown as a vector signal flow graph in Fig. 1c.

In this form, the structure is *still* noncomputable due to one remaining delay-free loop. But, since it is now in a standard NLSS form (albeit one without states), the K-method can be used to render it computable. This yields

$$\begin{cases} \boldsymbol{i}_C = g(\boldsymbol{p}) \\ \boldsymbol{p} = \mathbf{E}\boldsymbol{a}_E \\ \boldsymbol{b}_E = \mathbf{M}\boldsymbol{a}_E + \mathbf{N}\boldsymbol{i}_C \end{cases} \text{with} \quad \begin{aligned} \mathbf{E} &= \mathbf{C}_{12}(\mathbf{I} + \mathbf{S}_{11}\mathbf{H}\mathbf{C}_{22})\mathbf{S}_{12} \\ \mathbf{M} &= \mathbf{S}_{21}\mathbf{H}\mathbf{C}_{22}\mathbf{S}_{12} + \mathbf{S}_{22} \\ \mathbf{N} &= \mathbf{S}_{21}\mathbf{H}\mathbf{C}_{21}, \end{aligned} \tag{15}$$

where $h(\cdot) : \boldsymbol{v}_C \to \boldsymbol{i}_C$ and $g(\cdot) : \boldsymbol{p} \to \boldsymbol{i}_C$ are related by

$$\begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{i}_C \end{bmatrix} = \begin{bmatrix} \mathbf{I} & -\mathbf{K} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_C \\ \boldsymbol{i}_C \end{bmatrix}. \tag{16}$$

In general the loop resolution matrix $\mathbf{K}$ (the namesake of the K-method) depends on the chosen discretization method [57]. With no dynamics, we simply have $\mathbf{K} = \mathbf{F}$ from (14). The explicit framework (15) is shown as a vector signal flow graph in Fig. 1d. Since each entry in $\boldsymbol{p}$ is formed by different linear combination of voltages and currents, we can consider $\boldsymbol{p}$ to be a pseudo-wave variable, albeit one without an immediate physical interpretation [60].

The transformation (16) allows us to tabulate solutions to $h(\cdot)$, which is explicit and easy to tabulate, and transform it to $g(\cdot)$, a

domain where it would be difficult to tabulate directly. The general framework for a reference circuit with multiple/multiport nonlinearities is hence rendered tractable.

5. DERIVING SYSTEM MATRICES

(15)–(16) / Fig. 1d requires $\mathbf{C}$ and $\mathbf{S}$ matrices that characterize the system; we briefly review their derivations.

$\mathbf{C}$ follows directly from the standard WDF voltage wave definition. Recalling (5)–(6) / Fig. 1b, $\mathbf{C}$ defines the relationship among the voltages $\boldsymbol{v}_C$, currents $\boldsymbol{i}_C$, incident waves $\boldsymbol{a}_C$, and reflected waves $\boldsymbol{b}_C$ at the internal ports. Accordingly, the values of these matrices come from wave variable definitions. Rearranging the standard voltage wave definition [1, 61] yields

$$\underbrace{\begin{aligned}\boldsymbol{a}_C &= \boldsymbol{v}_C + \mathbf{R}_I \boldsymbol{i}_C \\ \boldsymbol{b}_C &= \boldsymbol{v}_C - \mathbf{R}_I \boldsymbol{i}_C\end{aligned}}_{\text{voltage wave definition}} \quad \rightarrow \quad \underbrace{\begin{aligned}\boldsymbol{v}_C &= \boldsymbol{a}_C - \mathbf{R}_I \boldsymbol{i}_C \\ \boldsymbol{b}_C &= \boldsymbol{a}_C - 2\mathbf{R}_I \boldsymbol{i}_C\end{aligned}}_{\text{rearranged}} \tag{17}$$

where $\mathbf{R}_I$ is a diagonal matrix of internal port resistances. Hence:

$$\mathbf{C} = \begin{bmatrix}\mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22}\end{bmatrix} = \begin{bmatrix}-\mathbf{R}_I & \mathbf{I} \\ -2\mathbf{R}_I & \mathbf{I}\end{bmatrix}. \tag{18}$$

$\mathbf{S}$ matrices are found using a method which leverages Modified Nodal Analysis on an equivalent circuit.

Recalling (9)–(10) / Fig. 1b, $\mathbf{S}$ defines how incident waves $\boldsymbol{a}_E$ and $\boldsymbol{a}_I$ scatter to yield reflected waves $\boldsymbol{b}_E$ and $\boldsymbol{b}_I$; hence $\mathbf{S}$ is derived from the topology around the root. A derivation for $\mathbf{S}$ was first presented in [46]. We briefly review the derivation here—it is discussed in detail in a companion paper [47].

First, an *instantaneous Thévenin port equivalent* to the $\mathcal{R}$-type topology (which may include multiport linear elements) is formed; each port n is replaced by a resistive voltage source with value a_n and resistance R_n (equal to the port resistance). This circuit is described by Modified Nodal Analysis as

$$\underbrace{\begin{bmatrix}\mathbf{Y} & \mathbf{A} \\ \mathbf{B} & \mathbf{D}\end{bmatrix}}_{\text{MNA matrix } \mathbf{X}} \begin{bmatrix}\boldsymbol{v}_n \\ \boldsymbol{j}\end{bmatrix} = \begin{bmatrix}\boldsymbol{i}_s \\ \boldsymbol{e}\end{bmatrix}, \tag{19}$$

where $\mathbf{X}$ partitions $\mathbf{Y}$, $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{D}$ are found by well-known element stamp procedures [55, 56, 58, 59]. Confronting (19) with wave variable definitions and compatibility requirements yields

$$\mathbf{S} = \mathbf{I} + 2\begin{bmatrix}\mathbf{0} & \mathbf{R}\end{bmatrix}\mathbf{X}^{-1}\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}^{\mathsf{T}}, \tag{20}$$

where $\mathbf{R} = \text{diag}(\mathbf{R}_I, \mathbf{R}_E)$ is a diagonal matrix of port resistances. $\mathbf{S}_{11}$, $\mathbf{S}_{12}$, $\mathbf{S}_{21}$, and $\mathbf{S}_{22}$ are found simply by partitioning $\mathbf{S}$.

6. CASE STUDY

As a case study on the methods presented in this paper, we model the first clipping stage from the Big Muff Pi distortion pedal [62]. This circuit has *three* nonlinear elements: two 1N914 diodes (D_3 and D_4) and a 2N5089 BJT. We combine the two diodes into a single one-port nonlinearity as in [28, 29, 34, 35, 45]. Hence the WDF we derive has *three* nonlinear ports: the diode pair, the BJT's V_{BE} junction, and the BJT's V_{CE} junction.

The names, values, and graph edges of each circuit element are shown in Table 1; coefficients pertaining to the nonlinear behavior of the diodes and transistor are shown in Table 2.

Table 1: *Big Muff Pi Circuit Elements.*

element	value	edge
V_{in}	(input) V	1
R_{17}	470 kΩ	5
R_{18}, V_{CC}	10 kΩ, 9 V	9
R_{19}	10 kΩ	3
R_{20}	100 kΩ	4
R_{21}	150 Ω	8
C_5	100 nF	2
C_6	470 pF	7
C_{12}	1 μF	6

Table 2: *Big Muff Pi Circuit Elements.*

element	value	description
$I_{s(d)}$	2.52 nA	diode reverse saturation current
V_T	25.85 mV	thermal voltage
I_s	5.911 fA	BJT reverse saturation current
α_F	0.9993	BJT forward current gain
α_R	0.5579	BJT reverse current gain

The derivation of a WDF from the Big Muff Pi clipping stage is shown in Fig. 2; details follow.

The Big Muff Pi first clipping stage circuit is shown in Fig. 2a. Using the method of Fränken *et al.* [49], a connected graph structure is formed. In this graph (Fig. 2b), each circuit port corresponds to a graph edge, which is assigned an integer index. As described in §3, we extend the replacement graph approach of Fränken *et al.* to the nonlinear case—all nodes that are connected to a nonlinear element (nodes d, e, f, and g) are bundled together using a replacement graph, shown in gray in Fig. 2b.

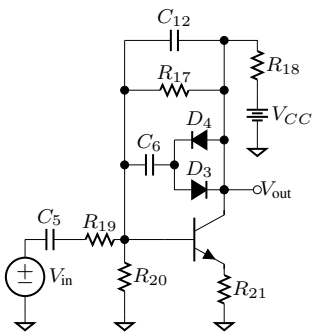
A standard search for split components yields the graph shown in Fig. 2e. As expected, the replacement graph has no splits and has ended up embedded in an $\mathcal{R}$-type topology which cannot be decomposed further into series and parallel connections.

We form a modified SPQR tree by designating the $\mathcal{R}$-type topology as the root of a tree and letting the remaining one-ports and adaptors dangle below it (Fig. 2d). Here we break from standard technique, by extracting the three nonlinear ports (the diode pair $D_{3,4}$, V_{BC}, and V_{BE}, designated with a dark gray background) from the $\mathcal{R}$-type adaptor and placing them *above* the root element (designated by a light gray background).

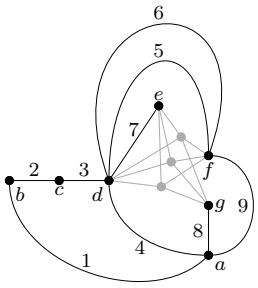
A WDF tree (Fig. 2e) follows from the modified SPQR tree of Fig. 2d; a rearranged version of Fig. 2a which highlights the derived adaptor structure is shown in Fig. 2f. Note that V_{in} is given a fictitious 1 Ω source resistance to avoid having a non-root ideal voltage source, which would cause computability issues.

Recall that a WDF adaptor can have only a single "reflection-free port." Since the $\mathcal{R}$-type adaptor would need *three* reflection-free ports to guaranteed realizability in the standard WDF framework, we make recourse to the techniques presented in §4 to handle the collection of the $\mathcal{R}$-type adaptor and the nonlinearities. The rest of the adaptors and one-ports form WDF subtrees which are already computable with traditional techniques.

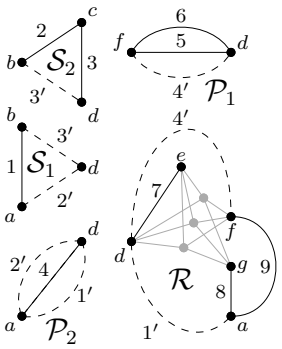
The behavior of $\mathcal{R}$ and the nonlinearities is described by a system in the form of Fig. 1b. Here, w–K converter matrix $\mathbf{C}$ partitions are given by the vector form of standard voltage wave definitions (18). Scattering matrix $\mathbf{S}$ partitions describe the scattering behavior of the root $\mathcal{R}$-type adaptor (Fig. 2g) and can be found using our MNA-derived method (20) on its instantaneous Thévenin port equivalent (Fig. 2h). Our matrix of port resistances



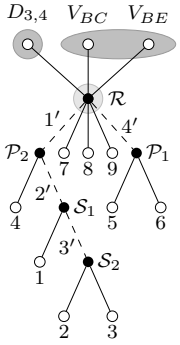
(a) *Big Muff Pi clipping stage.*



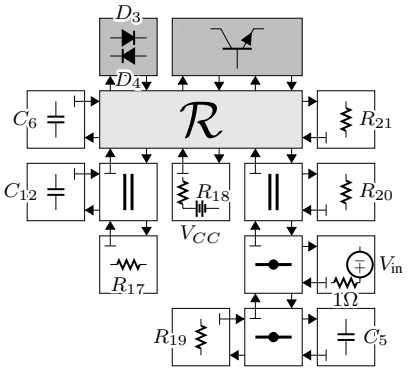
(b) *Graph structure.*



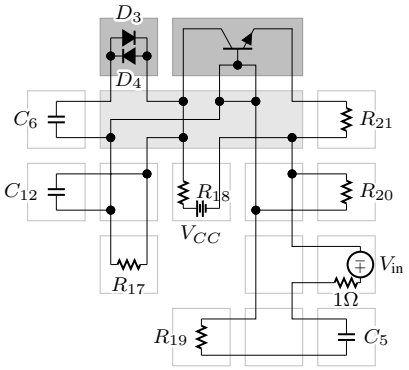
(c) *Finding split components.*



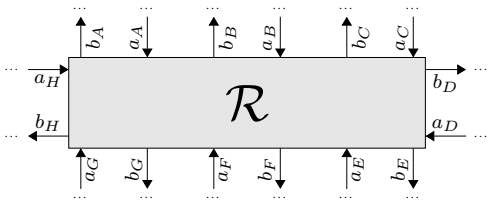
(d) *Modified SPQR.*



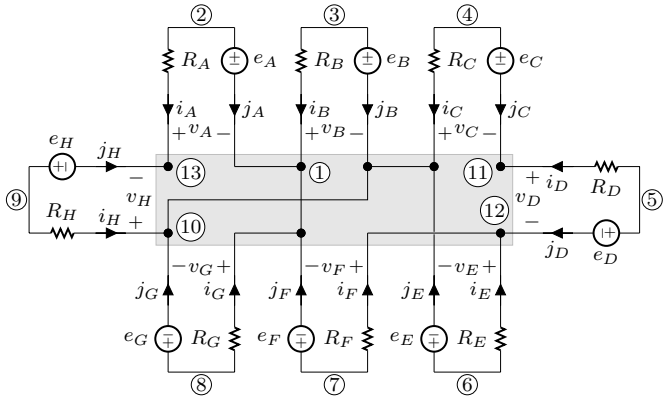
(e) *Corresponding WDF adaptor structure.*



(f) *Circuit arranged to highlight WDF structure.*



(g) *$\mathcal{R}$-type adaptor detail.*



(h) *Instantaneous Thévenin port equivalent.*

Figure 2: *Deriving a WDF adaptor structure for the Big Muff Pi clipping stage.*

$$\mathbf{X} = \left[\begin{array}{c|c} \mathbf{Y} & \mathbf{A} \\ \hline \mathbf{B} & \mathbf{D} \end{array}\right] =$$

	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪	⑫	⑬	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
②	G_A	0	0	0	0	0	0	0	0	0	0	$-G_A$	1	0	0	0	0	0	0	0
③	0	G_B	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
④	0	0	G_C	0	0	0	0	0	$-G_C$	0	0	0	0	0	1	0	0	0	0	0
⑤	0	0	0	G_D	0	0	0	0	0	$-G_D$	0	0	0	0	0	1	0	0	0	0
⑥	0	0	0	0	G_E	0	0	0	0	0	$-G_E$	0	0	0	0	0	1	0	0	0
⑦	0	0	0	0	0	G_F	0	0	0	0	$-G_F$	0	0	0	0	0	0	1	0	0
⑧	0	0	0	0	0	0	G_G	0	0	0	0	0	0	0	0	0	0	0	1	0
⑨	0	0	0	0	0	0	0	G_H	$-G_H$	0	0	0	0	0	0	0	0	0	0	1
⑩	0	0	$-G_C$	0	0	0	0	$-G_H$	G_C+G_H	0	0	0	0	-1	0	0	-1	0	-1	0
⑪	0	0	0	$-G_D$	0	0	0	0	0	G_D	0	0	0	0	-1	0	0	0	0	0
⑫	0	0	0	0	$-G_E$	$-G_F$	0	0	0	0	G_E+G_F	0	0	0	0	-1	0	0	0	0
⑬	$-G_A$	0	0	0	0	0	0	0	0	0	0	G_A	0	0	0	0	0	0	0	-1
A	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	1	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0
C	0	0	1	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0
D	0	0	0	1	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0
E	0	0	0	0	1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	1	0	0	0	-1	0	0	0	0	0	0	0	0

Figure 3: *Big Muff $\mathcal{R}$-type adaptor MNA matrix—example resistor stamp in light shading, example voltage source stamp in dark shading.*

is $\mathbf{R} = \text{diag}(\mathbf{R}_I, \mathbf{R}_E)$, where $\mathbf{R}_I = \text{diag}(R_A, R_B, R_C)$ and $\mathbf{R}_E = \text{diag}(R_D, R_E, R_F, R_G, R_H)$ are matrices of internal and external port resistances. $\mathbf{R}_E$ is prescribed by the port resistances of the WDF subtrees, and the port resistances in $\mathbf{R}_I$ are assigned arbitrarily as $R_A = R_B = R_C = 1$ kΩ; we cannot make all three internal ports reflection-free, and they will get resolved by the K-method anyways, so we don't even try to adapt any of them.

The MNA system matrix $\mathbf{X}$ that arises from the stamp procedure, with node ① as the datum node, is shown in Fig. 3.

The behavior of our vector of nonlinearities is described in the Kirchhoff domain by $h(\cdot)$, which maps a vector of voltages $\boldsymbol{v}_C = [v_{\text{diodes}}, v_{\text{BC}}, v_{\text{BE}}]^\mathsf{T}$ to a vector of currents $\boldsymbol{i}_C = [i_{\text{diodes}}, -i_C, i_E]^\mathsf{T}$ according to the Shockley ideal diode law,

$$i_{\text{diodes}} = I_{s(d)}(e^{v_{\text{diodes}}/V_T} - 1) - I_{s(d)}(e^{-v_{\text{diodes}}/V_T} - 1)\,, \quad (21)$$

and the Ebers–Moll BJT model,

$$i_C = I_s(e^{V_{\text{BE}}/V_T} - 1) - I_s/\alpha_R(e^{V_{\text{BC}}/V_T} - 1) \quad (22)$$

$$i_E = I_s/\alpha_F(e^{V_{\text{BE}}/V_T} - 1) - I_s(e^{V_{\text{BC}}/V_T} - 1)\,. \quad (23)$$

$h(\cdot)$ is tabulated with a $201\times201\times201$ grid across the circuit's operating range of voltages (determined experimentally with SPICE).

Since this structure contains multiple noncomputable loops, it is collapsed down into a NLSS form (Fig. 1c), where the values of $\mathbf{E}$, $\mathbf{F}$, $\mathbf{M}$, and $\mathbf{N}$ are given as functions of $\mathbf{C}_{11}$, $\mathbf{C}_{12}$, $\mathbf{C}_{21}$, $\mathbf{C}_{22}$, $\mathbf{S}_{11}$, $\mathbf{S}_{12}$, $\mathbf{S}_{21}$, and $\mathbf{S}_{22}$ by (14). This structure *still* contains a noncomputable loop through $h(\cdot)$ and $\mathbf{F}$, but since it is a standard NLSS system (albeit one without states), we can render it computable using the K-method [57–59], yielding a system in the form of Fig. 1d / (15). Matrices $\mathbf{E}$, $\mathbf{M}$, and $\mathbf{N}$ are already known, and a scattered tabulation $g(\cdot)$ is found from the gridded tabulation $h(\cdot)$, using $\mathbf{K} = \mathbf{F}$ according to (16).

To explain how the system is computable, let us walk through one time step of the algorithm. All of the WDF subtrees which dangle from $\mathcal{R}$ are handled in the normal way, and the collection of $\mathcal{R}$ and the nonlinearities are handled according to (15). At each time step during runtime, these subtrees deliver a length-5 vector of incident waves $\boldsymbol{a}_E = [a_D, a_E, a_F, a_G, a_H]^\mathsf{T}$ to $\mathcal{R}$. We find length-3 vector $\boldsymbol{p}$ from $\boldsymbol{a}_E$ by $\boldsymbol{p} = \mathbf{E}\boldsymbol{a}_E$. Using scattered interpolant methods[3], we find length-3 vector $\boldsymbol{i}_C$ from $\boldsymbol{p}$ by $\boldsymbol{i}_C = g(\boldsymbol{p})$. Finally the length-5 output vector $\boldsymbol{b}_E$ of the root $\mathcal{R}$–nonlinearity collection is found by combining contributions from $\boldsymbol{a}_E$ and $\boldsymbol{i}_C$ by $\boldsymbol{b}_E = \mathbf{M}\boldsymbol{a}_E + \mathbf{N}\boldsymbol{i}$. $\boldsymbol{b}_E$ is propagated down into the standard WDF subtrees and we can advance to the next time step.

Simulation results are shown in Fig. 4. On top is a 15-ms-long guitar input signal V_{in}. The middle panel compares a "ground-truth" SPICE simulation and the results of our WDF algorithm. They are nearly a match, confirming the validity of our novel WDF formulation. The error signal between the two is shown in the bottom panel; small discrepancies can be ascribed to limitations of the $201\times201\times201$ lookup table, (expected) aliasing in the WDF, linear resampling of the SPICE output to WDF simulation's constant time grid for comparison, and general numerical concerns.

7. CONCLUSION AND FUTURE WORK

In this paper, we've presented a framework for modeling lumped systems (in particular, electronic circuits) with multiple and multiport nonlinear elements and arbitrary topologies as Wave Digital Filters. We expand on the graph-theoretic approach of Fränken *et al.* [48, 49], applying it rather to yield an adaptor structure for circuits with multiple/multiport nonlinearities. Replacing the collection of all nonlinear elements with a single replacement graph during the separation-pair-finding process yields a modified SPQR tree, where the root node tends to be an $\mathcal{R}$-type adaptor, with standard WDF subtrees hanging below and a vector of nonlinearities hanging above. This confines issues of computability to a *minimal subsystem* comprised of the $\mathcal{R}$-type adaptor and vector of nonlinearities at the root of a WDF tree.

[3] `http://www.mathworks.com/help/matlab/ref/scatteredinterpolant-class.html`

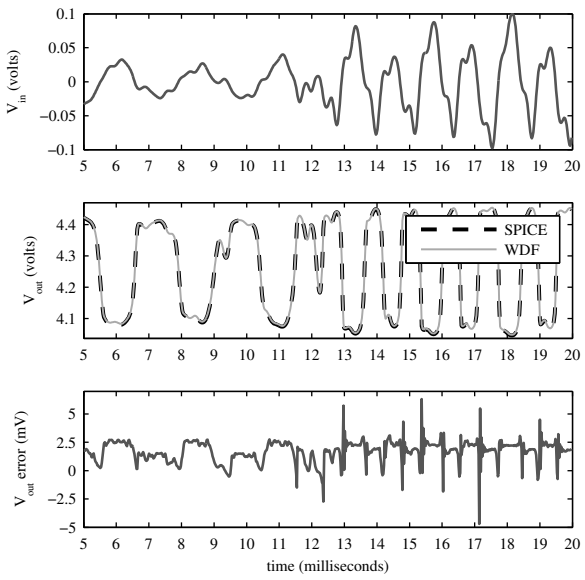


Figure 4: *Big Muff Pi clipping stage simulation. V_{in} (top), V_{out} from SPICE and WDF (middle), and WDF V_{out} error (bottom). SPICE used a maximum time step of 10^{-5} seconds and WDF simulation used a sampling rate of 44.1 kHz.*

This combination of this $\mathcal{R}$-type adaptor and the *vector* of nonlinearities is noncomputable. To render it computable, we rearrange it so our system of "consolidated multiplies" $\mathbf{E}$, $\mathbf{F}$, $\mathbf{M}$, and $\mathbf{N}$ is solvable with the K-method [57–59]. The advantage of solving the root collection with the K-method is that tabulation can be done simply in the Kirchhoff domain, where $\boldsymbol{i}_C$–$\boldsymbol{v}_C$ relationships are usually explicit and memoryless.[4] A disadvantage is that when gridded tabulation is transformed to the $\boldsymbol{p}$–$\boldsymbol{i}_C$ domain, it will no longer be gridded, limiting interpolation to scattered methods.

We've focused on creating a general problem framework with the potential for solution through diverse methods, not only the K-method-derived approach we've proposed. Future work should investigate computational efficiency and other methods for solving the localized minimal root subsystem.

The internal *structure* of $\mathbf{S}$, $\mathbf{C}$, $\mathbf{E}$, $\mathbf{F}$, $\mathbf{M}$, and $\mathbf{N}$ could potentially be exploited for computational saving in future work.

Using iterative techniques to tabulate the $\boldsymbol{p}$–$\boldsymbol{i}_C$ domain directly, rather than transforming an $\boldsymbol{i}_C$–$\boldsymbol{v}_C$ tabulation, it should be possible to create a gridded tabulation, opening up the possibility of using gridded interpolant methods. This would require good initial guesses for the iterative solver or the use of, e.g., Newton Homotopy [58, 59]. Potentially, a transformed solution could be used as initial guesses for forming a gridded interpolation [34].

We could consider keeping our root topology $\mathcal{R}$ and vector of nonlinearities entirely in the wave domain, as in Fig. 1a. In this case, a nonlinear wave-domain function $f(\cdot)$ would need to solve $\boldsymbol{a}_I = f(\boldsymbol{b}_I)$. This system could *also* be resolved with the K-method. However, it is unlikely that a closed-form solution to $f(\cdot)$ will be available, so an explicit solution can't be tabulated for $f(\cdot)$ and then transformed. With iterative techniques, tabulation in the transformed $f(\cdot)$ domain should be possible.

As another option, the instantaneous Thévenin port equivalent concept could be applied to all external ports of the nonlinear root node topology and vector nonlinearity, creating a Kirchhoff-domain subsystem grafted onto standard WDF subtrees below. As in the scattering matrix derivation [47], each external port n in the root scattering matrix would be replaced by its instantaneous Thévenin equivalent—a series combination of a voltage source with a value equal to the incident wave a_n and a resistor equal to the port resistance R_n—and then the system could be solved with the K-method or any other Kirchhoff-domain method.

In our main formulation or the others proposed above, one could develop alternate derivations that exchange the roles of currents and voltages, using e.g., $\boldsymbol{v}_C = g^{-1}(\boldsymbol{i}_C)$. This may be particularly applicable to using iterative methods to tabulate the nonlinearities; Yeh *et al.* report that iteration on terminal voltages converges faster than iteration on device currents [58].

Future work should explore these alternate formulations and practical aspects of N-dimensional scattered interpolation. Our exposition uses standard voltage wave variables for simplicity of presentation, but for time-varying structures, power waves provide theoretical energetic advantages [15, 61, 63]. For alternative wave variable definitions, (17)–(18) will be modified as appropriate.

Energetic concerns in the root subsystem remain unexplored and potentially problematic, e.g., for highly resonant circuits. Previous work investigates energetic aspects of nonlinear table interpolation techniques in WDFs [41]—although it is limited to the one-port case, it could serve as a prototype for future work.

[4] Nonlinearities "with memory," e.g., NL inductors/capacitors, can be expressed as instantaneous aspects combined with WDF "mutators" [11].

8. REFERENCES

[1] A. Fettweis, "Wave digital filters: Theory and practice," *Proc. IEEE*, vol. 74, no. 2, pp. 270–327, 1986.

[2] J. O. Smith III, *Physical Audio Signal Processing for Virtual Musical Instruments and Audio Effects*, online book, 2010 edition.

[3] V. Välimäki et al., "Introduction to the special issue on virtual analog audio effects and musical instruments," *IEEE Trans Audio, Speech, Language Process.*, vol. 18, no. 4, pp. 713–714, 2010.

[4] V. Välimäki et al., *Virtual Analog Effects*, chapter 12, pp. 473–522, John Wiley & Sons, second edition, 2011, Appears in [64].

[5] V. Välimäki, "Virtual analog modeling," in *Int. Conf. Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, Sept. 5 2013, keynote.

[6] E. Barbour, "The cool sound of tubes," *IEEE Spectr.*, vol. 35, no. 8, pp. 24–35, Aug. 1998.

[7] K. Meerkötter and R. Scholz, "Digital simulation of nonlinear circuits by wave digital filter principles," in *Proc. IEEE Int. Symp. Circuits Syst.*, June 1989, vol. 1, pp. 720–723.

[8] T. Felderhoff, "Simulation of nonlinear circuits with period doubling and chaotic behavior by wave digital filter principles," *IEEE Trans. Circuits Syst. I: Fundam. Theory Appl.*, vol. 41, no. 7, 1994.

[9] T. Felderhoff, "A new wave description for nonlinear elements," in *Proc. IEEE Int. Symp. Circuits Syst.*, Sept. 1996, vol. 3, pp. 221–224.

[10] A. Sarti and G. De Poli, "Generalized adaptors with memory for nonlinear wave digital structures," in *Proc. European Signal Process. Conf. (EUSIPCO)*, Sept. 1996, vol. 3, pp. 1941–1944.

[11] A. Sarti and G. De Poli, "Toward nonlinear wave digital filters," *IEEE Trans. Signal Process.*, vol. 47, no. 6, pp. 1654–1668, 1999.

[12] F. Pedersini et al., "Block-wise physical model synthesis for musical acoustics," *Electron. Lett.*, vol. 35, no. 17, pp. 1418–1419, 1999.

[13] F. Pedersini et al., "Object-based sound synthesis for virtual environments using musical acoustics," *IEEE Signal Process. Mag.*, vol. 17, no. 6, pp. 37–51, Nov. 2000.

[14] G. De Sanctis, "Una nuova metodologia per l'implementazione automatica di strutture ad onda numerica orientata alla modellazione ad oggetti di interazioni acustiche," M.S. thesis, Politecnico di Milano, Italy, 2002.

[15] S. Bilbao et al., "A power-normalized wave digital piano hammer," in *Proc. Meeting Acoust. Soc. Amer.*, Cancun, Mexico, 2002, vol. 114.

[16] S. Bilbao et al., "The wave digital reed: A passive formulation," in *Proc. Int. Conf. Digital Audio Effects (DAFx-03)*, London, UK, Sept. 8–11 2003, vol. 6, pp. 225–230.

[17] M. Karjalainen, "BlockCompiler: Efficient simulation of acoustic and audio systems," in *Proc. Audio Eng. Soc. (AES) Conv.*, Amsterdam, The Netherlands, Mar. 22–25 2003, vol. 114.

[18] G. De Sanctis et al., "Automatic synthesis strategies for object-based dynamical physical models in musical acoustics," in *Proc. Int. Conf. Digital Audio Effects*, London, UK, Sept. 8–11 2003, vol. 6.

[19] S. Petrausch and R. Rabenstein, "Wave digital filters with multiple nonlinearities," in *Proc. European Signal Process. Conf. (EUSIPCO)*, Vienna, Austria, Sept. 2004, vol. 12.

[20] M. Karjalainen, "Efficient realization of wave digital components for physical modeling and sound synthesis," *IEEE Trans. Audio, Speech, Language Process.*, vol. 16, no. 5, pp. 947–956, 2008.

[21] M. Karjalainen and J. Pakarinen, "Wave digital simulation of a vacuum-tube amplifier," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Toulouse, France, May 14–19 2006, pp. 153–156.

[22] J. Pakarinen et al., "Wave digital modeling of the output chain of a vacuum-tube amplifier," in *Proc. Int Conf. Digital Audio Effects (DAFx-09)*, Como, Italy, Sept. 1–4 2009, pp. 153–156.

[23] J. Pakarinen and M. Karjalainen, "Enhanced wave digital triode model for real-time tube amplifier emulation," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 4, pp. 738–746, 2010.

[24] S. Petrausch, *Block based physical modeling*, Ph.D. diss., Friedrich-Alexander-Universität Erlangen–Nürnberg, Germany, 2006.

[25] R. Rabenstein et al., "Block-based physical modeling for digital sound synthesis," *IEEE Signal Process. Mag.*, pp. 42–54, Mar. 2007.

[26] R. Rabenstein and S. Petrausch, "Block-based physical modeling with applications in musical acoustics," *Int. J. Appl. Math. Comput. Sci.*, vol. 18, no. 3, pp. 295–305, 2008.

[27] D. T.-M. Yeh, "Tutorial on wave digital filters," `https://ccrma.stanford.edu/~dtyeh/papers/wdftutorial.pdf`, Jan. 25 2008.

[28] D. T.-M. Yeh and J. O. Smith III, "Simulating guitar distortion circuits using wave digital and nonlinear state-space formulations," in *Proc. Int. Conf. Digital Audio Effects (DAFx-08)*, Espoo, Finland, Sept. 1–4 2008, vol. 11, pp. 19–26.

[29] D. T.-M. Yeh, *Digital implementation of musical distortion circuits by analysis and simulation*, Ph.D. diss., Stanford Univ., CA, 2009.

[30] A. Sarti and G. De Sanctis, "Systematic methods for the implementation of nonlinear wave-digital structures," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 56, no. 2, pp. 460–472, 2009.

[31] G. De Sanctis and A. Sarti, "Virtual analog modeling in the wave-digital domain," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 4, pp. 715–727, 2010.

[32] R. C. D. de Paiva et al., "Real-time audio transformer emulation for virtual tube amplifiers," *EURASIP J. Adv. Signal Process.*, 2011.

[33] P. Raffensperger, "Toward a wave digital filter model of the Fairchild 670 limiter," in *Proc. Int. Conf. Digital Audio Effects (DAFx-12)*, York, UK, Sept. 17–21 2012, vol. 15.

[34] R. C. D. Paiva et al., "Emulation of operational amplifiers and diodes in audio distortion circuits," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 59, no. 10, pp. 688–692, 2012.

[35] R. C. D. de Paiva, *Circuit modeling studies related to guitars and audio processing*, Ph.D. diss., Aalto Univ., Espoo, Finland, 2013.

[36] S. D'Angelo and V. Välimäki, "Wave-digital polarity and current inverters and their application to virtual analog audio processing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Kyoto, Japan, Mar. 2012, pp. 469–472.

[37] S. D'Angelo et al., "New family of wave-digital triode models," *IEEE Trans. Audio, Speech, Language Process.*, vol. 21, no. 2, pp. 313–321, 2013.

[38] S. D'Angelo, *Virtual Analog Modeling of Nonlinear Musical Circuits*, Ph.D. diss., Aalto University, Espoo, Finland, Sept. 2014.

[39] T. Schwerdtfeger and A. Kummert, "A multidimensional signal processing approach to wave digital filters with topology-related delay-free loops," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Florence, Italy, May 4–9 2014, pp. 389–393.

[40] T. Schwerdtfeger and A. Kummert, "A multidimensional approach to wave digital filters with multiple nonlinearities," in *Proc. European Signal Process. Conf. (EUSIPCO)*, Lisbon, Portugal, Sept. 1–5 2014.

[41] K. J. Werner and J. O. Smith III, "An energetic interpretation of nonlinear wave digital filter lookup table error," in *Proc. IEEE Int. Symp. Signals, Circuits, Syst. (ISSCS)*, Iaşi, Romania, July 9–10 2015.

[42] A. Bernardini et al., "Multi-port nonlinearities in wave digital structures," in *Proc. IEEE Int. Symp. Signals, Circuits, Syst. (ISSCS)*, Iaşi, Romania, July 9–10 2015.

[43] A. Bernardini et al., "Modeling a class of multi-port nonlinearities in wave digital structures," in *Proc. European Signal Process. Conf. (EUSIPCO)*, Nice, France, Aug. 31 – Sept. 4 2015, vol. 23.

[44] A. Bernardini, "Modeling nonlinear circuits with multiport elements in the wave digital domain," M.S. thesis, Politecnico di Milano, Italy, Apr. 2015.

[45] K. J. Werner et al., "An improved and general diode clipper model for wave digital filters," in *Proc. Audio Eng. Soc. (AES) Conv.*, New York, NY, Oct. 29 – Nov. 1 2015, vol. 139.

[46] K. J. Werner et al., "A general and explicit formulation for wave digital filters with multiple/multiport nonlinearities and complicated topologies," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust. (WASPAA)*, New Paltz, NY, Oct. 18–21 2015.

[47] K. J. Werner et al., "Wave digital filter adaptors for arbitrary topologies and multiport linear elements," in *Proc. Int. Conf. Digital Audio Effects*, Trondheim, Norway, Nov. 30 – Dec. 3 2015, vol. 18.

[48] D. Fränken et al., "Generation of wave digital structures for connection networks containing ideal transformers," in *Proc. Int. Symp. Circuits Systems (ISCAS)*, May 25–28 2003, vol. 3, pp. 240–243.

[49] D. Fränken et al., "Generation of wave digital structures for networks containing multiport elements," *IEEE Trans. Circuits Systems I: Reg. Papers*, vol. 52, no. 3, pp. 586–596, Mar. 2005.

[50] A. Fettweis, "Some principles of designing digital filters imitating classical filter structures," *IEEE Trans. Circuit Theory*, vol. 18, no. 2, pp. 314–316, 1971.

[51] A. Fettweis, "Digital filters structures related to classical filter networks," *Archiv Elektronik Übertragungstechnik (AEÜ, Int. J. Electron. Commun.)*, vol. 25, pp. 79–89, 1971.

[52] A. Fettweis, "Pseudo-passivity, sensitivity, and stability of wave digital filters," *IEEE Trans. Circuit Theory*, vol. 19, no. 6, 1972.

[53] A. Fettweis, "Reciprocity, inter-reciprocity, and transposition in wave digital filters," *Int. J. Circuit Theory Appl.*, vol. 1, no. 4, pp. 323–337, Dec. 1973.

[54] A. Fettweis and K. Meerkötter, "On adaptors for wave digital filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 23, no. 6, 1975.

[55] C.-W. Ho et al., "The modified nodal approach to network analysis," *IEEE Trans. Circuits Systems*, vol. 22, no. 6, pp. 504–509, 1975.

[56] J. Vlach and K. Singhal, *Computer methods for circuit analysis and design*, Springer, 1983.

[57] G. Borin et al., "Elimination of delay-free loops in discrete-time models of nonlinear acoustic systems," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 5, pp. 597–605, 2000.

[58] D. T.-M. Yeh et al., "Automated physical modeling of nonlinear audio circuits for real-time audio effects—part I: Theoretical development," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 4, pp. 728–737, 2010.

[59] D. T.-M. Yeh, "Automated physical modeling of nonlinear audio circuits for real-time audio effects—part II: BJT and vacuum tube examples," *IEEE Trans. Audio, Speech, Language Process.*, vol. 20, no. 4, pp. 1207–1216, 2012.

[60] S. S. Lawson, "On a generalization of the wave digital filter concept," *Int. J. Circuit Theory Appl.*, vol. 6, no. 2, pp. 107–120, 1978.

[61] S. Bilbao, *Wave and Scattering Methods for Numerical Simulation*, John Wiley & Sons, New York, July 2004.

[62] Electrosmash, "Bi Muff Pi analysis," `http://www.electrosmash.com/big-muff-pi-analysis`.

[63] G. Kubin, "Wave digital filters: Voltage, current, or power waves?," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Tampa, FL, Apr. 1985, vol. 10, pp. 69–72.

[64] U. Zölzer, Ed., *DAFX: Digital Audio Effects*, John Wiley & Sons, second edition, 2011.

SIMULATIONS OF NONLINEAR PLATE DYNAMICS: AN ACCURATE AND EFFICIENT MODAL ALGORITHM

Michele Ducceschi, *

Acoustics and Audio Group,
University of Edinburgh,
Edinburgh, UK
v1mducce@exseed.ed.ac.uk

Cyril Touzé,

IMSIA, ENSTA ParisTech, CNRS, CEA, EDF,
Université Paris-Saclay,
Palaiseau Cedex, France
cyril.touze@ensta-paristech.fr

ABSTRACT

This paper presents simulations of nonlinear plate vibrations in relation to sound synthesis of gongs and cymbals. The von Kármán equations are shown and then solved in terms of the modes of the associated linear system. The modal equations obtained constitute a system of nonlinearly coupled Ordinary Differential Equations which are completely general as long as the modes of the system are known. A simple second-order time-stepping integration scheme yields an explicit resolution algorithm with a natural parallel structure. Examples are provided and the results discussed.

1. INTRODUCTION

Nonlinear vibrations of plates constitute a large domain of research which embraces fields ranging from mechanical and civil engineering, to physics and sound synthesis [1, 2, 3, 4, 5]. The latter is the subject of interest in this work, because plates are, geometrically speaking, the simplest kind of *idiophone* although they share the same salient dynamical features of idiophones of more complex shape (bells, shells, gongs). Hence, numerical simulations of plates offer appealing possibilities for sound synthesis with direct applications to the sound of gongs and cymbals. However, because of the complexity of the nonlinear dynamics, time-domain simulations have become a feasible possibility only in recent years, when compared to sound synthesis of simpler systems such as linear bars [6] and plates [7]. The first complete study on full time-domain simulations of the classic nonlinear plate equations (the von Kármán equations) is due to Bilbao [8]. In this work a rectangular plate with simply-supported conditions is considered and a solution given in terms of a stable, second-order, implicit Finite Difference scheme. The stability condition is achieved through numerical conservation of the energy of the system. Subsequent work by Bilbao and Torin extended the case of a single plate to a full 3D multi-plate environment, where the plates are coupled through pressure waves in air [9]. On the other hand, a modal approach was proposed by Chadwick [10] to treat the case of shells. More recently, a modal approach was proposed by these authors to solve the von Kármán plate equations [11]. Such model is based on the projection of the original system onto the linear modes. Time integration is performed using the time-stepping scheme borrowed from Bilbao [8] and adapted to the modal equations so to give energy conservation mode by mode. Such approach, despite not being *per se* "better" than Finite Differences, has anyhow proven useful for a number of reasons, namely

- for particular combinations of geometry and boundary conditions, the modes are known analytically and therefore the coupling coefficients and eigenfrequencies can be calculated with very high precision (the case of a circular plate with a free boundary, notoriously difficult to treat using Finite Differences, falls into this category);
- damping can be implemented mode by mode resulting resulting in a much more accurate (frequency-dependent) representation of losses.

In turn, the modal approach should be regarded as a practical alternative to Finite Differences in the case of plates. In this work, the modal approach is described and some applications shown. With respect to [11], the time-stepping scheme selected in the present work is a simpler Störmer-Verlet scheme instead of the more complex implicit scheme. The simplification obtained through this choice comes at a high cost: the *stability* of the resulting numerical algorithm. It is, of course, the case to stress the importance of numerical stability when dealing with sound synthesis routines: as a general rule, an algorithm design unable to guarantee stability should be frowned upon.

For the case of the nonlinear modal equations in this work, however, an exception will be made. In fact, stable modal and Finite Difference algorithms have been extensively studied in the aforementioned works, and therefore the focus in this paper will be on *efficiency* rather than stability. In addition, the Störmer-Verlet scheme is perfectly adapted to the problem of the *linear* plate, being energy conserving and stable. The choice of this scheme is justified on the basis that

- the resulting algorithm is *explicit* and therefore does not require to solve a linear system of algebraic equations;
- for the modal equations, the calculation of the nonlinear term can be *highly parallelised* thus allowing fast computations.

The paper is composed as follows: in section 2 the equations are presented along with the modal approach. Section 3 presents the Störmer-Verlet integration scheme and resulting algorithm. Section 4 presents two case studies; convergence of the coupling coefficients is shown. Section 5 discusses in more detail the simulations obtained with scheme.

2. MODEL EQUATIONS

The literature presents a number of model equations describing the nonlinear behaviour of plates. Such models rely on different assumptions which simplify the complexity of the system [1]. In

* This work was supported by the *Royal Society* and the *British Academy* through a *Newton International Fellowship*

general, the nonlinearity of a plate is of geometric type, meaning that nonlinear effects come into existence when the amplitude of vibrations increases above a reference amplitude. In actual fact, for such high amplitudes, flexural displacements must entail some kind of in-plane motion which results in a coupling of the modes of vibrations. Amongst the many possible models that describe such dynamics, the von Kármán model represents a particularly attractive choice. Such model, in fact, describes with high degree of accuracy the nonlinear dynamics of plates (at least for vibrations up to a few times the thickness), despite a relatively straightforward extension of the underlying linear model (the Kirchhoff plate equation). In the von Kármán model, the in-plane displacement is originated by a second order correction to the linear strain tensor [12]. Such correction results in a modal coupling for the flexural modes. The von Kármán equations for the flexural displacement $w(\mathbf{x}, t)$ are written as

$$\rho h \ddot{w} + D\Delta\Delta w = \mathcal{L}(w, F) + p(\mathbf{x}, t) - R(\dot{w}),$$
$$\Delta\Delta F = -\frac{Eh}{2}\mathcal{L}(w, w),$$

where ρ is the material volume density, h the plate thickness, and D stands for flexural rigidity: $D = Eh^3/12(1-\nu^2)$, with E and ν respectively Young modulus and Poisson ratio. Δ represents the two-dimensional Laplacian operator, while $p(\mathbf{x}, t)$ stands for the normal external forcing, and $R(\dot{w})$ is a generic expression for the viscous damping depending on the velocity field. The function $F(\mathbf{x}, t)$ is known as *Airy stress function* and it accounts for the in-plane displacement. The operator $\mathcal{L}$ is generally referred to as the *von Kármán operator* or *Monge-Ampère form* in the literature and may be expressed in intrinsic coordinates, for two functions $f(\mathbf{x})$ and $g(\mathbf{x})$, as [13]

$$\mathcal{L}(f, g) = \Delta f \Delta g - \boldsymbol{\nabla}\boldsymbol{\nabla} f : \boldsymbol{\nabla}\boldsymbol{\nabla} g,$$

where : denotes the doubly contracted product of two tensors.

2.1. Modal approach

The von Kármán equations are now solved formally using a *modal approach*. This means that both the transverse and in-plane functions are expanded onto a series of base functions corresponding to the modes of the associated linear system (which form therefore a complete set over the domain of the plate). Note that the following derivation is independent of the choice of the *geometry of the plate* and of the *boundary conditions*. Let $\{\Phi_k(\mathbf{x})\}_{k\geq 1}$ be the eigenmodes of the transverse displacement. These functions are the solutions of the Sturm-Liouville eigenvalue problem

$$\Delta\Delta\Phi_k(\mathbf{x}) = \frac{\rho h}{D}\omega_k^2 \Phi_k(\mathbf{x}), \tag{2}$$

together with the associated boundary conditions. In Eq. (2), ω_k stands for the k^{th} radian eigenfrequency. The linear modes are defined up to a constant of normalisation that can be chosen arbitrarily. For the sake of generality, S_w denotes the constant of normalisation of the function $\bar{\Phi} = S_w \frac{\Phi_k(\mathbf{x})}{\|\Phi_k\|}$. The norm is obtained from a scalar product $< \alpha, \beta >$ between two functions $\alpha(\mathbf{x})$ and $\beta(\mathbf{x})$, defined as

$$< \alpha, \beta >= \int_S \alpha\, \beta \, \mathrm{d}S \qquad \longrightarrow \qquad \|\Phi_k\|^2 =< \Phi_k, \Phi_k >,$$

where S represents the area of the plate.

The eigenmodes for the Airy stress function are denoted as $\{\Psi_k(\mathbf{x})\}_{k\geq 1}$. They satisfy the following eigenvalue problem

$$\Delta\Delta\Psi_k(\mathbf{x}) = \zeta_k^4 \Psi_k(\mathbf{x}), \tag{3}$$

together with the associated boundary conditions for F. The linear modes so defined are orthogonal with respect to the scalar product, and are therefore a suitable function basis [14]. Orthogonality between two functions $\Lambda_m(x, y), \Lambda_n(x, y)$ is expressed as

$$< \Lambda_m, \Lambda_n >= \delta_{m,n}\|\Lambda_m\|^2,$$

where $\delta_{m,n}$ is Kronecker's delta.

The Partial Differential Equations (1) for the perfect plate are discretised by expanding the two unknowns w and F along their respective eigenmodes

$$w(\mathbf{x}, t) = S_w \sum_{k=1}^{N_\Phi} \frac{\Phi_k(\mathbf{x})}{\|\Phi_k\|} q_k(t),$$
$$F(\mathbf{x}, t) = S_F \sum_{k=1}^{N_\Psi} \frac{\Psi_k(\mathbf{x})}{\|\Psi_k\|} \eta_k(t),$$

where $q_k(t)$ and $\eta_k(t)$ represent respectively the modal transverse displacement and the modal coordinate for the Airy stress function. The integers N_Φ and N_Ψ are intended to be *finite* for any numerical simulation. Following a standard projection technique and using the orthogonality relationship leads to discretisation of system (1) [15]. The discretised von Kármán equations read

$$\ddot{q}_s + \omega_s^2 q_s + 2\xi_s\omega_s\dot{q}_s = $$
$$-\frac{ES_w^2}{\rho}\sum_{k,m,n}^{N_\Phi}\left[\sum_{l=1}^{N_\Psi}\frac{H_{m,n}^l E_{k,l}^s}{2\zeta_l^4}\right] q_k q_m q_n + p_s(t), \tag{5}$$

The tensors appearing in Eq. (5) can be rewritten as

$$H_{i,j}^k = \frac{\int_S \Psi_k \mathcal{L}(\Phi_i, \Phi_j)\mathrm{d}S}{\|\Psi_k\|\,\|\Phi_i\|\,\|\Phi_j\|}$$
$$E_{i,j}^s = \frac{\int_S \Phi_s \mathcal{L}(\Phi_i, \Psi_j)\mathrm{d}S}{\|\Phi_s\|\,\|\Phi_i\|\,\|\Psi_j\|}.$$

and expresses the nonlinear coupling between in-plane and transverse motions. Note that in Eq. (5), the modal external force has been expressed as

$$p_s(t) = \frac{1}{\rho h S_w \|\Phi_s\|}\int_S p(\mathbf{x}, t)\Phi_s(\mathbf{x})\mathrm{d}S.$$

In general, one may choose a separable form for $p(\mathbf{x}, t)$ and consider a Dirac delta for the space component (corresponding to pointwise forcing). Hence

$$p(\mathbf{x}, t) = \delta(\mathbf{x} - \mathbf{x}_0)P(t); \rightarrow p_s(t) = \frac{\Phi_s(\mathbf{x}_0)}{\rho h S_w \|\Phi_s\|}P(t).$$

Note also that in the same equation a modal damping term has been introduced as $2\xi_s\omega_s\dot{q}_s$. This choice is justified as metallic plates are usually only slightly damped [16].

Eq. (5) may be further simplified by introducing the fourth-order tensor $\Gamma_{j,k,l}^s$ as [13, 15]

$$\Gamma_{j,k,l}^s = \sum_{r=1}^{N_\Psi}\frac{H_{k,l}^r E_{j,r}^s}{2\zeta_r^4}. \tag{6}$$

Remarks. System (5) is composed of N_Φ coupled Ordinary Differential Equations (ODEs). The left-hand side is the equation of a classic harmonic oscillator with loss, whereas the right-hand side is composed of a generic forcing term $p_s(t)$ plus the nonlinear coupling. This coupling involves the product of three modal coordinates and therefore for small amplitudes of vibrations (say, $w \ll h$) it becomes negligible. When $w \approx h$ the nonlinear term is responsible for weakly nonlinear phenomena, notably amplitude-dependent frequency of vibration which, thinking from the standpoint of sound synthesis, is perceived as a *pitch-bend*. Finally, when $w > h$ nonlinearities dominate the dynamics acting as a strong forcing term. Dynamically, this regime is characterised by a cascade of energy from large to small wavelengths, whose properties constitute a topic of research on its own within the realm of Wave Turbulence [4, 17, 18, 19, 20]. Typically, the cascade is perceived as a crash or a shimmering sound, and is one of the most dramatic examples of nonlinear phenomena in musical acoustics. Remarkably, these three regimes are very well described by the von Kármán equations, despite the absence of in-plane inertia. For a study on the effect of such inertia term in the von Kármán system, one may refer to [21].

3. TIME INTEGRATION

Time integration of system (5) is performed using a Störmer-Verlet scheme. For that, time is discretised according to a *sampling rate* $fs = 1/k$, where k is the time-step. Discrete time operators are now introduced. The most obvious operator is the identity operator, denoted by

$$1\mathbf{q}^n = \mathbf{q}^n.$$

Note that the notation $\mathbf{q}^n$ indicates that the vector $\mathbf{q}$ is evaluated at the discrete time kn (it must not be confused with an exponent). The backward and forward shift operators are, respectively,

$$e_{t-}\mathbf{q}^n = \mathbf{q}^{n-1}; \qquad e_{t+}\mathbf{q}^n = \mathbf{q}^{n+1}.$$

Backward, centered and forward approximations to first time derivative are defined as

$$\delta_{t-} \equiv \frac{1}{k}(1 - e_{t-}); \delta_{t\cdot} \equiv \frac{1}{2k}(e_{t+} - e_{t-}); \delta_{t+} \equiv \frac{1}{k}(e_{t+} - 1).$$

An approximation to the second time derivative can be constructed by combining the previous operators. In such a way, a particular form employed here is given by

$$\delta_{tt} \equiv \delta_{t+}\delta_{t-} = \frac{1}{k^2}(e_{t+} - 2 + e_{t-})$$

The Störmer-Verlet scheme is

$$\delta_{tt}\mathbf{q}^n + \mathbf{K}\mathbf{q}^n + \mathbf{C}\delta_{t\cdot}\mathbf{q}^n = -\mathbf{nl}^n + \mathbf{p}^n.$$

In the equation above, the matrices $\mathbf{K}$, $\mathbf{C}$ denote the normalised stiffness and damping matrices (independent of the step n). In practice, if N_Φ denoted the length of the vectors, then these matrices are $N_\Phi \times N_\Phi$; the two matrices are diagonal

$$K_{m,m} = \omega_m^2; \qquad C_{m,m} = 2\xi_m\omega_m. \tag{7}$$

The vector $\mathbf{nl}$ is the vector of the nonlinear terms, acting as a coupling. This is, simply

$$nl_m^n = \frac{ES_w^2}{\rho}\sum_{j,k,l=1}^{N_\Phi}\Gamma_{j,k,l}^m q_j^n q_k^n q_l^n. \tag{8}$$

Finally, $\mathbf{p}$ is the vector containing the forcing terms, *i*.e.

$$p_m^n = \frac{\Phi_m(\mathbf{x}_0)}{\|\Phi_m\|\rho h S_w}P^n.$$

Developing the discrete operators gives the following algebraic system, to be solved for the variable $\mathbf{q}^{n+1}$

$$\left(\frac{\mathbb{I}_{N_\Phi}}{k^2} + \frac{\mathbf{C}}{2k}\right)\mathbf{q}^{n+1} = \left(\frac{2\mathbb{I}_{N_\Phi}}{k^2} - \mathbf{K}\right)\mathbf{q}^n + \mathbf{p}^n - \mathbf{nl}^n + \left(\frac{\mathbf{C}}{2k} - \frac{\mathbb{I}_{N_\Phi}}{k^2}\right)\mathbf{q}^{n-1}, \tag{9}$$

where $\mathbb{I}_{N_\Phi}$ is the $N_\Phi \times N_\Phi$ identity matrix.

Remarks. Scheme (9) is *second-order accurate* and *explicit*: the matrices multiplying the vectors are diagonal and therefore each line in the system can be solved independently (without requiring a linear system solver). This might seem puzzling at a first glance because (5) is a system of *coupled* equations. Indeed, the update of the modal scheme is composed of not only Eq. (9), but also of Eq. (8): this is where coupling happens at each update.

More generally, the Störmer-Verlet scheme is *symmetric* and *symplectic*, and is thus volume-preserving in phase space for Hamiltonian flows [22].

The associated discrete *linear* system is conservative under the Störmer-Verlet algorithm. In practice, by setting to zero the forcing ($\mathbf{p}^n$), damping ($\mathbf{C}$) and nonlinear term ($\mathbf{nl}^n$) in scheme (9), the following relation can be derived [23]

$$\delta_{t+}\sum_{s=1}^{N_\Phi} S_w^2\frac{\rho h}{2}\left[(\delta_{t-}q_s^n)^2 + \omega_s^2 q_s^n\,(e_{t-}q_s^n)\right] = 0,$$

or

$$\delta_{t+}\sum_{s=1}^{N_\Phi}(\tau_s^n + \upsilon_s^n) = 0$$

which corresponds to

$$\frac{d}{dt}\sum_{s=1}^{N_\Phi} S_w^2\frac{\rho h}{2}\left[\dot{q}_s^2(t) + \omega_s^2 q_s^2(t)\right] = 0.$$

or

$$\frac{d}{dt}\sum_{s=1}^{N_\Phi}(T_s + U_s) = 0.$$

for the continuous case. In practice, τ_s^n and υ_s^n are the corresponding discrete kinetic and potential energies of the linear plate, which can be seen as the sum of uncoupled harmonic oscillators.

Conservation of discrete energy allows to derive a stability condition for the scheme applied to the linear plate equation. Let $\hat{\omega}_s$ denote the largest eigenfrequency of the system, then the scheme is stable when [23]

$$\hat{k} < \frac{2}{\hat{\omega}_s}. \tag{10}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \dots & \vdots \\ a_{M1} & a_{M2} & a_{M3} & \dots & a_{MN} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_M \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{pmatrix}$$

Figure 1: Parallel structure of update (8) for the nonlinear term. Each colour represents an operation that can be solved independently in a parallel environment.

When this condition is enforced, then the discrete energies are positive definite and the scheme is stable (*i.e.* the solutions are bounded over time). This *same* condition can be extended to the nonlinear system when the time integration scheme proposed in [8] is applied. The resulting algorithm, however, becomes implicit and the computational time increases.

A second remark has to do with the efficiency of the scheme. Most time is spent in updating the nonlinear term, (8). However, such an update presents a highly parallel structure, being basically a series of products of matrices (see fig. 1). In MATLAB, matrix multiplication is automatically assigned to multiple cores when possible, but in theory many more (and faster) possibilities are open to anyone wishing to port the modal code to other languages (C, CUDA, etc.).

4. CASE STUDIES

System (5) is completely general, meaning that the integration scheme (9) can be applied as long as the coupling coefficients (6) are known together with the radian frequencies in Eq. (2). In general, analytic solutions are not available for given geometry and boundary conditions, but there are important cases which constitute an exception. Two of such cases are presented now: a circular plate with free boundary and a rectangular plate with simply supported boundary.

4.1. Free Circular Plate

A circular plate of radius a with a free edge is first considered. The boundary conditions then read, for the two unknowns $w(r,\theta,t)$ and $F(r,\theta,t)$ [24] (an index after a comma indicates a derivative in that direction)

$$\begin{aligned} &\forall t,\ \forall\theta\in[0,2\pi],\ \text{at}\quad r=a: \\ &\quad w_{,rr}+\frac{\nu}{a}w_{,r}+\frac{\nu}{a^2}w_{,\theta\theta}=0\,, \\ &\quad w_{,rrr}+\frac{1}{a}w_{,rr}-\frac{1}{a^2}w_{,r}+\frac{2-\nu}{a^2}w_{,r\theta\theta}-\frac{3-\nu}{a^3}w_{,\theta\theta}=0\,, \\ &\quad F_{,r}+\frac{1}{a}F_{,\theta\theta}=0\,,\quad F_{,r\theta}+\frac{1}{a}F_{,\theta}=0\,. \end{aligned}$$

Despite a seemingly complex form of the boundary conditions, an analytical solution exists in the form of Bessel functions. The eigenfrequencies are then obtained as the zeros of such functions. The eigenfunctions are here denoted by either a single integer number p - sorting the frequencies from small to large - or by a pair (k_c, k_d), where k_c denotes the number of nodal diameters and k_d the number of nodal circles. As it is usual with circular symmetry, asymmetric modes with $k_c \neq 0$ are degenerated so that two eigenvectors are found for the same eigenfrequency. For the circular plate, eigenfrequencies for both the transverse and in-plane problems are analytic so that the numerical values of $\{\omega_i,\, \zeta_j\}_{i,j\geq 1}$ used to feed the model can be considered exact. In the truncation process, the number of in-plane modes N_Ψ may be selected according to the desired accuracy for the cubic coupling coefficient $\Gamma^p_{p,p,p}$ defined in Eq. (6), see *e.g.* [24, 15, 25]. As explained in [24], some specific rules exist, so that, for a given transverse mode p, only a few in-plane modes participate with a non-vanishing contribution to the summation in Eq. (6). The rules are as follows:

- For a purely axisymmetric mode $\Phi_{(0,k_d)}$, only the axisymmetric in-plane modes $\{\Psi_{(0,i)}\}_{i\geq 1}$ participate to the summation.
- For an asymmetric mode $\Phi_{(k_c,k_d)}$ with $k_c \neq 0$, then the coupling involve only axisymmetric in-plane modes $\{\Psi_{(0,i)}\}_{i\geq 1}$ as well as asymmetric in-plane modes having twice the number of nodal diameters $\{\Psi_{(2k_c,i)}\}_{i\geq 1}$.

Hence for a given mode p, the convergence of the summation for $\Gamma^p_{p,p,p}$ is achieved within a small subset of all the possible in-plane modes. Let N_Ψ^{conv} be the cardinal of this subset of admissible modes. The convergence of three coefficients $\Gamma^p_{p,p,p}$ is shown in Fig. 2, for three different modes of high frequencies. It is seen that convergence is assured even for such high-order coefficients. An axisymmetric mode with a many nodal circles, mode (0,18) has been selected together with a purely asymmetric one, mode (50,0), and a mixed mode : (24,8). Table 1 presents the converged values of a few nondimensional coefficients. For asymmetric modes (k_c, k_d) with $k_c \neq$0, the modes are degenerate.

CIRCULAR PLATE

mode label p	$\bar{\omega}_p$	$\bar{\Gamma}^p_{p,p,p}$	N_Ψ^{conv}
1,2	1	$1.90\cdot10^{0}$	3
3	1.8	$8.58\cdot10^{0}$	4
4,5	2.3	$1.70\cdot10^{1}$	4
715,716	527.7	$8.44\cdot10^{6}$	65
846	627.6	$2.86\cdot10^{6}$	36
881,882	658.1	$1.78\cdot10^{6}$	50

RECTANGULAR PLATE

mode label p	$\bar{\omega}_p$	$\bar{\Gamma}^p_{p,p,p}$	N_Ψ^{conv}
1	1	$2.00\cdot10^{1}$	12
20	13.9	$9.50\cdot10^{3}$	286
72	48.1	$1.07\cdot10^{5}$	239
336	208.7	$2.50\cdot10^{6}$	25
422	261.5	$5.88\cdot10^{6}$	103
589	361.9	$1.23\cdot10^{7}$	132

Table 1: Convergence of the nondimensional coupling coefficients $\bar{\Gamma}^p_{p,p,p}$ for the circular plate and a rectangular plate with aspect ratio $2/3$. Modes are sorted according to increasing eigenfrequency. N_Ψ^{conv} indicates *upper bounds* of the number of in-plane modes needed for displayed accuracy. The normalised eigenfrequencies $\bar{\omega}_p$ are also shown.

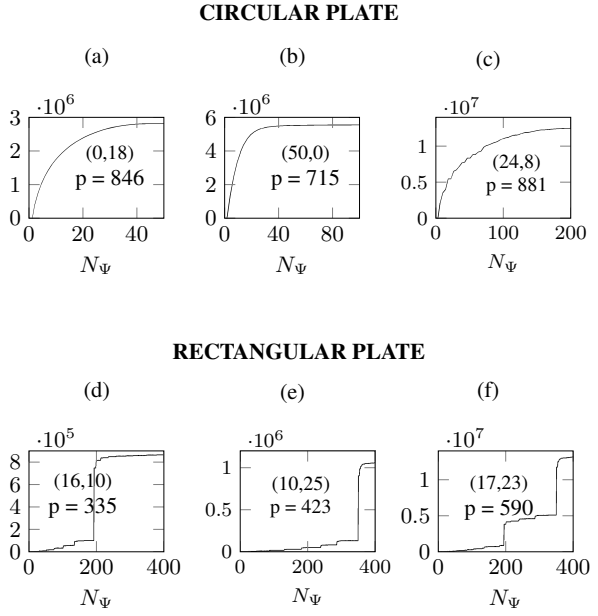


Figure 2: Plots of a few nondimensional values of coupling coefficients, $\bar{\Gamma}^p_{p,p,p}$. (a)-(c): Circular plate. Numbers in brakets are (k_c, k_d) corresponding to nodal circles and diameters. (e)-(f): Rectangular plate of aspect ratio 2/3. Numbers in brakets are (k_1, k_2) of Eq. (12).

4.2. Simply-Supported Rectangular Plate

Consider a rectangular plate of lateral dimensions L_x, L_y. Let n, t denote the normal and tangential directions to the boundary δS. Simply-supported boundary conditions may be given as

$$\begin{aligned} &\forall t,\ \forall \mathbf{x} \in \delta S : \\ &\quad w = w_{,nn} + \nu w_{,tt} = 0, \\ &\quad F_{,nt} = F_{,tt} = 0. \end{aligned} \tag{11}$$

Such conditions, despite not describing a load-free edge (a desirable case for sound synthesis), have the advantage of being particularly simple. The solution for the transverse modes, in fact, is given in terms of sine functions [14]

$$\Phi_k(\mathbf{x}) = \sin\frac{k_1\pi x}{L_x}\,\sin\frac{k_2\pi y}{L_y} \qquad \text{for integers } k_1,\ k_2.$$

The eigenfrequencies are then easily obtained as

$$\omega_k^2 = \frac{D}{\rho h}\left[\left(\frac{k_1\pi}{L_x}\right)^2 + \left(\frac{k_2\pi}{L_y}\right)^2\right]^2. \tag{12}$$

The conditions for the in-plane function, on the other hand, can be worked out to yield a simplified form. Consider in fact the following conditions

$$F = F_{,n} = 0 \quad \forall \mathbf{x} \in \delta S;$$

it is clear that these conditions are sufficient (but not necessary) to satisfy (11) [13]. Such conditions, along with Eq. (3), reduce the quest for the eigenfunctions Ψ_k to the clamped plate problem. Despite not having a closed-form solution, this problem was recently shown to have a semi-analytical solution based on the Rayleigh-Ritz method, yielding $\sim$ 400 eigenfunctions and associated frequencies with precision to, at least, four significant digits [15]. As opposed to the circular case, it is difficult to have an *a priori* knowledge on the coupling rules. This is because the form of the in-plane eigenfunctions is not known analytically, and thus only a numerical investigation can help in laying out coupling rules. As for the circular case, coefficients of the kind $\Gamma^p_{p,p,p}$ are investigated. In general, when either k_1 or k_2 is small, convergence is quite fast, see for example fig. 2(f). However, modes for which neither k_1 or k_2 is small, convergence is much slower and presents a staircase-like behaviour, like in fig. 2(d). Nonetheless, table 1 shows accuracy to a few decimal digits is still assured even for coefficients involving high-frequency modes.

Remarks. Table 1 shows the convergence of the nonlinear coupling coefficients. With respect to other numerical techniques, namely Finite Differences, such order of convergence is out-of-reach for reasonable sample rates [15]. This remark should be kept in mind when comparing the *efficiency* (or numerical burden) of the modal method versus a Finite Difference scheme: for the same degree of accuracy, Finite Differences are much less efficient than modes.

On the other hand, the two case studies presented here are somewhat an exception as they have analytical or semi-analytical solutions for the modes. When facing a case for which the modes are not known, the modal approach becomes somewhat less appealing than other numerical methods.

5. SIMULATIONS

Now that the coupling coefficients are eigenfrequencies have been calculated for the case studies, scheme (9) may be applied. First, excitation and loss are presented, and then results of simulations shown.

5.1. Excitation

For pointwise forcing, two different excitation mechanisms will be considered here: 1. a strike and 2. a sinusoidal forcing.

Strikes are of course the most direct way to excite a plate. In general, one wishes to have control on the "loudness" and "brilliance" of the sound. To a first approximation, these parameters may be controlled by making use of a relatively simple form for the excitation, a raised cosine. This function is defined as

$$P(t) = \begin{cases} \frac{\bar{P}}{2}\left[1 + \cos\left(\pi(t-t_0)/T_{wid}\right)\right] & \text{if} \quad |t-t_0| \le T_{wid}\ ; \\ 0 & \text{if} \quad |t-t_0| > T_{wid}\ . \end{cases}$$

It is seen that one has control over two parameters: T_{wid}, or half of the total contact duration, and $\bar{P}$, the maximum amplitude. As a rule of thumb, one may use a shorter T_{wid} and a larger $\bar{P}$ to increase the "loudness" and "brightness" of the output. See also fig. 3(a).

Sinusoidal forcing (see fig. 3(b)) may instead by applied to observe the system undergo two bifurcations, leading first to a quasi-periodic and then to a turbulent motion. Perceptually, the three regimes are very well recognisable ranging from an almost monochromatic sound to a loud, shimmering cascade.

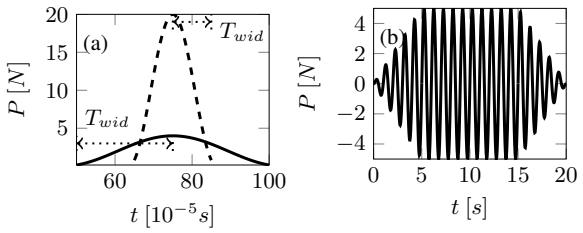


Figure 3: (a): examples of raised cosine functions used to simulate a strike, in the case of "hard" contact (dashed) and "soft" contact (thick). (b): example of sinusoidal forcing of frequency 1Hz and amplitude 5N increased linearly to a steady value and then decreased.

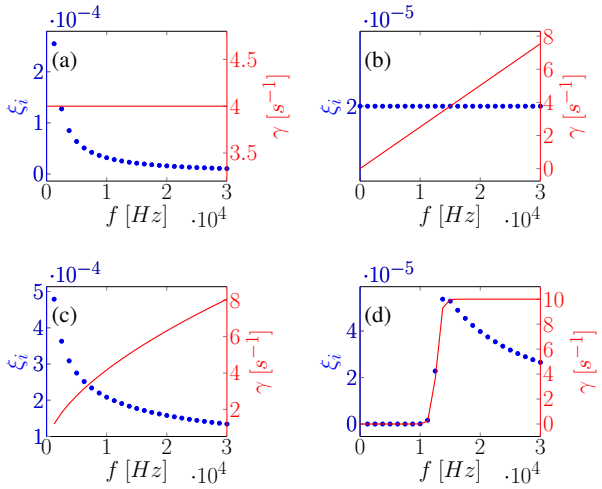


Figure 4: Examples of damping coefficients $\xi_i(\omega)$ (dots) and resulting damping laws $\gamma(\omega) = 2\xi_i\omega_i$ (thick lines). (a): $\xi_i = 2/\omega_i$. (b): $\xi_i = 2 \cdot 10^{-5}$. (c): damping law as measured by Humbert [26] for a plate of thickness $h = 0.5$mm, $\xi_i = 8 \cdot 10^{-3}\omega_i^{-0.4}$. (d): $\xi_i = 5/[\omega_i(\exp(32 - \omega_i/2500) + 1)]$.

5.2. Loss

For sound synthesis, loss bears about a lot of perceptual information. One of the advantages of the modal approach is that a complex decay can be simulated by setting by hand the modal damping coefficients ξ_m in Eq. (7). For example, if one wishes to simulate a damping law which dissipated energy at the same rate at all scales, one may set $\xi_m = c/\omega_m$; a linear damping law is obtained for $\xi_m = c$ for constant c. Of course, the possibilities are endless, see also fig. 4

5.3. Examples

Fig. 6 reports the simulations for three different cases: 1. a soft strike; 2. a hard strike; 3. a sinusoidally forced plate. For nonlinear dynamics, in the case of a strike, the pictorial example of fig. 5 shows one snapshot immediately after a strike and one at later moment when the profile is fully turbulent.

The plate under consideration is a small rectangular plate of

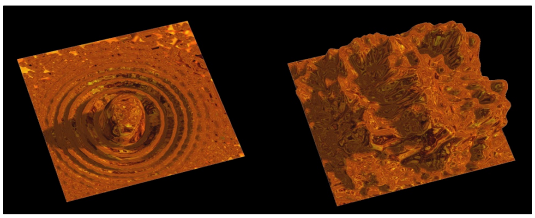


Figure 5: Snapshots of displacement field on top of a plate just after a strike (left) and during fully developed nonlinear state (right), with the presence of large and small wavelengths.

dimensions 21cm$\times$ 18cm $\times$ 1mm. For this plate, the eigenfrequencies are such that $\omega_{250}/2\pi = 21610$Hz, suggesting that even a relatively small number of modes suffices to reproduce a rich sound spanning a large interval of the audible range. The output can be extracted by recording the displacement at one point on the plate, w_0^n, and by taking a discrete time derivative in order to high-pass the frequency spectrum.

Because instability may set in, one must set the sampling frequency to a reasonably high rate in order to obtain convergence. The reference sampling rate in this case is given by Eq. (10), which gives the stability condition for the linear plate. Because in this case the dynamics is nonlinear, one has to guess a value of the sampling rate which assures convergence, but which remains sufficiently small in order to allow fast computations. Empirically, it is found that for

$$k_{\text{sim}} = \hat{k}/2$$

the solution converges even for a strongly nonlinear dynamics (at least for vibrations up to $2-3$ times the thickness) and hence such is the time-step selected for the simulations.

Fig. 6(a)-(b) is the typical case of weakly nonlinear vibrations, with amplitude $w \sim h$. To simulate a soft strike using a raised cosine, the particular parameters employed here are $T_{wid} =$ 1ms, $\bar{P} =$ 100N. The damping coefficients are set as $\xi_i = 8 \cdot 10^{-3}\omega_i^{-0.4}$. The total number of modes retained is fairly small, $N_\Phi = 100$, covering frequencies up to 9000Hz. The number of in-plane modes is $N_\Psi = 50$. In this case, the modes are weakly coupled giving rise to pitch-bends and amplitude-dependent frequency of vibration. The damping law selected allows to simulate a very enjoyable decay, with complex auditory cues.

Fig. 6(c)-(d) represents the case of strong nonlinear dynamics. The plate is activated by a raised cosine with $T_{wid} = 0.8$ms, $\bar{P} =$ 300N. Damping coefficients and number of modes are the same as for the previous case. Here, the amplitude of vibrations has a maximum at $2h$, indicating that more nonlinear phenomena may set in. In fact, by looking at the velocity spectrogram one can notice that just after the strike the modes are not very well distinguishable: this is a trace of a turbulent dynamics corresponding to a shimmering sound typical of gongs. After this initial transient, loss removes energy from the system until the modes are again perfectly distinguishable, and eventually killed.

Perhaps the most interesting case is represented in fig. 6(e)-(f). In this case the plate is activated by a sinusoid close to the 5^{th} eigenfrequency of the system. The amplitude of the sinusoid is increased from 0 to 116N in 1s, then kept steady for 5s and eventually decreased to zero in 2s. The plate undergoes 2 bifurcations denoted in the spectrogram by thick dashed lines: at the

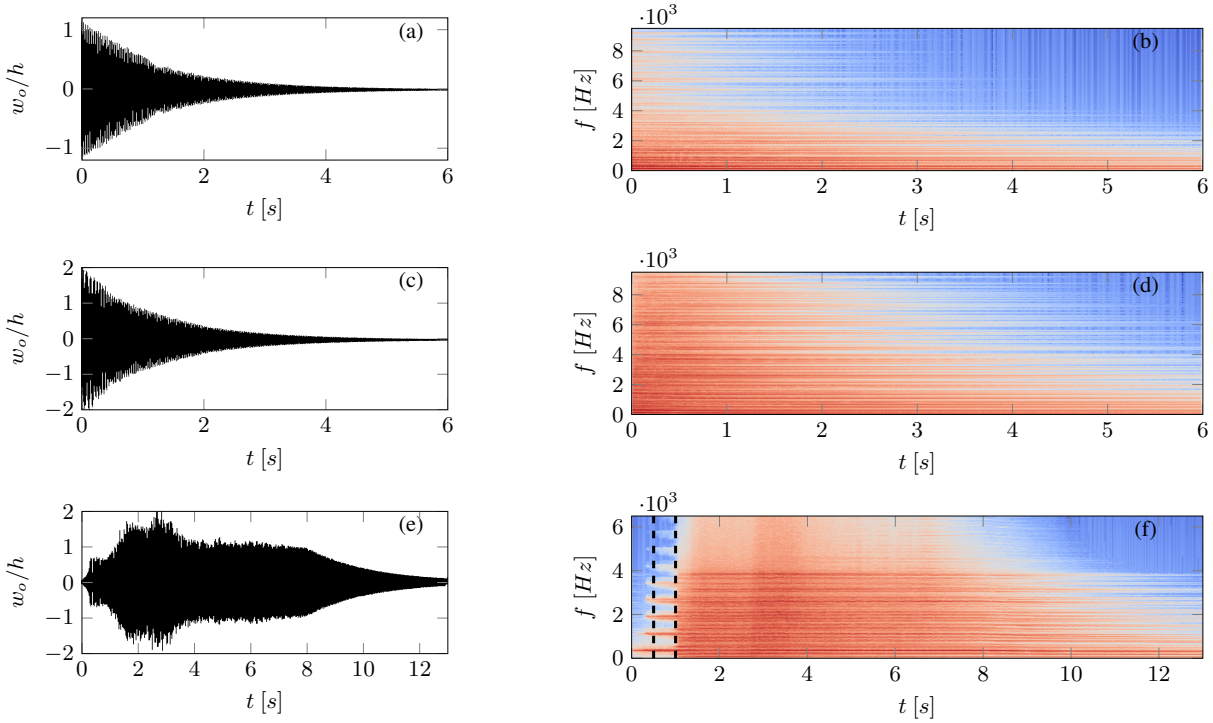


Figure 6: Numerical simulations of a rectangular plate with dimensions 21cm× 18cm × 1mm. (a)-(b): Time series of displacement and velocity spectrogram of "soft" strike, with $\bar{P}$ = 100N, T_{wid} = 1ms, $\xi_i = 8 \cdot 10^{-3}\omega_i^{-0.5}$, $N_\Psi = 50$, $N_\Phi = 100$. (c)-(d): Time series of displacement and velocity spectrogram of "hard" strike, with $\bar{P}$ = 300N, T_{wid} = 0.8ms, $\xi_i = 8 \cdot 10^{-5}\omega_i^{-0.4}$,$N_\Psi = 50$, $N_\Phi = 100$. (e)-(f): Time series of displacement and velocity spectrogram of sinusoidally forced plate, with sinusoid of frequency $1.02 f_5$ and maximum amplitude 116N, increased linearly for 1s, kept steady for 5s, and then decreased to zero in 2s; $\xi_i = 4 \cdot 10^{-3}\omega_i^{-0.4}$, $N_\Phi = 72$, $N_\Psi = 60$; dashed lines on spectrogram indicate bifurcations from a linear regime, to a quasi-periodic regime, to a turbulent regime.

start one can clearly hear a monochromatic sound at the selected frequency; after the first bifurcation the same frequency is modulated by higher harmonics; and after the second bifurcation all the modes are activated in a turbulent bath. The damping law selected are able to give the sound a natural richness with complex harmonic relations.

Calculation times in MATLAB are quite fast. For the strikes (*i.e.* $N_\Psi = 50$, $N_\Phi = 100$, $k_{\text{sim}} = \hat{k}/2$) the calculation time is about 8 times real-time, on a machine equipped with an Intel i7 CPU at 2.40GHz. For the sinusoid (*i.e.* $N_\Psi = 50$, $N_\Phi = 60$, $k_{\text{sim}} = \hat{k}/2$) the calculation time is about 1.5 times real-time.

6. CONCLUSIONS

This work presented an explicit modal scheme for the nonlinear plate equations. For the cases which present a semi-analytic solution for the modes, it was shown that the eigenfrequencies and coupling coefficients can be calculated to a very high precision. The modal update is completely general as long as such frequencies and coefficients are known. Two case studies were presented, including the important case of a circular plate with a free boundary. Numerical simulations were provided to show that the modal scheme can simulate efficiently a complex dynamics, resulting in very realistic sound synthesis.

7. REFERENCES

[1] A. H. Nayfeh, *Nonlinear interactions: analytical, computational and experimental methods*, Wiley series in nonlinear science, New-York, 2000.

[2] M. Amabili, *Nonlinear vibrations and stability of shells and plates*, Cambridge University Press, 2008.

[3] C. Touzé, S. Bilbao, and O. Cadot, "Transition scenario to turbulence in thin vibrating plates," *Journal of Sound and Vibration*, vol. 331, no. 2, pp. 412–433, 2012.

[4] G. Düring, C. Josserand, and S. Rica, "Weak turbulence for a vibrating plate: Can one hear a Kolmogorov spectrum?," *Physical Review Letters*, vol. 97, pp. 025503, 2006.

[5] S. Bilbao, *Numerical Sound synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, Wiley, 2009.

[6] A. Chaigne and V. Doutaut, "Numerical simulations of xylophones. i. time-domain modeling of the vibrating bars," *The Journal of the Acoustical Society of America*, vol. 101, no. 1, 1997.

[7] C. Lambourg, A. Chaigne, and D. Matignon, "Time-domain simulation of damped impacted plates. II: Numerical model and results," *The Journal of the Acoustical Society of America*, vol. 109, no. 4, 2001.

[8] S. Bilbao, "A family of conservative finite difference schemes for the dynamical von Kármán plate equations," *Numerical Methods for Partial Differential Equations*, vol. 24, no. 1, pp. 193–216, 2007.

[9] A. Torin and S. Bilbao, "A 3D multi-plate environment for sound synthesis," in *Proceedings of the 16th International Conference on Digital Audio Effects*, 2013.

[10] J. Chadwick, S. An, and D. James, "Harmonic shells: a practical nonlinear sound model for near-rigid thin shells," *ACM Transactions on Graphics (SIGGRAPH ASIA Conference Proceedings)*, vol. 28, no. 5, pp. Article 119, 2009.

[11] M. Ducceschi and C. Touzé, "Modal approach for nonlinear vibrations of damped impacted plates: Application to sound synthesis of gongs and cymbals," *Journal of Sound and Vibration*, vol. 344, pp. 313 – 331, 2015.

[12] L.D. Landau and E.M. Lifschitz, *Theory of Elasticity, Third Edition*, Elsevier Butterworth Heinemann, 1986.

[13] O. Thomas and S. Bilbao, "Geometrically nonlinear flexural vibrations of plates: In-plane boundary conditions and some symmetry properties," *Journal of Sound and Vibration*, vol. 315, no. 3, pp. 569–590, 2008.

[14] P. Hagedorn and A. DasGupta, *Vibrations and Waves in Continuous Mechanical Systems*, Wiley, 2007.

[15] M. Ducceschi, C. Touzé, S. Bilbao, and C.J. Webb, "Nonlinear dynamics of rectangular plates: investigation of modal interaction in free and forced vibrations," *Acta Mechanica*, vol. 225, no. 1, pp. 213–232, 2014.

[16] J. Woodhouse, "Linear damping models for structural vibration," *Journal of Sound and Vibration*, vol. 215, no. 3, pp. 547 – 569, 1998.

[17] A. Boudaoud, O. Cadot, B. Odille, and C. Touzé, "Observation of wave turbulence in vibrating plates," *Physical Review Letters*, vol. 100, pp. 234504, 2008.

[18] N. Mordant, "Are there waves in elastic wave turbulence ?," *Physical Review Letters*, vol. 100, pp. 234505, 2008.

[19] P. Cobelli, P. Petitjeans, A. Maurel, V. Pagneux, and N. Mordant, "Space-time resolved wave turbulence in a vibrating plate," *Physical Review Letters*, vol. 103, pp. 204301, 2009.

[20] M. Ducceschi, O. Cadot, C. Touzé, and S. Bilbao, "Dynamics of the wave turbulence spectrum in vibrating plates: A numerical investigation using a conservative finite difference scheme," *Physica D*, vol. 280-281, pp. 73–85, 2014.

[21] S. Bilbao, O. Thomas, C. Touzé, and M. Ducceschi, "Conservative numerical methods for the full von Kármán plate equations," *Numerical Methods for Partial Differential Equations*, (accepted for publication) 2015.

[22] E. Hairer, C. Lubich, and G. Wanner, *Geometric numerical integration: structure-preserving algorithms for Ordinary differential equations*, Springer, 2006, second edition.

[23] M. Ducceschi, *Nonlinear vibrations of thin rectangular plates. A numerical investigation with application to wave turbulence and sound synthesis*, Ph.D. thesis, Ecole doctorale de l'Ecole Polytechnique, 2014.

[24] C. Touzé, O. Thomas, and A. Chaigne, "Asymmetric nonlinear forced vibrations of free-edge circular plates, part I: theory," *Journal of Sound and Vibration*, vol. 258, no. 4, pp. 649–676, 2002.

[25] C. Touzé, M. Vidrascu, and D. Chapelle, "Direct finite element computation of non-linear modal coupling coefficients for reduced-order shell models," *Computational Mechanics*, vol. 54, no. 2, pp. 567 – 580, 2014.

[26] T. Humbert, O. Cadot, G. Düring, C. Josserand, S. Rica, and C. Touzé, "Wave turbulence in vibrating plates: The effect of damping," *EPL (Europhysics Letters)*, vol. 102, no. 3, pp. 30002, 2013.

AN ALGORITHM FOR A VALVED BRASS INSTRUMENT SYNTHESIS ENVIRONMENT USING FINITE-DIFFERENCE TIME-DOMAIN METHODS WITH PERFORMANCE OPTIMISATION

Reginald L Harrison, Stefan Bilbao
Acoustics and Audio Group,
University of Edinburgh
Edinburgh, UK
r.l.harrison-3@sms.ed.ac.uk *
s.bilbao@ed.ac.uk

James Perry
Edinburgh Parallel Computing Centre,
University of Edinburgh
Edinburgh, UK
j.perry@epcc.ed.ac.uk

ABSTRACT

This paper presents a physical modelling sound synthesis environment for the production of valved brass instrument sounds. The governing equations of the system are solved using finite-difference time-domain (FDTD) methods and the environment is implemented in the C programming language. Users of the environment can create their own custom instruments and are able to control player parameters such as lip frequency, mouth pressure and valve openings through the use of instrument and score files.
The algorithm for sound synthesis is presented in detail along with a discussion of optimisation methods used to reduce run time. Binaries for the environment are available for download online for multiple platforms.

1. INTRODUCTION

The problem of synthesis of brass instrument sounds, such as those from the instrument shown in Fig. 1, has been an active research topic within the field of physical modelling and there are various avenues of approach. Modal methods are used, e.g., in the MoReeSC [1] software package. Digital waveguide methods model wave propagation using delay lines, where the effects of loss and dispersion are lumped into terminating filters [2, 3, 4]. A combination of methods that involves parallel convolution and modal methods along with one way nonlinear wave propagation have also been successful [5, 6].
Although these are all efficient methods of synthesis, the assumptions made to improve performance can lead to awkward implementation of time varying tube configurations—as is the case for articulated valved instrument sounds. More general numerical methods, such as finite difference time domain techniques (FDTD), can be used in this case and are suitable for modelling the valved brass instrument system [7, 8]. Although computationally more intensive, FDTD algorithms for brass instrument synthesis can be run on modern personal computers and, with the use of optimisation methods in the C programming language, simulation times can approach or surpass real-time.
After the model equations for the valved brass instrument system are described in Section 2, the FDTD approximations to continuous operators are introduced in Section 3 and then used in Section 4 to create discrete versions of the system equations and their corresponding updates. Optimisation techniques in the C programming environment are discussed along with benchmarking times in Section 5 and finally concluding remarks and planned extensions to the environment are presented in Section 6. Supplementary materials including the environment binaries, example sounds and test files are available for download at
http://www2.ph.ed.ac.uk/~s0916351/dafx15.html.

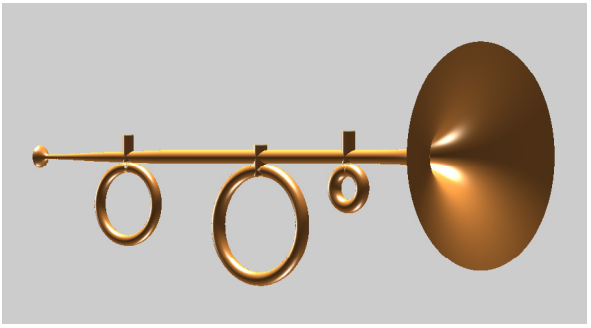


Figure 1: Functional representation of a valved brass instrument. Loops below the main instrument bore are the additional lengths of tubing that airflow can be diverted into through the use of pistons.

2. BRASS INSTRUMENT SYSTEM

A typical valved brass instrument can be separated, functionally, into three components: the input (excitation mechanism), resonator (instrument bore and valve sections), and radiation (interaction of the bore with the surrounding acoustic space). Waves propagate within the instrument bore which includes additional pieces of tubing that waves can be partitioned into through the use of valves. The input and radiation models can be defined separately and then coupled to the extremities of the instrument bore. This section presents the system equations for each component of the model.

2.1. Wave Propagation

Starting in the frequency domain, a model for wave propagation inside a section of an acoustic tube that includes viscous and thermal losses [9] may be written as

$$\partial_x \left(S\hat{v} \right) = -S\hat{Y}\hat{p}, \quad \partial_x \hat{p} = -\hat{Z}\hat{v} \tag{1}$$

* Author for contact

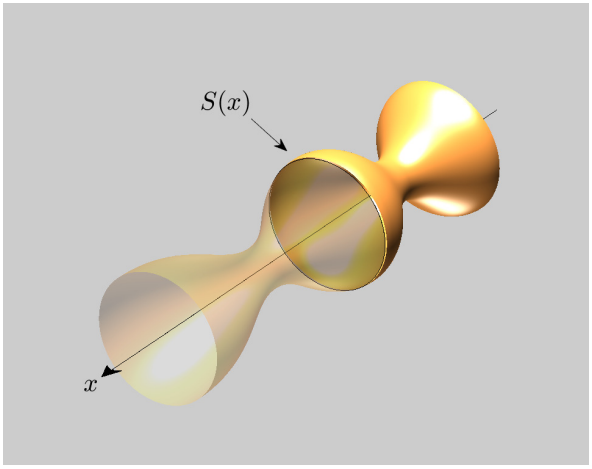


Figure 2: An acoustic tube showing the variation of the surface area, $S(x)$, with axial coordinate x.

where $\hat{p}(x,\omega)$ and $\hat{v}(x,\omega)$ are the acoustic pressure and particle velocity as functions of an angular frequency ω and axial coordinate x, where $x \in [0, L]$ and L is the total length of the instrument. $S(x)$ is the bore cross-section; see Fig. 2. $\hat{Z}(\omega)$ and $\hat{Y}(\omega)$ represent the characteristic series impedance and shunt admittance per unit length of the system that include wave propagation and the viscous and thermal losses. The $\hat{(\cdot)}$ operator denotes a frequency domain function and ∂_x denotes first order partial differentiation with respect to axial coordinate x.

A complete description of the impedance and admittance in equation (1) relies on Bessel functions for tubes of circular cross section. In practical time domain implementations, various approximations are necessary. A typical approximation strategy involves a high frequency approximation leading to a series of fractional powers of $j\omega$ as per, e.g., [10], accompanied by truncation of the series. Transformation to the time domain follows from the replacements $(j\omega)^u \rightarrow \partial_{t^u}$, where ∂_{t^u} is the u^{th} partial derivative with respect to time, t. Equations (1) become

$$\rho\partial_t v + fv + g\partial_{t^{1/2}} v + \partial_x p = 0 \tag{2a}$$

$$\frac{S}{\rho c^2}\partial_t p + q\partial_{t^{1/2}} p + \partial_x (Sv) = 0 \tag{2b}$$

where

$$f = \frac{3\eta\pi}{S}, \quad g = 2\sqrt{\frac{\rho\eta\pi}{S}}, \quad q = \frac{2(\gamma-1)}{\nu c^2}\sqrt{\frac{\eta\pi S}{\rho^3}} \tag{3}$$

and ∂_t and $\partial_{t^{1/2}}$ are first and half order partial derivatives with respect to time. Under further approximations this model reduces to the Webster-Lokshin model [11], as illustrated in [7]. The symbols ρ, c, η, ν^2 and γ are respectively: the density, speed of sound, viscosity, Prandtl number and ratio of specific heats. Values of these constants as a function of temperature are given by Benade [12] and are reprinted by Keefe [10].

2.2. Valve Junctions

In instruments such as the trumpet, valves are employed to direct airflow from the main bore into additional pieces of tubing, the gross

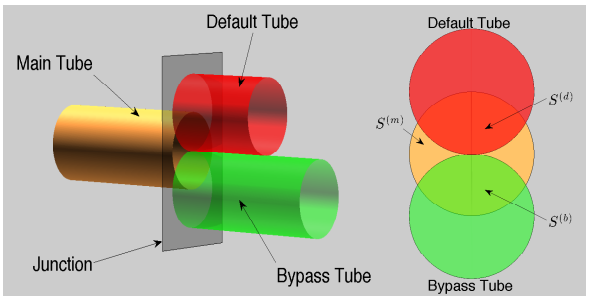


Figure 3: Left: Schematic of a valve junction showing the main tube in copper, the default valve tube in red and the bypass tube in green. The grey rectangle represents the junction surface. Right: Overlap of tubes at valve junction. Notice that in the case of circular tubes, the total overlapped surface between the three tubes will be less than the surface area of the main tube at the junction.

effect of which is to lengthen the instrument and lower its resonances. In normal use players either have the valve fully engaged or not at all, therefore only one possible path is available at a time for waves to propagate within the instrument. It is possible, however, to hold the valve in a partially open configuration, in which case the interaction between three pieces of tubing must be considered which results in more complex instrument resonances and sounds of a multiphonic timbre. Consider the system in Fig. 3, where one piece of tubing, labelled *main*, with cross-sectional surface area $S^{(m)}$ at the junction is overlapping two separate pieces of tubing, labelled *default* and *bypass*, with cross sectional areas $S^{(d)}$ and $S^{(b)}$ at the junction. The surface areas at the junction are defined by

$$S^{(d)} = q^{(d)} S^{(m)}, \quad S^{(b)} = q^{(b)} S^{(m)} \tag{4}$$

where $q^{(d)}(t)$ and $q^{(b)}(t)$ define the ratios of overlap between the default or bypass tubes with the main tube so that

$$q^{(d)} + q^{(b)} \leq 1 \tag{5}$$

A wave propagates through the main tube until it meets the junction between the three tubes. At this point the wave is then split between the default and bypass tubes. At the junction, the pressure is assumed constant at the point of contact of the three tubes and the volume velocity at the junction sums to zero [13, 8].

$$p^{(m)} = p^{(d)} = p^{(b)} \tag{6a}$$

$$S^{(m)} v^{(m)} = S^{(d)} v^{(d)} + S^{(b)} v^{(b)} \tag{6b}$$

Changing the overlap in (5) changes the partitioning of the airflow in (6b) and therefore modifies the resonance's of the instrument.

2.3. Radiation

Radiation of sound from a brass instrument, to the simplest approximation, can be considered in the same way as that from an unflanged cylinder, for which Levine and Schwinger proposed a suitable model [14]. To apply this in the time domain, rational approximations must again be made which result in a radiation impedance in terms of a normalised frequency $\omega' = a\omega/c$ [15]

$$Z_R = \rho c \left(\frac{(1+\Gamma)\Lambda j\omega' + \Gamma\Lambda (j\omega')^2}{1 + \Gamma + (\Lambda + \Gamma\Theta) j\omega' + \Gamma\Lambda\Theta (j\omega')^2} \right) \tag{7}$$

where

$$\Gamma = 0.505, \quad \Lambda = 0.613, \quad \Theta = 1.111 \tag{8}$$

and a is the radius of the tube at its radiating end. The system described by equation (7) has an equivalent circuit form that is described in [7]. For brevity the network equations are presented in combined form

$$\bar{v} = v_R + \left(\frac{1}{\Gamma\rho c} + \frac{\Theta a}{\rho c^2}\frac{d}{dt}\right) p_R \tag{9a}$$

$$\bar{p} = \Lambda\rho a \frac{dv_R}{dt} \tag{9b}$$

$$\bar{p} = \left(1 + \frac{1}{\Gamma} + \frac{\Theta a}{c}\frac{d}{dt}\right) p_R \tag{9c}$$

where $\bar{v} = v(L,t)$ and $\bar{p} = p(L,t)$ are values taken at the end of the instrument, v_R and p_R denote network variables (equivalent current and voltage) and d/dt denotes differentiation respect to time.

2.4. Lip Reed

The subject of lip reed modelling has seen a large amount of investigation, and models of varying degrees of complexity are available. For this synthesis environment a simple one degree of freedom, outward striking reed model been chosen as the excitation mechanism of the instrument [16]. The reed dynamics are described by

$$\frac{d^2y}{dt^2} + \sigma\frac{dy}{dt} + \omega_0^2 y = \frac{S_r \Delta p}{\mu_r} \tag{10a}$$

where $y(t)$ is the reed displacement from its equilibrium position H, and ω_0 and σ are the reed's natural resonance angular frequency and damping parameter. S_r and μ_r are the effective surface area and mass of the reed. $\Delta p(t)$ is the pressure difference between the mouth pressure, $p_m(t)$, and that within the instrument embouchure so that

$$\Delta p = p_m - p\,(0,t) \tag{10b}$$

The dynamics of the lip reed are coupled to the instrument through a Bernoulli term, generated by the pressure difference between the mouth and instrument, and by a volume velocity produced by the motion of the reed

$$u_m = w\,[y+H]_+ \sqrt{\frac{2|\Delta p|}{\rho}}\mathrm{sign}\,(\Delta \mathrm{p}) \tag{10c}$$

$$u_r = S_r \frac{dy}{dt} \tag{10d}$$

$$S\,(0)\,v\,(0,t) = u_m + u_r \tag{10e}$$

where w is the effective lip width and $u_m(t)$ and $u_r(t)$ denote volume velocities generated by the pressure difference and the lip motion. The function $[\cdot]_+$ is defined as $[\cdot]_+ = \max(\cdot, 0)$, meaning that when the lips are closed there is no flow.

3. FDTD SCHEME

3.1. Difference Operators

Before presenting the update schemes for the system equations it is useful to define the discrete operators used in this work. Consider a grid function, ζ_l^n, defined for integer $l = 0, \ldots, N$ and $n = 0, 1 \ldots,$. Such a grid function represents an approximation to an underlying function $\zeta(x,t)$, as $\zeta_l^n \approxeq \zeta\,(lh, nk)$, where here, h is the grid spacing and k is the time step (in audio applications, the inverse of the sampling frequency).
Let $e_{x\pm}$ and $e_{t\pm}$ be spatial and temporal shift operators

$$e_{x\pm}\zeta_l^n = \zeta_{l\pm1}^n, \quad e_{t\pm}\zeta_l^n = \zeta_l^{n\pm1} \tag{11}$$

Combinations of these basic shift operators can then be used to arrive at various approximations to partial derivatives as well as averaging operators, which approximate a multiplication by unity, that can centre schemes; see Table 1 for a full list of discrete operators used in this work and the operators they approximate.

Table 1: *List of discrete operators and the continuous operators they approximate. Note that the constants a_r and b_r are defined in Section 3.2.*

Spatial Operators	Expression	Approximates
Forwards Difference, δ_{x+}	$(e_{x+} - 1)/h$	∂_x
Backwards Difference, δ_{x-}	$(1 - e_{x-})/h$	
Backwards Average, μ_{x-}	$(1 + e_{x-})/2$	1
Temporal Operators	**Expression**	**Approximates**
Forwards Difference, δ_{t+}	$(e_{t+} - 1)/k$	$\partial_t, \frac{d}{dt}$
Backwards Difference, δ_{t-}	$(1 - e_{t-})/k$	
Centered Difference, $\delta_{t\cdot}$	$(e_{t+} - e_{t-})/(2k)$	
Forwards Average, μ_{t+}	$(e_{t+} + 1)/2$	
Backwards Average, μ_{t-}	$(1 + e_{t-})/2$	1
Centered Average, $\mu_{t\cdot}$	$(e_{t+} + e_{t-})/2$	
Fractional Derivative, $\delta_{t^{1/2}}$	$\sqrt{\frac{2}{k}}\frac{\sum_{r=0}^{M} b_r e_{t-}^r}{\sum_{r=0}^{M} a_r e_{t-}^r}$	$\partial_{t^{1/2}}$

3.2. Approximation of Fractional Derivatives

Implementation of fractional derivatives can be performed through the use of an IIR filter constructed using a truncated Continued Fraction Expansion (CFE) of the bilinear transform which is used to approximate the square root of $j\omega$ [17]

$$(j\omega)^{1/2} \approx \sqrt{\frac{2}{k}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)} \tag{12}$$

where $z = e^{j\omega k}$. The expansion of the numerator and denominator of this expression can then be used along with Viscovatov's algorithm [18] for computing the CFE. In essence, this algorithm requires adding 0 to each new level of the CFE which allows for factorisation of z^{-1} and inversion of the rest of the fraction. Neglecting the factor $\sqrt{2/k}$, the process for converting the expansion, whose numerator and denominator coefficients are $\beta_r^{(0)}$ and $\alpha_r^{(0)}$, of the bilinear transform to a CFE is as follows

$$\frac{\left(1-z^{-1}\right)^{1/2}}{\left(1+z^{-1}\right)^{1/2}} \approx \underbrace{\frac{\sum_{r=0}^{\infty}\beta_r^{(0)} z^{-r}}{\sum_{r=0}^{\infty}\alpha_r^{(0)} z^{-r}}}_{\text{Expansion}} + \underbrace{\frac{\beta_0^{(0)}}{\alpha_0^{(0)}} - \frac{\beta_0^{(0)}}{\alpha_0^{(0)}}}_{+0} \tag{13}$$

$$= \frac{\beta_0^{(0)}}{\alpha_0^{(0)}} + \frac{\sum_{r=1}^{\infty}\left(\beta_r - \frac{\beta_0^{(0)}}{\alpha_0^{(0)}}\alpha_r^{(0)}\right) z^{-r}}{\sum_{r=0}^{\infty}\alpha_r^{(0)} z^{-r}}$$

$$= \xi^{(0)} + \frac{j\omega}{\left[\frac{\sum_{r=0}^{\infty}\alpha_r^{(0)} z^{-r}}{\sum_{r=1}^{\infty}\left(\beta_r^{(0)} - \xi^{(0)}\alpha_r^{(0)}\right) z^{-(r-1)}}\right]}$$

where $\xi^{(0)} = \beta_0^{(0)}/\alpha_0^{(0)}$. For each iteration, i, of this process, the expansion coefficients within the square brackets can be rewritten as in the form of equation (13) by using $\beta_r^{(i)} = \alpha_r^{(i-1)}$, $\alpha_r^{(i)} = \beta_{r+1}^{(i-1)} - \xi^{(i-1)}\alpha_{r+1}^{(i-1)}$ and $\xi^{(i)} = \beta_0^{(i)}/\alpha_0^{(i)}$. Applying this process $2M$ times leads to a truncated CFE

$$(j\omega)^{1/2} \approx \xi^{(0)} + \cfrac{z^{-1}}{\xi^{(1)} + \cfrac{z^{-1}}{\ddots + \cfrac{z^{-1}}{\xi^{(2M)}}}}$$

The two lowest levels can be rewritten as

$$\frac{\xi^{(2M-1)}\xi^{(2M)} + z^{-1}}{\xi^{(2M)}} = \frac{b_0^{(1)} + b_1^{(1)} z^{-1}}{a_0^{(1)}}$$

which can be considered as a new series expansion with numerator and denominator coefficients $b_r^{(i)}$ and $a_r^{(i)}$. This can then be iteratively inverted to create a new polynomial fraction where $b_0^{(i)} = \xi^{(2M-i)} b_0^{(i-1)}$, $b_r^{(i)} = a_{r-1}^{(i-1)} + \xi^{(2M-i)} b_r^{(i-1)}$ for $r > 0$ and $a_r^{(i)} = b_r^{(i-1)}$ for $r \geq 0$.
Transforming into the time domain using $z^{-1} \rightarrow e_{t-}$ and reintroducing the factor $\sqrt{2/k}$ leads to a discrete approximation to the fractional derivative.

$$\delta_{t^{1/2}} = \sqrt{\frac{2}{k}} \frac{\sum_{r=0}^{M} b_r e_{t-}^r}{\sum_{r=0}^{M} a_r e_{t-}^r} \tag{14}$$

where b_r and a_r are the final iteration inversion of the CFE and have been normalised so that $a_0 = 1$. For sound synthesis at a sample rate of 44.1kHz, $M = 20$ is a suitable order filter for typical brass instrument geometries. This significantly increases the storage requirements of the algorithm as a whole, thus increasing the operation count and creating a bottleneck in performance.

3.3. Interleaved Grid and Difference Equations

For FDTD simulation of wave propagation in an acoustic tube it is useful to employ an interleaved time-space grid for the pressure and velocity fields, similar to work presented in electro-magnetics [19]. The pressure field is sampled at integer multiple time-space locations and the velocity field on the half integer points so that $p_l^n \approx p\left(lh, nk\right)$ and $v_{l+1/2}^{n+1/2} \approx v\left(\left(l+1/2\right)h, \left(n+1/2\right)k\right)$; see Fig. 4 at top. This leads to a representation in terms of $N+1$ pressure points and N velocity points, where $N = \text{floor}(L/h)$. The bore profile of the instrument must also be defined on both spatial grids. In this work the bore profile is sampled using linear interpolation on the velocity grid so that $S_{l+1/2} = S\left(\left(l+1/2\right)h\right)$. For the pressure grid the bore profile is defined as the spatial average of the neighbouring points on the velocity grid, $\bar{S}_l = 0.5\left(S_{l+1/2} + S_{l-1/2}\right)$; see Fig. 4 at bottom. Under these conditions, it can be shown that the Courant condition [20] must be satisfied for stability of simulations; that is $\lambda = ck/h$ where $\lambda \leq 1$ [7]. The different discretisations of the bore cross section mean that the functions in (3) must also be sampled: q is sampled on the pressure grid using $\bar{S}_l$, f and g are sampled on the velocity grid using $S_{l+1/2}$. See Table 2. For the valve sections, $S^{(d)}(x)$ is set from the location of the default tube in the main bore, although scaled by $q^{(d)}$. The default tube is therefore the path straight through the valve when the valve is not pressed. The basic profile of $S^{(b)}(x)$ is a cone whose entrance and exit areas are equal to the area of the main bore where the valve section begins and ends and has a length different to the default path so that the instrument's resonances are modified when waves propagate through this tube. The entrance and exit of the bypass tube are then scaled by $q^{(b)}$ to represent the constriction where the tube leaves and reenters the valve; the length of these constricted sections being equal to the length of the default tube. As the lengths of the tubes are distinct, the grid spacing must be chosen separately for the main, default and bypass sections; these are written as h_m, h_d and h_b respectively.

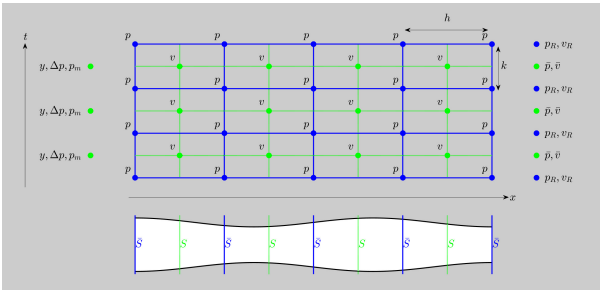


Figure 4: Top: Interleaved grid for pressure (blue lines and nodes), and velocity (green lines and nodes) fields with spatial and temporal step sizes of h and k. The network variables, p_R and v_R, are computed simultaneously with pressure values whereas $\bar{p}$ and $\bar{v}$ and the lip variables (y, Δp and p_m) are computed with the velocity variables. Bottom: Discretisation of bore profile with S at the velocity field locations and $\bar{S}$ at the pressure field locations.

The network variables in the radiation model are aligned with integer time steps whereas the main radiation variables and the lip model values are aligned with half integer time steps.

4. UPDATE EQUATIONS

Individual update equations can be derived by applying the difference operators in Table 1 to the system equations in Section 2. The operators can then be expanded and the equations rearranged to yield an update for the next time-step.

4.1. Wave Propagation Update

For the wave propagation in the bore, equations (2b) and (2a) become

$$\left(\rho\delta_{t-} + f_{l+1/2}\mu_{t-} + g_{l+1/2}\delta_{t^{1/2}}\mu_{t-}\right) v_{l+1/2}^{n+1/2} + \delta_{x+} p_l^n = 0 \tag{15a}$$

$$\left(\frac{\bar{S}_l}{\rho c^2}\delta_{t+} + q_l\delta_{t^{1/2}}\mu_{t+}\right) p_l^n + \delta_{x-}\left(S_{l+1/2} v_{l+1/2}^{n+1/2}\right) = 0 \tag{15b}$$

Due to the occurrences of the discrete fraction derivative operator $\delta_{t^{1/2}}$, it is then necessary to multiply through by $\sum_{r=0}^{M} a_r e_{t-}^r$ and then rearrange to get the update equations

$$v_{l+1/2}^{n+1/2} = \sum_{r=0}^{M} Q_{vv}^{(r)} e_{t-}^r v_{l+1/2}^{n-1/2} - Q_{vp}^{(r)} e_{t-}^r p_l^n \tag{16a}$$

$$p_l^{n+1} = \sum_{r=0}^{M} Q_{pp}^{(r)} e_{t-}^r p_l^n - Q_{pv}^{(r)} e_{t-}^r v_{l+1/2}^{n+1/2} \tag{16b}$$

Where the multiplying coefficients are defined in Table 2.

4.2. Valve Junctions Update

At the valve junctions, that is where the main tube meets the default and bypass tubes, equations (6a) and (6b) become

$$p_{N_J}^{(m)} = p_0^{(d)} = p_0^{(b)} = p_J \tag{17a}$$

$$\mu_{x-}\left(S_{N_J+1/2}^{(m)} v_{N_J+1/2}^{(m)}\right) = \mu_{x-}\left(S_{1/2}^{(d)} v_{1/2}^{(d)}\right) + \mu_{x-}\left(S_{1/2}^{(b)} v_{1/2}^{(b)}\right) \tag{17b}$$

where N_J denotes the index in the main tube where the valve lies. The following identity can be used to couple each tube together using the volume velocities

$$\delta_{x-} = \frac{2}{h}\left(1 - \mu_{x-}\right) = \frac{2}{h}\left(\mu_{x-} - e_{x-}\right) \tag{18}$$

Using this and that the pressure in all three tubes is the same at the valve junction, equation (15b) can be written as

$$p_J^{n+1} = \sum_{r=0}^{M} Q_{Jp}^{(r)} e_{t-}^r p_J^n + Q_{Jv}^{(r)} e_{t-}^r \left(S_{N_J+1/2}^{(m)} v_{N_J+1/2}^{(m)} - S_{1/2}^{(d)} v_{1/2}^{(d)} - S_{1/2}^{(b)} v_{1/2}^{(b)}\right) \tag{19}$$

at the valve boundary. When the default and bypass tubes combine back into the main tube, the sign of $Q_{Jv}^{(r)}$ is inverted. See Table 3 for coefficient definitions. Fig. 5 shows simulated wave propagation within a trumpet with partially open valves. When the wave encounters the valve junction it splits as it travels through the default and bypass tubes.

4.3. Radiation Update

The network variables in equations (9a), (9b) and (9c) become

$$\bar{v}^{n+1/2} = \mu_{t+} v_R^n + \left(\frac{\mu_{t+}}{\Gamma \rho c} + \frac{\Theta a}{\rho c^2}\delta_{t+}\right) p_R^n \tag{20a}$$

$$\bar{p}^{n+1/2} = \Lambda \rho a \delta_{t+} v_R^n \tag{20b}$$

$$\bar{p}^{n+1/2} = \left(\left(1 + \frac{1}{\Gamma}\right)\mu_{t+} + \frac{\Theta a}{c}\delta_{t+}\right) p_R^n \tag{20c}$$

where $\bar{S}_N \bar{v}^{n+1/2} = \mu_{x-}\left(S_{N+1/2} v_{N+1/2}^{n+1/2}\right)$ and $\bar{p}^{n+1/2} = \mu_{t+} p_N^n$. The updates for v_R and p_R can be written in terms of unknown pressure values at the end of the instrument

$$v_R^{n+1} = v_R^n + \frac{k}{2\Lambda\rho a}\left(p_N^{n+1} + p_N^n\right) \tag{21a}$$

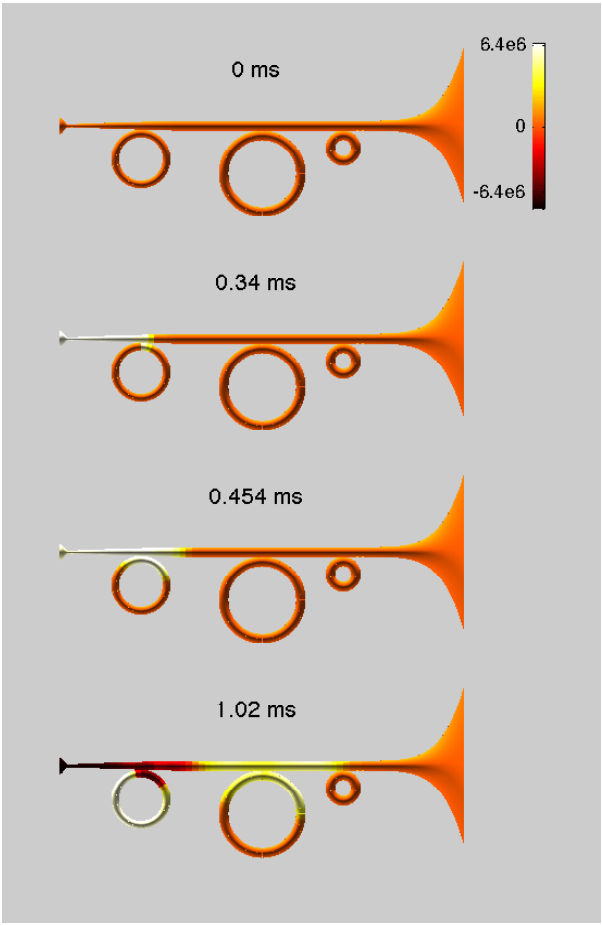


Figure 5: Pressure wave propagation within a valved brass instrument at several time steps simulated at 44.1 kHz excited with a raised cosine pulse and with all valves in a half-open configuration, at times as indicated, and illustrating splitting of traveling waves at valve junctions.

$$p_R^{n+1} = \left[\frac{1}{2}\left(1 + \frac{1}{\Gamma}\right) + \frac{\Theta a}{ck}\right]^{-1}\left(\frac{p_N^{n+1} + p_N^n}{2} + \left[\frac{\Theta a}{ck} - \frac{1}{2}\left(1 + \frac{1}{\Gamma}\right)\right] p_R^n\right) \tag{21b}$$

These relations along with the second identity in equation (18) can be used to update equation (15b) at the radiating end of the instrument. It is sufficient to use the lossless version of this equation (neglecting the fractional derivative term) to get

$$p_N^{n+1} = Q_{rp} p_N^n + Q_{rv} v_{N-1/2}^{n+1/2} + Q_{rv_R} v_1^n + Q_{rp_R} p_R^n \tag{22}$$

See Table 4 for radiation update coefficients.

4.4. Lip Reed Update

Equations (10a) to (10e) can be written as

$$\left(\delta_{t+}\delta_{t-} + \sigma\delta_{t\cdot} + \omega_0^2\mu_{t\cdot}\right) y^{n+1/2} = \frac{S_r \Delta p^{n+1/2}}{\mu_r} \tag{23a}$$

$$\Delta p^{n+1/2} = p_m^{n+1/2} - \mu_{t+} p_0^n \tag{23b}$$

$$u_m^{n+1/2} = w\left[y^{n+1/2} + H\right]_+ \sqrt{\frac{2|\Delta p^{n+1/2}|}{\rho}} \operatorname{sign}\left(\Delta p^{n+1/2}\right) \tag{23c}$$

$$u_r^{n+1/2} = S_r \delta_{t\cdot} y^{n+1/2} \tag{23d}$$

$$\mu_{x-}\left(S_{1/2} v_{1/2}^{n+1/2}\right) = u_m^{n+1/2} + u_r^{n+1/2} \tag{23e}$$

Equation (23a) becomes

$$y^{n+3/2} = \left(Q_{y1} y^{n+1/2} + Q_{y2} y^{n-1/2} + Q_{yp} \Delta p^{n+1/2}\right) \tag{24}$$

The lip model can be coupled to the instrument in a similar manner to a clarinet reed [21]. Equation (23a) can be rewritten in terms of $\delta_{t\cdot} y^{n+1/2}$ and $\Delta p^{n+1/2}$ using

$$\delta_{t+}\delta_{t-} = \frac{2}{k}\left(\delta_{t\cdot} - \delta_{t-}\right), \quad \mu_{t\cdot} = k\delta_{t\cdot} + e_{t-} \tag{25}$$

Then using equations (23c), (23d) and (23e) leads to an expression in terms of $\Delta p^{n+1/2}$ and $\mu_{x-}\left(S_{1/2} v_{1/2}^{n+1/2}\right)$.

At $l = 0$ equation (16b) can be rewritten in terms of $\Delta p^{n+1/2}$ and $\mu_{x-}\left(S_{1/2} v_{1/2}^{n+1/2}\right)$ by using the first identity in equation (18) along with

$$\delta_{t+} = \frac{2}{k}\left(\mu_{t+} - 1\right) \tag{26}$$

and equation (23b) when the fractional derivatives are neglected. This then combines to create the quadratic expression

$$|\Delta p^{n+1/2}| + \frac{d_3^n}{d_2 + c_1}\sqrt{|\Delta p^{n+1/2}|} - \left|\frac{d_1^n - c_2^n}{d_2 + c_1}\right| = 0 \tag{27}$$

provided

$$\operatorname{sign}\left(\Delta p^{n+1/2}\right) = -\operatorname{sign}\left(\frac{d_1^n - c_2^n}{d_2 + c_1}\right)$$

The value of the pressure difference can then be used to update the lip position and pressure at the input of the acoustic tube.

$$p_0^{n+1} = 2\left(p_m^{n+1/2} - \Delta p^{n+1/2}\right) - p_0^n \tag{28}$$

See Table 5 for lip update coefficients.

5. ENVIRONMENT DEVELOPMENT AND OPTIMISATION

The brass environment was originally developed in the MATLAB prototyping platform [22] and then ported to the C programming language. Input to the environment consists of instrument and score files; output is in .wav format, and is drawn from the pressure at the radiating end of the instrument. Bore profile definition is in the hands of the user, either through manual specification of coordinate/radius pairs, or using a simplified set of concatenated sections; valve positions and lengths of the default and bypass tubes may also be supplied. The score file consists of breakpoint functions which specify the time variation of mouth pressure, lip parameters and valve openings.

For a single-valved trumpet of length 1.381m, with default and bypass tube lengths of 0.02m and 0.2m, the MATLAB code takes approximately 14s to generate 1 second of output on an Intel Core i5-4300U running at 2.5GHz. See Table 6 for a list of optimisation methods and respective accelerations. A direct translation from MATLAB to C, using the same algorithms and data structures, results in a 5x speed up—typical of a MATLAB-to-C conversion. To further improve performance, two methods of parallel execution

Table 2: *Coefficients in wave propagation update.*

$$Q_{vv}^{(r)} = \frac{2\rho(a_r - a_{r+1}) - k\left(f_{l+1/2}(a_r + a_{r+1}) + g_{l+1/2}(b_r + b_{r+1})\right)}{\left(2\rho + k f_{l+1/2}\right) a_0 + k g_{l+1/2} b_0}$$

$$Q_{vp}^{(r)} = \frac{2k a_r \delta_{x+}}{\left(2\rho + k f_{l+1/2}\right) a_0 + k g_{l+1/2} b_0}$$

$$Q_{pp}^{(r)} = \frac{2\bar{S}_l\left(a_r - a_{r+1}\right) - \rho c^2 k q_l \left(b_r + b_{r+1}\right)}{2\bar{S}_l a_0 + \rho c^2 k q_l b_0}$$

$$Q_{pv}^{(r)} = \frac{2\rho c^2 k a_r \delta_{x-}\left(S_{l+1/2}\,\cdot\right)}{2\bar{S}_l a_0 + \rho c^2 k q_l b_0}$$

$$f_{l+1/2} = \frac{3\eta\pi}{S_{l+1/2}},\ g_{l+1/2} = 2\sqrt{\frac{\rho\eta\pi}{S_{l+1/2}}},\ q_l = \frac{2\left(\gamma - 1\right)}{\nu c^2}\sqrt{\frac{\eta\pi\bar{S}_l}{\rho^3}}$$

Table 3: *Coefficients in valve junction update.*

$$\begin{aligned} Q_{Jp}^{(r)} = \ & Q_{J0}\left[\left(h_m + h_d q^{(d)} + h_b q^{(b)}\right)\frac{\bar{S}_{N_J}}{2\rho c^2 k}\left(a_r - a_{r+1}\right)\right. \\ & \left. -\left(h_m + h_d\sqrt{q^{(d)}} + h_b\sqrt{q^{(b)}}\right)\frac{\gamma - 1}{2\nu c^2}\sqrt{\frac{\eta\pi\bar{S}_{N_J}}{\rho^3}}\left(b_r + b_{r+1}\right)\right] \\ Q_{Jv}^{(r)} = \ & a_m Q_{J0} \\ Q_{J0} = \ & \left[\left(h_m + h_d q^{(d)} + h_b q^{(b)}\right)\frac{\bar{S}_{N_J}}{2\rho c^2 k} a_0\right. \\ & \left. + \left(h_m + h_d\sqrt{q^{(d)}} + h_b\sqrt{q^{(b)}}\right)\frac{\gamma - 1}{2\nu c^2}\sqrt{\frac{\eta\pi\bar{S}_{N_J}}{\rho^3}} b_0\right]^{-1} \end{aligned}$$

Table 4: *Coefficients in radiation update.*

$$Q_{rp} = \frac{1 - Q_{r0}}{1 + Q_{r0}} \quad Q_{rv} = \frac{2\rho c^2 k S_{N-1/2}}{\bar{S}_N h\left(1 + Q_{r0}\right)}$$

$$Q_{rv_R} = -\frac{2\rho c^2 k}{h\left(1 + Q_{r0}\right)}$$

$$Q_{rp_R} = Q_{rv_R}\left(\frac{1}{2\Gamma\rho c} - \frac{\Theta a}{\rho c^2 k} + \left(\frac{1}{2\Gamma\rho c} + \frac{\Theta a}{\rho c^2 k}\right)\times \left[\frac{1}{2}\left(1 + \frac{1}{\Gamma}\right) + \frac{\Theta a}{ck}\right]^{-1}\left(\frac{\Theta a}{ck} - \frac{1}{2}\left(1 + \frac{1}{\Gamma}\right)\right)\right)$$

$$Q_{r0} = \frac{\rho c^2 k}{h}\left(\frac{k}{2\Lambda\rho a} + \left(\frac{1}{2\Gamma\rho c} + \frac{\Theta a}{\rho c^2 k}\right)\left[\frac{1}{2}\left(1 + \frac{1}{\Gamma}\right) + \frac{\Theta a}{ck}\right]^{-1}\right)$$

Table 5: *Coefficients in lip update.*

$$Q_{y1} = Q_{y0}\frac{2}{k^2}, \quad Q_{y2} = Q_{y0}\left(\frac{\sigma}{2k} - \frac{1}{k^2} - \frac{\omega_0^2}{2}\right)$$
$$Q_{yp} = Q_{y0}\frac{S_r}{\mu_r}, \quad Q_{y0} = \left(\frac{1}{k^2} + \frac{\sigma}{2k} + \frac{\omega_0^2}{2}\right)^{-1}$$
$$d_1^n = S_r\frac{\left(2\delta_{t-} - k\omega_0^2 e_{t-}\right)y^{n+1/2}}{2+\sigma k+\omega_0^2 k^2}, \quad d_2 = \frac{kS_r^2}{\mu_r\left(2+\sigma k+\omega_0^2 k^2\right)}$$
$$d_3^n = w|y^{n+1/2} + H|_+\sqrt{\frac{2}{\rho}}$$
$$c_1 = \frac{\bar{S}_0 h}{\rho c^2 k}, \quad c_2^n = S_{1/2}v_{1/2}^{n+1/2} + c_1\left(p_m - p_0^n\right)$$

have been considered along with modification to the set up of the control stream and are presented below.

5.1. GPU Acceleration

Use of NVIDIA GPUs for simulation of other instrument systems programmed using the CUDA platform [23] has shown significant performance increases by solving different parts of the problem on individual cores [24, 25].
These methods are extremely effective for large scale problems, such as e.g., systems defined in 2D and 3D, but in the case of the small 1D brass instrument system the overheads required to transfer data between GPUs results in performance that is 4.7x slower than serial C when run on Tesla K20c GPUs.

5.2. Vectorisation

Modern CPUs contain powerful vector units capable of performing multiple floating point or integer operations with a single instruction (a programming model known as Single Instruction Multiple Data). This gives parallelism at a much finer-grained level than the method described above, and because the program is still running as a single thread on one CPU core, the synchronisation and data transfer bottlenecks of GPU methods can be avoided. Although compilers can sometimes automatically vectorise code, it is often still necessary to vectorise manually to get the best results, especially for more complex operations. Manual vectorisation involves using compiler intrinsics (special functions that map directly to machine instructions) or assembly language to program the vector unit directly.
The brass code was vectorised using the AVX instructions available on modern Intel and AMD CPUs [26]. The AVX unit provides 256-bit wide vector registers capable of storing 4 double precision floating point values, and parallel execution units capable of operating on all 4 values at once. The inner loops of the pressure and velocity updates for the main and bypass tubes were vectorised, as these are by far the most time consuming elements of the code. The default tube updates were not vectorised as the default tube is generally very small and takes very little time to process even in serial. This more than doubles performance relative to serial C and is 11.5x faster than MATLAB.

5.3. Interpolation of Control Stream

A final optimisation for both the serial and AVX versions of the code involves interpolating control stream values for the lip and valve inputs on-the-fly at each time step instead of pre-computing them all at startup and storing them in arrays. The intention here is primarily to save memory, but in fact it also has the effect of significantly speeding up both versions of the code: the serial C performance for the trumpet became 7.1x faster than the original MATLAB, and the AVX performance increased to 16.3x faster than MATLAB. These improvements are due to a reduction in the amount of data read from the main memory at each time step, relieving traffic on the memory bus which is often a bottleneck on modern systems.

Table 6: *Run times and speed increases for different optimisation methods for 1s of output sound using a trumpet bore of length* $1.381\mathrm{m}$*, with a single valve with default tube length* $0.02\mathrm{m}$ *and bypass tube length* $0.2\mathrm{m}$*. Test instrument and score files are available online along with final environment binaries. Times were taken from a machine with Intel Core i5-4300U except for those on GPU which were run on Tesla K20c GPUs.*

Code version	Run time (s)	Speed-up over MATLAB
MATLAB	14.02	1x
Serial C	2.69	5.2x
GPU	12.75	1.1
AVX	1.22	11.5x
Serial C (memory optimised)	1.98	7.1
AVX (memory optimised)	0.86	16.3x

6. CONCLUSIONS AND FUTURE WORK

The environment binaries are available online at `http://www2.ph.ed.ac.uk/~s0916351/dafx15.html`, along with user documentation. This work is still in the early stages of development and there are multiple extensions planned to improve performance and to add additional features. The dominant time-intensive feature of the algorithm lies with the approximation of the fractional derivatives, which themselves are features of the approximations to the impedance and admittance formulae. It would therefore be of interest to lower the order of this filter. Thompson *et al* [27] present an analog filter structure to model the viscous and thermal losses present in an acoustic tube that does not require the use of fractional derivatives. Although their method still requires a relatively high order of time steps to be stored, optimisation techniques, such as those used by Abel *et al* [28], can be used to create lower order filter designs; see [29] for preliminary work on this problem.
Due to the (usually) short length of the default tubes, when discretising these sections the spacing, h, is increased which leads to a reduction in the bandwidth over these sections; for normal instrument geometries this is up to about $16\mathrm{kHz}$. This could be improved through interpolation of the position of the valve junctions.
Currently, the output from the environment is a mono sound file that is taken from the very end of the instrument. To add spatialisation to the produced sound it would be possible to embed the instrument within a 3D space by replacing the current radiation model with an energy coupling between the instrument and room, similar to work that has been done with percussion instruments [24, 25]. In this case the room simulation can be performed using GPUs, with potentially very large acceleration.
The basic design described here could be used in order to simulate a clarinet by a small alteration to the excitation mechanism (to

simulate a reed rather than a lip), and in replacing valve sections by toneholes. The excitation model could also be extended to include collisions between the lips or the reed and lay. See [30] for some preliminary results.

A more fundamental alteration to the environment would be to model nonlinear wave propagation within the instrument bore. These processes are responsible for the characteristic "brassy" or "cuivré" sound present when instruments such as the trumpet and trombone are played at high dynamic levels [31, 32]. These systems, however, require entirely new numerical design techniques in order to cope with severe stability issues as shocks develop at the wave front [33].

7. ACKNOWLEDGMENTS

This work was supported by the European Research Council, under grant number StG-2011-279068-NESS.

8. REFERENCES

[1] F. Silva, C. Vergez, P. Guillemain, J. Kergomard, and V. Debut, "Moreesc: A framework for the simulation and analysis of sound production in reed and brass instruments," *Acta Acust united Ac*, vol. 100, pp. 126–138, 2014.

[2] P. Cook, "Tbone: An interactive waveguide brass instrument synthesis workbench for the next machine," in *Proc. of the Int. Computer Music Conference*, Montreal, Canada, 1991, pp. 297–299.

[3] T. Hélie, R. Mignot, and D. Matignon, "Waveguide modeling of lossy flared acoustic pipes: Derivation of a kelly-lochbaum structure for real-time simulations," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, October 2007.

[4] R. Mignot, T. Hélie, and D. Matignon, "Digital waveguide modeling for wind insturments: Building a state-space representation based on webster-lokshin model," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 4, pp. 843–854, 2010.

[5] C. Vergez, *Trompette et Trompettiste: Un Système Dynamique Non Linéaire à Analyser, Modéliser et Simuler dans un Contexte Musical*, Ph.D. thesis, Universite Paris 6, 2000.

[6] C. Vergez and P. Tisserand, "The brass project, from physical models to virtual musical instruments: Playability issues," in *Lecture Notes in Computer Science: Computer Music Modeling and Retrieval*, R. Kronland-Martinet, T. Voinier, and S. Ystad, Eds., vol. 3902, pp. 24–33. Springer, Berlin/Heidelberg, Germany, September 2006.

[7] S. Bilbao and J. Chick, "Finite difference time domain simulation for the brass instrument bore," *J Acoust Soc Am*, vol. 134, no. 5, pp. 3860–3871, 2013.

[8] R. L. Harrison and J. Chick, "A single valve brass instrument model using finite-difference time-domain methods," in *Proc. of the Int. Symp. on Musical Acoustics.*, Le Mans, France, 2014.

[9] C. Zwikker and C.W. Kosten, *Sound absorbing materials*, Elsevier Pub. Co., 1949.

[10] D. H. Keefe, "Acoustical wave propagation in cylindrical ducts: Transmission line parameter approximations for isothermal and nonisothermal boundary conditions," *J Acoust Soc Am*, vol. 75, no. 1, pp. 58–62, 1984.

[11] T. Hélie, "Unidimensional models of acoustic propagation in axisymmetric waveguides," *J Acoust Soc Am*, vol. 114, no. 5, pp. 2633–2647, 2003.

[12] A. H. Benade, "On the propagation of sound waves in a cylindrical conduit," *J Acoust Soc Am*, vol. 44, no. 2, 1968.

[13] S. Bilbao, "Modelling of brass instrument valves," in *Proc. of the 14^{th} Int. Conference on Digital Audio Effects*, Paris, France, September 2011, pp. 337–343.

[14] H. Levine and J. Schwinger, "On the radiation of sound from an unflanged circular pipe," *Phys Rev*, vol. 73, no. 4, pp. 383–406, 1948.

[15] F. Silva, Ph. Guillemain, J. Kergomard, B. Mallaroni, and A. N. Norris, "Approximation formulae for the acoustic radiation impedance of a cylindrical pipe," *J Sound Vib*, vol. 322, pp. 255–263, 2009.

[16] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments, Second Edition*, Springer, 1998.

[17] Y. Chen, B. M Vinagre, and I. Podlubny, "Continued fraction expansion approaches to discretizing fractional order derivatives - an expository review," *Nonlinear Dynamics*, vol. 38, pp. 115–170, 2004.

[18] A. Cuyt, *Handbook of continued fractions for special functions. [electronic resource].*, [Dordrecht, Netherlands] : Springer, [2008], ©2008., 2008.

[19] K. S. Yee, "Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media," *IEEE Trans. Antennas Propag.*, vol. 14, no. 3, pp. 302–307, 1966.

[20] R. Courant, K. Friedrichs, and H. Lewy, "On the partial differential equations of mathematical physics," *Mathematische Annalen*, vol. 100, pp. 32–74, 1928.

[21] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, Wiley, 2009.

[22] MathWorks, "Matlab - the language of technical computing," October 2014, Last accessed October 2014.

[23] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," *Queue*, vol. 6, no. 2, pp. 40–53, 2008.

[24] S. Bilbao and C. Webb, "Timpani drum synthesis in 3d on gpgpus," in *Proc. of the 15^{th} Int. Conference on Digital Audio Effects*, York, UK, September 2012.

[25] A. Torin and S. Bilbao, "A 3-d multi-plate environment for sound synthesis," in *Proc. of the 16^{th} Int. Conference on Digital Audio Effects*, Maynooth, Ireland, September 2013.

[26] N Firasta, M. Buxton, P. Jinbo, Ka. Nasri, and S. Kuo, "Intel® avx : New frontiers in performance improvements and energy efficiency," *Intel White Paper*, 2008, Available online https://software.intel.com/en-us/articles/intel-avx-new-frontiers-in-performance-improvements-and-energy-efficiency.

[27] S. C. Thompson, T. B. Gabrielson, and D. M. Warren, "Analog model for thermoviscous propagation in a cylindrical tube," *J Acoust Soc Am*, vol. 135, no. 2, pp. 585–590, 2014.

[28] J. Abel, T. Smyth, and J. O. Smith III, "A simple, accurate wall loss filter for acoustic tubes," in *Proc. of the 6^{th} Int. Conference on Digital Audio Effects*, London, UK, September 2003.

[29] S. Bilbao, R. Harrison, J. Kergomard, B. Lombard, and C. Vergez, "On Passive Models of Viscothermal Wave Propagation in Acoustic Tubes," *J Acoust Soc Am*, vol. 138, no. 2, pp. 555–558, 2015.

[30] S. Bilbao, A. Torin, and V. Chatziioannou, "Numerical modeling of collisions in musical instruments," *Acta Acust united Ac*, vol. 101, pp. 292–299, 2015.

[31] A. Hirschberg, J. Gilbert, R. Msallam, and A. P. J. Wijnands, "Shock waves in trombones," *J Acoust Soc Am*, vol. 99, no. 3, pp. 1754–1758, 1996.

[32] J. Gilbert, L. Menguy, and M. Campbell, "A simulation tool for brassiness studies(l)," *J Acoust Soc Am*, vol. 123, no. 4, pp. 1854–1857, 2008.

[33] O. Richoux, B. Lombard, and J-F. Mercier, "Generation of acoustic solitary waves in a lattice of Helmholtz resonators," *Wave Motion*, vol. 56, pp. 85–99, 2015.

DOWNMMIX-COMPATIBLE CONVERSION FROM MONO TO STEREO IN TIME- AND FREQUENCY-DOMAIN

Marco Fink, Sebastian Kraft, Udo Zölzer

Department of Signal Processing and Communications,
Helmut-Schmidt University Hamburg
Hamburg, Germany
marco.fink@hsu-hh.de, sebastian.kraft@hsu-hh.de

ABSTRACT

Even in a time of surround and 3D sound, many tracks and recordings are still only available in mono or it is not feasible to record a source with multiple microphones for several reasons. In these cases, a pseudo stereo conversion of mono signals can be a useful preprocessing step and/or an enhancing audio effect. The conversion proposed in this paper is designed to deliver a neutral sounding stereo image by avoiding timbral coloration or reverberation. Additionally, the resulting stereo signal is downmix-compatible and allows to revert to the original mono signal by a simple summation of the left and right channels. Several configuration parameters are shown to control the stereo panorama. The algorithm can be implemented in time-domain or also in the frequency-domain with additional features, like center focusing.

1. INTRODUCTION

A noteworthy amount of recordings was and is still being done in mono for technical or pragmatic reasons. For example, basic broadcast program outside the studio environment is often recorded using a single microphone. Nevertheless, the later replay and mixing would benefit from a stereo recording in terms of spaciousness and pleasing sound. Therefore, the process to create stereo signals from mono recordings, also called pseudo-stereo, is a well-known and broadly utilized field of audio technology. While real stereo mixes are created by panning discrete sources to a specific position in the stereo panorama, pseudo-stereo does usually not involve knowledge about the sources. It rather randomly pans certain frequency components to the left and the right to achieve a decorrelation between both channels.

Early attempts to realize a mono-to-stereo (M2S) conversion used a delayed version of the input signal to provide a second channel [1]. The same author came up with the idea of applying complementary comb filters which were later extended to be phase-aligned in [2]. Alternatively, in [3, 4] different allpass network designs were proposed to obtain a strong decorrelation and to achieve a wide and also scalable stereo image. Another extension allowing more control is shown in [5]. For even stronger decorrelation, a frequency-domain filter design method is suggested in [6]. The above methods either impose a strong timbral coloration or they are not downmix compatible and reversible. Both are important features, though.

A completely different approach granting possibilities to design a specific auditory image is explained in [7]. However, it is not a pseudo stereo algorithm in the narrower sense as it requires complex user input to explicitly define pan positions for certain frequency bands and does not allow a fully automatic conversion. The same holds true for upmixing based on Directional Audio Coding (DirAC) [8]. However, the decorrelation mechanism in the DirAC synthesis are similar to the proposed approach to a certain extent.

The novel pseudo-stereo conversion algorithm is derived in Sec. 2 and its implementation in time- and frequency-domain are depicted in Sec. 3 and Sec. 4, respectively. Section 5 demonstrates the methods capabilities with a few experiments before concluding thoughts are provided in Sec. 6.

2. CONVERSION APPROACH

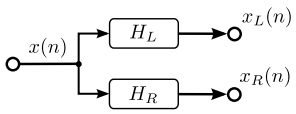


Figure 1: *Basic blockscheme of the pseudo-stereo system*

The basic idea of the pseudo-stereo system is to apply a filter $H_L(e^{j\Omega})$ to an input signal $x(n)$ to produce a first output channel $x_L(n)$ and a different filter $H_R(z)$ in parallel, producing a second output channel $x_R(n)$ as shown in Fig. 1. The static filters should have quite different characteristics to achieve a strong decorrelation between the two output channels and hence, introduce an impression of spatial width.

One major design criterion of the proposed system is downmix compatibility

$$\begin{aligned} x_M(n) &= x_L(n) + x_R(n) = \\ &= x(n) * h_L(n) + x(n) * h_R(n) \stackrel{!}{=} x(n-D), \end{aligned} \quad (1)$$

where the downmix signal $x_M(n)$ as the sum of the left and right stereo channel is identical to the unprocessed input except for a delay D. Transferring Eq. (1) to the frequency domain leads to

$$\begin{aligned} X_M(e^{j\Omega}) &= X(e^{j\Omega}) \cdot H_L(e^{j\Omega}) + X(e^{j\Omega}) \cdot H_R(e^{j\Omega}) = \\ &= X(e^{j\Omega}) \cdot \left(H_L(e^{j\Omega}) + H_R(e^{j\Omega}) \right) = \\ &\stackrel{!}{=} X(e^{j\Omega}) \cdot e^{-j\Omega D} \end{aligned} \quad (2)$$

and directly imposes the constraint

$$H_L(e^{j\Omega}) + H_R(e^{j\Omega}) \stackrel{!}{=} e^{-j\Omega D}, \quad (3)$$

where the sum of both transfer functions has constant magnitude and linear phase to guarantee downmix compatibility. Furthermore, for a neutral sounding stereo output without coloration artifacts the sum of the magnitude frequency responses

$$|H_L(e^{j\Omega})| + |H_R(e^{j\Omega})| \stackrel{!}{=} 1 \tag{4}$$

has to be constant. Additionally, both pseudo-stereo filters must feature conjugate symmetry

$$H_{L/R}(e^{j\Omega}) = H^*_{L/R}(e^{-j\Omega}) \tag{5}$$

to obtain real-valued impulse responses. Since these filters are linear phase with a constant group delay, they will not introduce a phase shift between the stereo output channels and only generate amplitude differences. Therefore, the resulting pseudo-stereo effect is only based on amplitude panning and not on time-delay panning. Hence, $|H_{L/R}(e^{j\Omega})|$ can be interpreted as panning coefficient in the range of $[0, \ldots, 1]$ corresponding to full panning from the complementary to the current channel at a certain frequency Ω. A value of 0.5 indicates center panning.

It was found that a regular pattern in the frequency domain, like provided by higher-order complementary comb filters, achieve great decorrelation but due to the uniform frequency response sound very unnatural. Hence, the frequency response of the system was designed to be diffuse and unstructured as shown in Fig. 2.

The actual implementation and the following filter design is achieved in the discrete fourier domain. To represent the frequency indexes of the sampled spectrum the variable k is used in the following. The proposed pseudo stereo filter design is based on a discrete real-valued noise sequence $R(k)$. The noise has a gaussian distribution with a standard deviation σ and a mean value of 0. The sequence $R(k)$ is scaled with w^2 and non-linearly clamped with an $\arctan$ function. Further normalization by π and an additive offset of $\frac{1}{2}$ yields the transfer function

$$H_L(k) = \left(\frac{1}{2} + \frac{1}{\pi}\arctan(w^2 \cdot R(k))\right) e^{-j\frac{2\pi k D}{N}} \tag{6}$$

of the left channel decorrelation filter. The amplitudes of that and its complementary filter $H_R(k) = 1 - H_L(k)$ are bound to a range $[0, \ldots, 1]$ and fulfill the requirements defined in (3) and (4). The parameter w allows dynamic control of the resulting stereo width. For $w = 0$ the filters have a constant magnitude response of 0.5 and hence, no panning is performed. With increasing w the values of the frequency responses are more and more clamped to the extreme values 0 and 1 (Fig. 2). In consequence, a higher degree of decorrelation is achieved.

The panning of very low and very high frequencies can be perceived annoying since the listener is used to hear certain instruments like bass guitar and bass drum in the center of the stereo panorama. Therefore, the amplitude panning in those frequency regions is disabled by setting

$$|H_{L/R}(k)| = 0.5, \quad \text{for } k < k_{\text{lo}} \vee k > k_{\text{hi}}, \tag{7}$$

where $k_{\text{lo/hi}}$ are the cut-off frequencies defining the passband to be actually processed.

An exemplary M2S conversion is shown in Fig. 4. A recording of a small singing ensemble with 4 voices is used as input signal $x(n)$. The spectrogram in Fig. 4a) shows the trend of the harmonic voices. The spectrograms for left and right output channel in Fig. 4b+c) indicate how the input signal was distributed. The

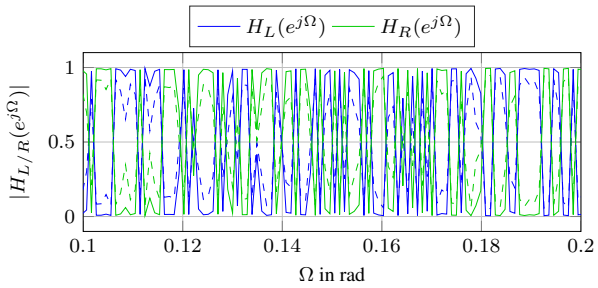


Figure 2: *Exemplary frequency response of the decorrelation filters for standard deviation $\sigma = 25$, stereo width $w = 1$ (solid) and $w = 0.1$ (dashed)*

alto voice at about 1000 Hz is mainly panned to the left channel, whereas the fundamental of the soprano voice at about 2200 Hz can be found in the right channel. The lower passband cutoff is set to $f_{\text{lo}} = 300$ Hz in this example. This results in the center panning of the bass voice showing a fundamental frequency of about 150 Hz.

3. TIME-DOMAIN IMPLEMENTATION

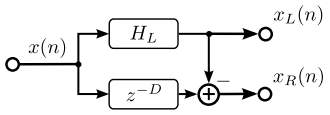


Figure 3: *Blockscheme of the time-domain realization*

The pseudo-stereo conversion can be performed in time-domain by FIR filtering the input signal with the impulse response

$$h_L(n) = \mathcal{F}^{-1}\{H_L(k)\} \tag{8}$$

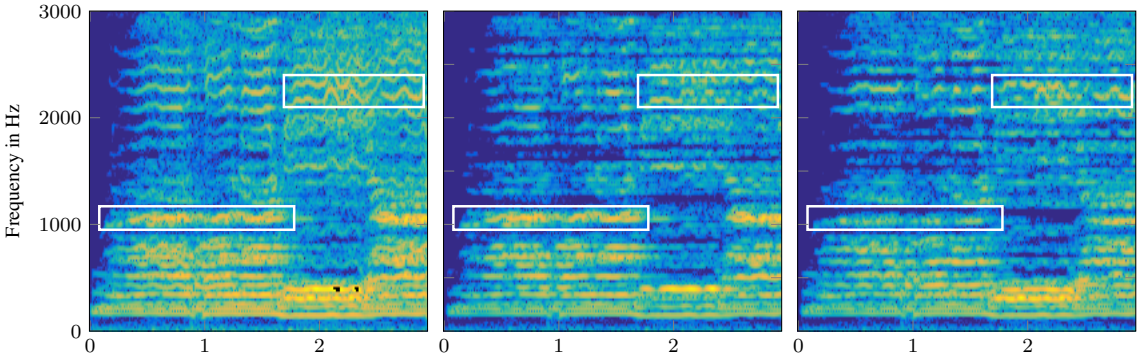
to obtain the left channel. Due to (3) and (4), the right channel can be computed by simply subtracting the FIR filter output from the time-delayed input signal

$$x_R(n) = x(n - D) - x_L(n), \tag{9}$$

where $D = \frac{N-1}{2}$ is the group delay of the FIR filter of length N. This only requires a single FIR filter together with a delay line and hence, is easy to realize on various platforms. On the other hand, the filter is static and the filtering operation for long impulse responses is computationally expensive. Furthermore, no dynamic control on the actual panning is provided and for example strong sources that are expected in the center of the stereo panorama could be diffusely panned. As the filter design is already done in the frequency domain it is a logical consequence to make use of fast convolution to reduce the complexity of the filtering and at the same time use the spectral information from the input signal do derive guidelines for the filter design process.

4. FREQUENCY-DOMAIN IMPLEMENTATION

First of all, the input signal $x(n)$ has to be transferred to the time-frequency domain with the help of a short-time fourier transform



(a) *Input signal* $x(n)$ (b) *Left output channel* $x_L(n)$ (c) *Right output channel* $x_R(n)$

Figure 4: *Exemplary M2S conversion of a choir sample*

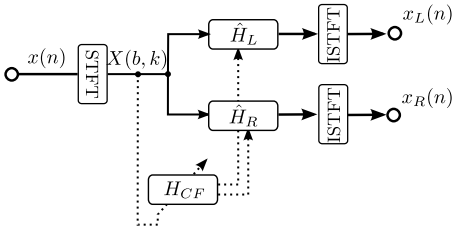


Figure 5: *Blockscheme of the frequency-domain realization*

(STFT) yielding the spectra $X(b,k)$, where b and k denote the block index and frequency index. The time-frequency representations of the output channels

$$X_{L/R}(b,k) = X(b,k) \cdot H_{L/R}(k) \tag{10}$$

are now computed with a dot-wise product and a following overlapping inverse STFT synthesis of the left and right output channel.

Until now, the pseudo stereo filters were applied statically. Hence, all sound sources of the mono input signal were diffusely panned to create a wide stereo panorama to enrich the input signal with a spacious character. Nevertheless, the diffuse panning of a dominant source like the singing voice in a musical piece constitutes a very unusual, discomforting listening experience. Therefore, the authors extended the M2S conversion scheme with a so-called center-focusing filter $H_{CF}(b,k)$ which aims to keep dominant sources in the center of the stereo panorama. Two features were considered to assess the dominance of a sound source and to compute $H_{CF}(b,k)$.

4.1. Normalized Energy Estimate

Assuming that signal sources expected to be in the center of the stereo mix feature the largest spectral energy, the center-focusing filter is defined as

$$H_{CF}(b,k) = (1-\alpha)\, H_{CF}(b-1,k) + \alpha\, X_n^2(b,k) \tag{11}$$

$$X_n(b,k) = \frac{|X(b,k)|}{\max\limits_k |X(b,k)|} \tag{12}$$

in the first variant. For every frame b, an amplitude-normalized spectrum $X_n(b,k)$ is computed using the maximum value of the current magnitude spectrum $|X(b,k)|$. $X_n(b,k)$ is then squared, weighted by α and added to $(1-\alpha)\, H_{CF}(b,k)$ to obtain a recursively smoothed estimation of the high energy regions in the spectrum.

4.2. Tonality

The second variant is based on tonalness measures. Several features to identify the tonalness of a signal were presented in [9]. In this study, the tonalness from the amplitude threshold feature $t_{\text{AT}}(b,k)$ and peakiness features $t_{\text{PK}}(b,k)$ are combined, resulting in the center-focusing filter

$$H_{CF}(b,k) = t_{\text{AT}}(b,k) \cdot t_{\text{PT}}(b,k). \tag{13}$$

4.3. Application of center-focusing filter

To involve the center-focusing filter, the fixed pseudo stereo filters $H_{L/R}(b,k)$ are weighted accordingly

$$\hat{H}_{L/R}(b,k) = H_{L/R}(k)\, H_{CF}(b,k) - \frac{1}{2}\,(1 - H_{CF}(b,k)) \tag{14}$$

to force the filters magnitude transfer function to a value of 0.5 for high values of $H_{CF}(b,k)$. The weighted pseudo stereo filters $\hat{H}_{L/R}(b,k)$ are computed for every frame b and applied to the input signal $X(b,k)$ as shown in Fig. 5. The center-focusing and the combined filter still meet the conditions of Eq. (1)-(4). The additional complexity, caused by the computation of $\hat{H}_{L/R}(b,k)$, is legitimated by the enriched naturality of the M2S stereo panorama.

5. EXPERIMENTS

A typical measurement to describe channel decorrelation or the width of a stereo panorama, especially for room measurements, is the so-called Interchannel Crosscorrelation Coefficient (ICC)

$$\text{ICC} = \max |\rho_{LR}(\tau)|, \tag{15}$$

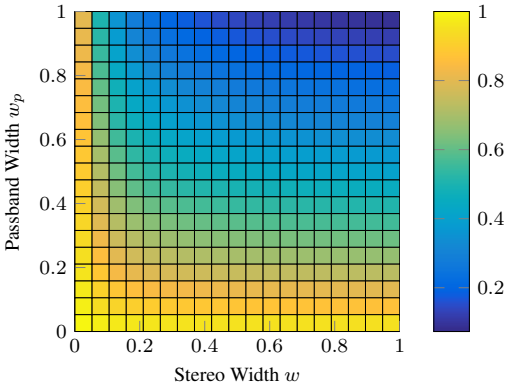


Figure 6: *ICC for white noise input and varying passband width and stereo width*

which is defined as the maximum value of the normalized cross-correlation function

$$\rho_{LR}(\tau) = \frac{\sum x_L(n) x_R(n-\tau)}{\sum x_L^2(n) \sum x_R^2(n-\tau)}. \quad (16)$$

The ICC for a white noise mono input after processing with the filter design from Eq. (6) and (7) in dependency of stereo width w and relative passband width $w_p = 1 - 2f_{\text{lo}}/f_s$ is plotted in Fig. 6. Fully correlated signals, resulting in ICC $= 1$, are obtained for $w = 0$ and $w_p = 0$. For increasing w and w_p the ICC value is decreasing continuously. The smallest value of 0.0637 indicates an almost full decorrelation for the maximum values of w and w_p. A FIR filter was designed using Eq. 6 with a length of $N = 2048$ samples, standard deviation $\sigma = 25$, and a sampling rate of $f_s = 44100$ in this example.

Another tool of audio engineers to graphically judge a stereo mix is the audio goniometer. It is basically a X-Y illustration of a stereo signal as shown in Fig. 7 and was originally created with oscilloscopes. A very narrow stereo mix or a mono signal would be illustrated as a straight line, whereas balanced stereo signals featuring level and phase differences tend to appear sphere-like. The stereo signal used for this example was a short pop music excerpt. The sphere-like shape of the original signal can be easily identified in Fig. 7. The downmix that is fed to the M2S system appears as a straight line whereas the M2S converted signal again features a sphere-like shape with a similar width as the original indicating a comparable stereo width.

To allow interested readers to experience the benefit of the M2S conversion, the authors provide some rendered wavefiles that can be found at `http://ant.hsu-hh.de/dafx15/M2S`.

6. CONCLUSION

This study proposed an approach to convert mono into stereo signals using pure amplitude panning. Necessary filter design constraints to perform downmix-compatible pseudo-stereo conversion without any timbral coloration were derived. The versatile design offers various control parameters to adjust stereo width and the cut-off frequencies of the passband to be processed. Subsequently, the described conversion system is implemented in time- and frequency-domain.

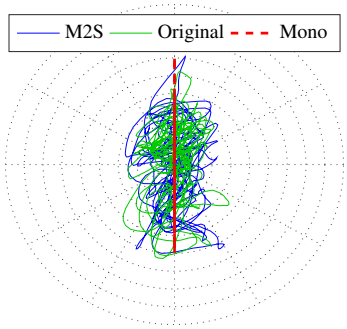


Figure 7: *Goniometer of a music sequence showing a reference stereo track, its downmix, and the M2S converted track*

The potential problem of a convenient listening experience due to a missing center focus is solved by extending the system with a center-focusing filter. This filter selects dominant spectral components according to predefined features, like normalized energy or tonalness, and repans them to the center.

Measurements proved the possibility of achieving nearly complete decorrelation, corresponding to extreme panning, when the conversion system is operated with extreme settings. For moderate settings, the mono-to-stereo conversion produces pleasing and natural sounding stereo panoramas that clearly enhance mono recordings with the arising spaciousness.

7. REFERENCES

[1] Holger Lauridsen, "Nogle forsog reed forskellige former rum akustik gengivelse," *Ingenioren*, vol. 47, pp. 906, 1954.

[2] Manfred R. Schroeder, "An artificial stereophonic effect obtained from a single audio signal," *J. Audio Eng. Soc*, vol. 6, no. 2, pp. 74–79, 1958.

[3] Benjamin B. Bauer, "Some techniques toward better stereophonic perspective," *J. Audio Eng. Soc*, vol. 17, no. 4, pp. 410–415, 1969.

[4] Robert Orban, "A rational technique for synthesizing pseudo-stereo from monophonic sources," *J. Audio Eng. Soc*, vol. 18, no. 2, pp. 157–164, 1970.

[5] Michael A. Gerzon, "Signal processing for simulating realistic stereo images," in *Audio Engineering Society Convention 93*, Oct 1992.

[6] Gary S. Kendall, "The decorrelation of audio signals and its impact on spatial imagery," *Computer Music J.*, vol. 19, no. 4, pp. 72–87, 1995.

[7] Christof Faller, "Pseudostereophony revisited," in *Audio Engineering Society Convention 118*, May 2005.

[8] Archontis Politis, Tapani Pihlajamäki, and Ville Pulkki, "Parametric Spatial Audio Effects," *Proc. of the 15th Int. Conference on Digital Audio Effects (DAFx-12)*, 2012.

[9] Sebastian Kraft, Alexander Lerch, and Udo Zölzer, "The Tonalness Spectrum: Feature-Based Estimation of Tonal Components," *Proc. of the 16th Int. Conference on Digital Audio Effects (DAFx-13)*, 2013.

DEVELOPMENT OF AN OUTDOOR AURALISATION PROTOTYPE WITH 3D SOUND REPRODUCTION

Erlend Magnus Viggen
Acoustics Research Centre, SINTEF ICT
Trondheim, Norway
erlendmagnus.viggen@sintef.no

Audun Solvang
Acoustics Research Centre, SINTEF ICT
Trondheim, Norway
audun.solvang@sintef.no

Jakob Vennerød
Acoustics Research Centre, SINTEF ICT
Trondheim, Norway
jakob.vennerod@sintef.no

Herold Olsen
Acoustics Research Centre, SINTEF ICT
Trondheim, Norway
herold.olsen@sintef.no

ABSTRACT

Auralisation of outdoor sound has a strong potential for demonstrating the impact of different community noise scenarios. We describe here the development of an auralisation tool for outdoor noise such as traffic or industry. The tool calculates the sound propagation from source to listener using the Nord2000 model, and represents the sound field at the listener's position using spherical harmonics. Because of this spherical harmonics approach, the sound may be reproduced in various formats, such as headphones, stereo, or surround. Dynamic reproduction in headphones according to the listener's head orientation is also possible through the use of head tracking.

1. INTRODUCTION

Noise from transportation and other activities impact many people over large areas. The sources in question (e.g. airplanes and motor vehicles) are often quite noisy, frequent, and cover large distances. For this reason, authorities throughout the world require computations of community noise in order to determine how many people and which areas are impacted. The computations use methods such as [1, 2] and result in *noise maps*, where noise levels are given as equivalent sound pressure levels in dB. A widely used such measure is the day-evening-night equivalent sound level L_{den}, where penalties are added to evening and night-time noise to reflect its additional impact.

However, from these numbers it is very difficult to get an intuitive understanding of the experienced impact of a particular noise situation, and next to impossible to get such an understanding for non-acousticians, such as the people making decisions that cause community noise, and the people in the affected communities. For instance, the short-term but strong noise from overflights may result in the same equivalent levels as the long-term but weaker noise from traffic, even though the two situations are radically different.

In order to aid the understanding of a given noise situation, it would be valuable to have a tool for *auralising* the situation. The term auralisation refers to an auditory representation of something; like visualisation, but with hearing instead of vision [3]. Additionally, such a tool could be used in order to auralise the effect of various noise mitigation scenarios, such as the placement and height of noise screens and the use of noise-reducing road surfaces. Letting noise affected communities listen to various scenarios in this way and make decisions among several options may also help reduce their noise annoyance [4].

The topic of this paper is the development of such an auralisation tool for providing a realistic representation of an outdoor noise situation such as traffic or industry. Our prototype tool is based on near-field recordings on a car, the Nord2000 model for outdoor sound propagation [1, 5, 6], and use of spherical harmonics [7] for spatial audio reproduction.

1.1. Outdoor auralisation

The topic of outdoor auralisation has been less explored than that of indoor auralisation, and these two topics pose different challenges. Outdoors, the distances between source and receiver are typically much longer, so that fully stochastic ray tracing is no longer a viable option due to the large number of rays required to cover faraway regions with sufficient density. As diffraction over and around screens and buildings must be well-handled, diffraction is a non-negligible effect. The number of relevant reflections is typically far lower outdoors than indoors, since the sound is not trapped in an enclosed geometry as it is in a closed room. On the other hand, due to complex ground effects, ground reflections in long-range sound propagation cannot be treated as simply as reflections typically are treated in indoor ray tracing.

Research on outdoor auralisation in the context of videogames and virtual worlds has been done by the GAMMA group at the University of North Carolina at Chapel Hill. In such cases, the geometry can often be decomposed into an open space domain and small domains around spatially separated objects of possibly complex shape. For example, Mehra et al. handled scattering and diffraction by various objects through an equivalent source method [8], while Yeh et al. coupled a ray tracing approach for the open space with a wave-based simulation method around the objects [9].

Some related work was done as part of the Swedish project *Listen – Auralization of urban soundscapes*, e.g. [10, 11, 12, 13]. The auralisation used for this project used a mix of the Nord2000 and Harmonoise [2] propagation models to construct 1/3 octave band filter levels to be applied to a source signal.

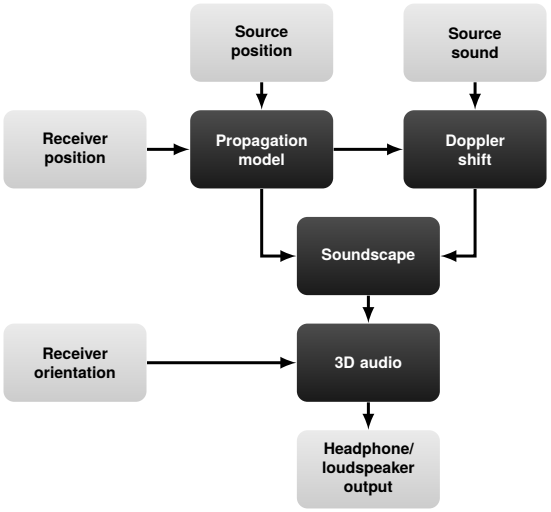


Figure 1: Simple schematic overview of the system, with inputs and outputs in light grey and core components in dark grey

1.2. The Nord2000 model

The Nord2000 model [1, 5, 6] is one of the world's most advanced comprehensive models for computing outdoor sound propagation. It compiles knowledge on many different aspects of outdoor sound, including detailed treatment of ground and structure reflections using the concept of Fresnel zones, the treatment of sound propagation over obstacles or valleys, atmospheric effects, and so forth.

In this model, the possible sound paths from source to receiver are first identified. Then, the sound propagation is subsequently calculated along each path, taking into account the terrain types and building heights along the vertical cross-section of the path. These sound paths may be reflected off buildings and screens, or diffracted around corners. At the receiver point, the various effects are summed to produce 1/3 octave band sound pressure levels. In cases where there are multiple paths from source to receiver, these may be summed fully or partially incoherently.

1.3. Our prototype

We aim to develop an augmented reality tool for an on-site listener to reproduce the noise from nearby virtual noise sources. The listener's head position and orientation may be followed using a positioning system and a head tracking system, and sound may be reproduced using a virtual auditory display (VAD). With a VAD, we combine headphones and head tracking to provide a realistic virtual localisation of the auralised sounds. For instance, if a new road is planned next to a residential area, the tool will allow a listener to walk around the area listening to a natural auralisation of the noise from the future road. The listener may also compare various noise-reducing scenarios.

The usefulness of on-site auralisation is supported by the literature, which indicates that a realistic visual representation of the auralised system is important [14, 15], and that listener mobility [16] and head movement [17, 18, 19] is important for the correct perception of the acoustic situation. With such an augmented reality tool, we would give the user the true visual view of the acoustic space while allowing them up to six degrees of freedom: three in their head position, and three in their head orientation. It is worth mentioning that the related field of virtual reality has matured rapidly in the last few years; a similar virtual reality tool where both the visual representation of the scene and the auralised sound is updated with the user's head orientation would also be a possibility.

Figure 2: Example of microphone positions for direct vehicle sound measurements

Currently, we are developing an off-line prototype of the auralisation model for traffic noise, where the receiver's position and orientation is specified manually. In this model, possible sound paths from source to receiver are determined and Nord2000 is used separately for each path to determine the sound propagation. Together with noise recordings, these sound paths can be used to determine a spherical harmonics representation of the sound at the listener point. From these spherical harmonics, it is possible to reproduce sound in a number of formats: headphones, stereo, surround, or virtual auditory display. We will specifically come back to the topic of stereo and VAD reproduction in section 2.6.

A simple overview of the system can be seen in Fig. 1. The various components of the system will be covered in the following section, followed by a description of the full prototype system and important lessons learned during development.

2. COMPONENTS OF AURALISATION PROTOTYPE

2.1. Source materials

The source materials were generated from recordings on the front and back of a Ford Mondeo, shown in Fig. 2. As a starting point, represented the vehicle as an omnidirectional source point, simply adding the microphone signals. This simple approach could be improved on by separating the various vehicle sound source types (e.g. engine and wheels) with separate recordings and source

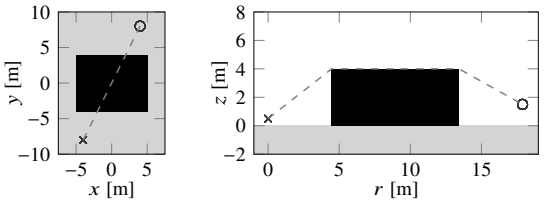


Figure 3: Vertically diffracted sound path (dashed) between source (cross) and receiver (circle), shown in a top-down view (left) and in the vertical cross-section along the path (right). The source and receiver are separated by a building (black).

directivities for each type. While this would require quite some additional work on establishing the different sources and their directivities, it would not be much more computationally heavy if the different source types could still be considered co-located point sources; the propagation model results could be re-used for each source type, each with an additional source directivity filter. However, as the research and implementation work of a more complex source model would be considerable, and as it is not certain that such a level of detail could be perceived in the aforementioned heavy traffic scenario, we retained the simple approach.

Recordings were made at 30, 50 and 80 km/h and the source materials for the velocity closest to the target vehicle velocity for the auralisation were chosen. The recordings are of finite length and in order to create soundscapes of arbitrary duration the recordings were prepared for repeated playback without audible splices. This was achieved by an overlap fade in/out with a fading function that keeps the intensity to unity for uncorrelated signals.

We assume that audible copies of the same sound degrade the fidelity due to the impression of multiple instances of the same car being present. In addition one will also have comb filter effects when adding time shifted correlated signals together. We therefore create a number of excerpts from the recordings. The number is dependent on the traffic density, the velocity of the cars and the length of the road. The number is chosen so that the expected time interval between a repeated sound signal is at least the length of the sound signal itself. We also include a restriction of minimum five excerpts to avoid the risk at low traffic densities of having very few excerpts which could be recognised in the auralisation.

2.2. Propagation model

At the base of the propagation model is a model of the geometry of the area to be simulated, including simple buildings and sound screens with constant height and constant horizontal cross-section, and a map of ground areas of various ground types. Different ground types are characterised by different flow resistivities which lead to different reflection properties. While the prototype currently only supports flat terrain with simple buildings and screens, the Nord2000 model can also handle more general geometries. In this model geometry, a single listener point is also placed at a given horizontal position and height.

We model a road with traffic as a line with evenly spaced source points placed along it. These static source points represent the different positions of a moving vehicle. The traffic is identified by a given vehicle speed and traffic density in passings per hour.

From each source point, possible sound paths to the listener are found. Straight-line sound paths are simple to find, and from the vertical terrain cross-section along these paths, the Nord2000 model can be used to determine the transfer function as 1/3 octave band levels from the source point to the receiver. In particular, the model uses the ground types and the terrain height profile (including man-made structures) in the vertical cross-section. An example of a vertical terrain cross-section is shown in Fig. 3.

Similarly, the Nord2000 model is used to determine the transfer function along reflected sound paths. The reflected paths are determined using a beam-tracing approach, with similar approaches described in the literature [20]. This beam-tracing is done using a 2D representation of surrounding buildings and noise screens where vertical walls represent objects for specular sound reflection in a horizontal plane.

Any object facing a noise source covers a limited horizontal sector seen from the source. This sector represents a sound propagation beam that will be reflected back from the object, corresponding to the mirror image of the source point. The reflected beam may hit other objects inside its angle, which are handled in a recursive manner. If the object covers a part of the beam, the beam is split into more narrow sub-beams separating between reflected and non-reflected sub-sectors. This builds a beam-tree representing all possible reflection paths down to a wanted maximum order of reflections. The final sound propagation lines (rays) from source points to receiver areas are easy to calculate accurately from this beam-tree. In our case, this technique is very efficient compared to traditional stochastic ray-tracing. One weakness of beam-tracing is that diffuse reflections cannot be determined. However, this does not matter in our case as such diffuse reflections are not accounted for in the Nord2000 model.

Beam-tracing is also useful for assessing reflections for sound propagation from discrete point sources to spatial receiver areas. Assuming reciprocity, it is equally efficient to assess sound propagation from source areas to discrete receiver points. The beam-tracing is simply done in reverse, since the resulting ray path is the same in both directions.

For each source point, each computed straight-line and reflected sound path from source to listener is stored, including its determined source-listener sound propagation time, its source-receiver transfer function as 1/3 octave band levels, and its azimuthal and polar angles from which the sound paths impinge on the listener.

As the Nord2000 model's processing of the sound paths from a source point to the listener is somewhat computationally expensive, a considerable amount of computational time may be saved for road sources by only directly computing a small and evenly spaced selection of the source points. Sound paths are organised by type, and the sound paths for the non-computed source points may be determined by interpolation of the sound paths of the same type from the neighbouring directly computed source points. As examples of sound path types, two sound paths that have been reflected by the same walls in the same order are of the same type, and two sound paths that have been diffracted around the same corner are of the same type.

2.3. Soundscape model

The propagation model serves as the basis of a time-varying multipath framework for processing the source material. In the next step, we determine the sound that impinges on the listener from each of the possible sound paths. We call this step the soundscape model.

The multipath transfer functions are estimated by a cubic

spline interpolation of the 1/3 octave band gain. We employ DFT-based overlap-add processing where transfer functions are applied corresponding to the vehicle's instantaneous position along its trajectory. The transfer functions are zero-phase, so the source material is pre-processed in order to create the necessary time delays and Doppler shifts.

In this pre-processing, each recording excerpt $p(t)$ is resampled through linear interpolation as

$$p_i(t) = p\left(t - \frac{r_i(t)}{c} + K\right), \tag{1}$$

where $p_i(t)$ is the received sound for sound path type i, $r_i(t)$ is the interpolated instantaneous sound path distance from source to receiver, c is the speed of sound, and K is an arbitrary time shift. As $p(t)$ is only defined for $t > 0$, K is chosen to avoid to avoid negative arguments,

$$K = -\min_{i,t}\left(t - \frac{r_i(t)}{c}\right). \tag{2}$$

Using this approach, each sound path type—the direct sound, each possible reflection type, and each possible diffraction type—gets the correct Doppler shift and relative time delay.

For each overlap-add window the multipath sound signals are encoded in spherical harmonics utilising the angle of incidence of the different sound paths and the relation presented below in (6). Representing the sound field in spherical harmonics allows decoding to arbitrary reproduction formats.

2.4. Spherical harmonics encoding

After the attenuation and angle of each incoming beam has been determined, the audio signals are combined into a spherical harmonics (SH) representation, frequently called Higher Order Ambisonics. The theory of sound field decomposition into a SH format has been extensively covered in the literature [21, 22, 23], so only a short description is included here.

Any sound field $p(r, \theta, \phi, \omega)$ can be decomposed with a modal representation such that

$$p(r, \theta, \phi, \omega) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} A_n^m(\omega) j_n(kr) Y_n^m(\theta, \phi) \tag{3}$$

where $A_n^m(\omega)$ are the frequency-domain spherical harmonic signals to be determined, $j_n(kr)$ are spherical Bessel functions of the first kind, and $Y_n^m(\phi, \theta)$ are the spherical harmonics [23]

$$Y_n^m(\theta, \phi) = \sqrt{\frac{2n+1}{4\pi}\frac{(n-m)!}{(n+m)!}} P_n^m(\cos\theta)\, \mathrm{e}^{\mathrm{i}m\phi} \tag{4}$$

where $P_n^m(\cos\theta)$ are the associated Legendre functions.

In practice, the series in (3) must be truncated to a finite order N for implementation. The above equation is valid for for interior problems where sources are placed outside the region of interest. Further, a virtual source (e.g. the direct sound from a vehicle or a surface reflection) can be encoded as a plane wave coming from a direction (θ_s, ϕ_s) with spherical harmonics:

$$A_{n,s}^m(\omega) = x_s(\omega) 4\pi \mathrm{i}^n Y_n^m(\theta_s, \phi_s)^* \tag{5}$$

where $x_s(\omega)$ is the source signal, including frequency-dependent distance attenuation. The total sound field which is a sum of all virtual sources can be described as:

$$A_n^m(\omega) = 4\pi \mathrm{i}^n \sum_s x_s(\omega) Y_m^n(\theta_s, \phi_s)^* \tag{6}$$

The sound field is now represented in a convenient format which can be stored and later reproduced with any 3D audio reproduction system, such as a loudspeaker or headphone Higher Order Ambisonics system, or mixed down to stereo or surround sound systems.

2.5. Traffic modelling

The different single pass-by signals can be used to create an artificial traffic sound signal. The traffic density is taken from the propagation model as passings per hour, and recomputed to a per-second pass-by rate λ.

Traffic pass-bys are modelled in a simple fashion as a Poisson process, so that the probability distribution of the waiting time Δt_{pass} between adjacent pass-bys follows an exponential distribution with probability density

$$P(\Delta t_{\text{pass}}) = \lambda \mathrm{e}^{-\lambda \Delta t_{\text{pass}}}. \tag{7}$$

The waiting time is drawn using an inverse transform sampling approach from the inverse cumulative distribution function of the exponential distribution,

$$-\frac{\ln(1-p)}{\lambda} \qquad \text{for } 0 \le p \le 1. \tag{8}$$

When drawing random and uniformly distributed numbers p, this function produces waiting times Δt_{pass} that are exponentially distributed with an average waiting time of $1/\lambda$, as they should be for a Poisson process.

Having drawn a set of waiting times $\Delta t_{\text{pass},n}$ such that the sum of these corresponds to the desired length of the traffic signal (e.g. a minute), the traffic signal can be assembled by summing the signals of individual pass-bys, delayed using the drawn waiting times.

2.6. Audio rendering

2.6.1. Stereo loudspeaker reproduction

The signal from a virtual microphone pointing in a direction given as (θ_m, ϕ_m) can be synthesised using the spherical harmonics representation of the soundscape model described in Sections 2.3 and 2.4 by the following relation

$$p(\theta_m, \phi_m, \omega) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} b_n A_n^m(\omega) Y_n^m(\theta_m, \phi_m), \tag{9}$$

where b_n dictates the directivity pattern.

The target is x-y stereo cardioid microphone configuration with opening angle of 180° that results in the following order-dependent directivity function:

$$b_n = \begin{cases} 1 & \text{for } n = 0, \\ \frac{1}{3} & \text{for } n = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

and the left and right loudspeaker signal:

$$p(\theta_l, \phi_l \pm 90^\circ, \omega) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} b_n A_n^m(\omega) Y_n^m(\theta_l, \phi_l \pm 90^\circ), \tag{11}$$

where (θ_l, ϕ_l) is the listener orientation.

2.6.2. Virtual auditory display

With virtual auditory display, we mean a headphone reproduction system with head-tracking that gives a realistic virtual localisation of the auralised sounds. Such a system is based on modelling the time-varying transfer function from the sound source to the ear drum in real time.

Here, we have implemented a VAD system based on the *mode-matching* approach [24] with virtual loudspeakers. This is done by considering at least $(N+1)^2$ such loudspeakers, evenly distributed on a spherical grid, radiating plane waves towards the origin where listener is placed. If each virtual loudspeaker $l = 1, 2, \ldots, L$ emits a signal $S_l(\omega)$, the sum of plane waves in the SH domain [23] can be expressed as

$$p(r,\theta,\phi,\omega) = 4\pi \sum_{l=1}^{L} S_l(\omega) \sum_{n=0}^{N} i^n j_n(kr) \sum_{m=-n}^{n} Y_n^m(\theta,\phi) Y_n^m(\theta_l,\phi_l)^* \quad (12)$$

which can again be equated with Equation 3, truncated to a finite order N, yielding [25]

$$\sum_{n=0}^{N}\sum_{m=-n}^{n} A_n^m(\omega) = \sum_{l=1}^{L} S_l(\omega) \sum_{n=0}^{N}\sum_{m=-n}^{n} Y_n^m(\theta_l,\phi_l)^* \quad (13)$$

since the spherical harmonics are orthogonal. This can again be expressed in matrix form

$$\mathbf{A} = \mathbf{SY} \quad (14)$$

where we must find the inverse of $\mathbf{Y}$ to extract the virtual loudspeaker signals. $\mathbf{Y}$ is here a matrix where each row contains complex conjugate spherical harmonics for each virtual loudspeaker angle. Often, the number of virtual loudspeakers will exceed the number of spherical harmonics, $(N+1)^2$, which causes $\mathbf{Y}$ to be non-square and the Moore-Penrose pseudoinverse must be used:

$$\mathbf{S} = \mathbf{DA}, \quad \mathbf{D} = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T \quad (15)$$

Note that the number of virtual loudspeakers must be greater than $(N+1)^2$ for $\mathbf{Y}$ to be (pseudo-)invertible.

When the virtual loudspeaker signals have been calculated by multiplying the SH signals with the pseudoinverse of $\mathbf{Y}$, it is straightforward to assign a Head-Related Transfer Function (HRTF) $H(\theta,\phi,\omega)$ to each virtual loudspeaker in order to obtain a binaural format. Mathematically, this is equivalent to converting the HRTF set into a SH-domain representation for each ear,

$$H(\theta,\phi,\omega) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} H_n^m(\omega) Y_n^m(\theta,\phi), \quad (16)$$

where, in practice, the series must be truncated to a finite order N. The SH representation of the sound field A_n^m must then be filtered with $H_{n,L}^m$ and $H_{n,R}^m$ for the left and right ear, respectively.

In binaural headphone reproduction, head-tracking is essential to obtain a realistic 3D sound experience [19]. Angular input from a head-tracker can be used to rotate the sound field in the opposite direction of the head rotation, stabilising the virtual sound field with respect to the real surroundings. With spherical harmonics, sound field rotation is achievable with Wigner-D weighting [26]. For a given Euler rotation with angles (α,β,γ), a $(N+1)^2 \times (N+1)^2$ rotation matrix $\mathbf{R}$ is defined. The calculation of this matrix can be calculated with real [27] or complex [28] spherical harmonics, but the details of the calculation are out of the scope of this paper.

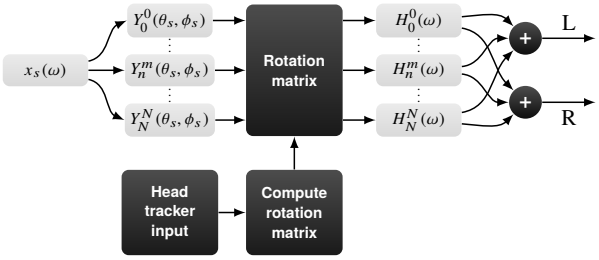


Figure 4: Block diagram of the binaural 3D audio reproduction system. Each SH-domain HRTF has two outputs $H_{n,L}^m$ and $H_{n,R}^m$.

Fig. 4 shows how the implementation is done. The block diagram shows only one source $x_s(\omega)$ from one direction (θ_s,ϕ_s), however, in practice all the direct sound and reflection sources must be mixed together after multiplying with the spherical harmonics coefficients. Subsequently, the signals are mixed with the rotation matrix, computed with input from the head-tracker. Finally, each resulting spherical harmonics channel is filtered with the corresponding SH-domain HRTF, and summed for the left and right ear. To reproduce sound for the VAD, we used a SH order $N = 4$ unlike the $N = 1$ order used for stereo reproduction.

2.6.3. High-frequency phase correction

The main advantage with the SH-domain representation is the ability to easily interpolate between measured HRTF angles, scalability and a flexible way of storing or transmitting the 3D sound field data. The main limitation of this approach is the high-frequency spatial aliasing given by the spherical harmonics series truncation. The rule-of-thumb is that near perfect reproduction only occurs for wave numbers $k < N/r$, where N is the truncation order and r is the reproduction radius in the loudspeaker array [29]. With HRTF reproduction, the reproduction radius is the distance from the ears to the centre of the virtual loudspeaker array (normally the centre of the head). This implies that one has to choose a relatively large N to achieve accurate high-frequency reproduction, which results in more computational load.

One possible solution for the high-frequency problem is to reduce the reproduction radius r at high frequencies. Since this radius is determined by the HRTF set, the high-frequency phase response of the HRTFs must be modified to accomplish this. By adjusting the phase such that the effective reproduction radius corresponds to $r_{\text{modified}} = N/k$, the high-frequency reproduction accuracy will improve. As a side effect, the interaural time difference (ITD) at high frequencies will no longer be correct. However, this is negligible since the binaural human auditory system does not rely on ITD at high frequencies, but rather on interaural level difference (ILD). The phase correction approach will improve the ILD reproduction accuracy. Without phase correction, both ITD and ILD will be incorrectly reproduced at high frequencies. This technique is described further and demonstrated in [25].

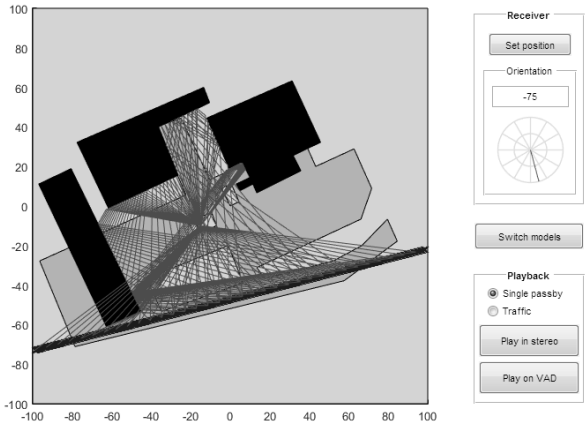


Figure 5: Overview of the prototype GUI for a 200 m × 200 m model of NTNU's P-25 parking lot with three buildings (black) and two ground types (lighter grey is grass, darker grey is asphalt). Sound paths are drawn from each computed source point (along the bottom black line) to a listener close to the geometry's centre.

3. TOTAL PROTOTYPE SYSTEM

Our prototype is developed as a single-threaded object-oriented MATLAB program. The prototype is presented to the user using a graphical user interface as shown in Fig. 5. The dominant aspect of the interface is a top-down view of the model's geometry, including source positions, listener position, structures, and the different ground areas, thus giving the listener a visual representation of what they are listening to. Additionally, the sound paths from the computed source points to the receiver may be plotted as they are being computed, making the underlying sound propagation computation more transparent to the listener. The listener may then play single pass-by or traffic sounds on stereo loudspeakers (Section 2.6.1) or a VAD (Section 2.6.2).

The user may choose where the virtual listener is positioned in the model geometry. As can be seen from Fig. 1, this requires recomputation of the entire propagation model, and since the other system components depend on this, they must also be recomputed. Such a full recomputation is time-consuming; on the order of half a minute in our implementation.

For stereo reproduction, the orientation of the virtual listener can also be altered. As Fig. 1 shows, this only requires recomputing the rendering from spherical harmonics to stereo, and thus changing the orientation of the listener is a quick process. Using the VAD, the listener's orientation can be changed in real time.

Additionally, the prototype can switch between two variations of the same geometry, in order to facilitate comparison between e.g. two alternative noise-reduction scenarios. If sound signals have already been computed for both geometries, this switch is almost instantaneous.

4. DISCUSSION OF INITIAL RESULTS

As the project is still in development we have not yet started to perform systematic listening tests. However, we have gained some useful insights during the development and internal testing of the prototype.

4.1. Performance

After performing some simple optimisations of our prototype, a full computation like the one shown in Fig. 5 takes on the order of half a minute to complete. This is dependent, however, on the complexity of the model: The required computation time increases with the number of reflections and the number of source points to be computed.

Our prototype has not been explicitly parallelised as MATLAB only supports this through the purchase of an additional toolbox, though we still benefit from MATLAB's parallelisation of some internal functions. However, the computation has a strong potential for parallelisation. In the propagation model, the computation for the various source points are fully independent of each other, and additionally the Nord2000 computation for each identified sound path is independent of the others. This is also the case for the soundscape model; the preprocessing of the direct sound and various reflected sound is independent as well as the overlap-add processing for each time window and sound path. However, the overlap-add processing is dependent on the preprocessing which is dependent on the propagation model.

The computation time is split evenly between the propagation model and the soundscape model. In the propagation model, most of the time is used for preparing and carrying out the Nord2000 calculations for the determined sound paths. Determining the paths themselves and extrapolating the calculated paths takes comparatively very little time. In the soundscape model, most of the time is spent on resampling the different recording excerpts to get the correct propagation delay and Doppler shift for each of the different sound path types.

Currently, our prototype would be too slow to track a listener moving at walking speed in real-time, especially with the $N = 4$ SH order used for VAD reproduction. However, this may be possible with a further optimised and fully parallelised implementation, with an increased amount of computational power. Alternatively, it may be possible to precompute some information for various listener positions in order to avoid having to perform recomputations on-the-fly.

4.2. Fidelity

Generating the source sound material recording the sound of a vehicle using microphones mounted on the vehicle itself is a straightforward and promising alternative to methods described in the literature [10], which are based on pass-by recordings combined with the use of simulated reverse propagation or additive synthesis in order to recreate an artificial source sound. However, our current sample set is limited to only one car. For such a tool to be generally useful for traffic auralisation, additional light and heavy vehicles must be recorded.

In the propagation model, a number of source points were not directly computed; their sound paths were instead interpolated from directly computed source points in their vicinity. While this does not have significant audible effects in an unblocked stretch of road, the interpolation is audible around structure edges with no horizontal diffraction in the model. In a case where all source points are directly computed, the sound from a single moving source would very suddenly become much louder when a virtual vehicle passes a corner so that its straight-line sound path to the listener is no longer blocked. With interpolation, there is instead a smooth artificial transition between the blocked and unblocked sound, essentially, a false diffraction effect.

Including horizontal diffraction in the model is thus important in order to ensure that virtual single vehicles moving behind or from behind a structure sound realistic. However, with no horizontal diffraction the effect is less pronounced in traffic simulations, due to the masking effect of multiple vehicles. Additionally, the false diffraction caused by the interpolation as described above also helps mask the lack of horizontal diffraction.

To begin with, we assumed that a virtual *x*-*y* stereo cardioid microphone configuration with opening angle of 90° would be suitable for decoding the spherical harmonics representation to a stereo loudspeaker reproduction. However, informal listening tests showed that the soundscape fidelity was noticeable degraded when the listener was positioned close to the road and facing away. We decided to use an opening angle of 180° instead, i.e. a back-to-back virtual microphone configuration, where the left and right loudspeaker signals correspond to cardioid microphones pointing 90° and −90° relative to the listener orientation, respectively.

Use of the spherical harmonics decomposition means that the soundscape can be auralised using virtual auditory display. In our implementation, this is done by reproducing the sound field with SH-based HRTFs through headphones, requiring head rotation compensation with a head-tracker. This can in theory facilitate perfect sound field reproduction, given a high enough SH truncation order, personalised HRTFs and headphone compensation, as well as a sufficiently low head-tracker latency.

In practice, the spherical harmonics must be truncated to a low order to reduce computational complexity, affecting high frequency reproduction accuracy. In our system we have limited the order to $N = 4$, giving accurate reproduction up to about 2.5 kHz. This can be somewhat compensated for by HRTF phase correction. Personalised HRTFs and headphone compensation is still an open research issue which we seek to investigate further in the future. The end-to-end head-tracker latency has been measured to around 100 ms in the MATLAB version, but should be decreased to < 60 ms in a final implementation [30].

5. CONCLUSION

In this prototype, we have adapted the Nord2000 model for outdoor noise propagation such that we calculate each sound path from the source point to the listener separately from the others instead of combining them. As a result, we find separately the direct, reflected, and diffracted sound paths that impinge on the listener from the sound source, with their 1/3 octave band transfer functions and their incoming azimuthal and polar angles. From this information and appropriate sound source material, we may represent the sound field at the listener point using spherical harmonics. The use of spherical harmonics allows for a wide range of alternatives for 3D sound reproduction: Headphones, stereo, surround, or virtual auditory display.

Conventional stereo reproduction was implemented with two virtual cardioid microphones in a back-to-back configuration, as this gives equal sound levels when the listener is facing towards or away from the road. Virtual auditory display was implemented with spherical harmonics-based HRTFs, and a head-tracker to compensate for head rotation. This gives the listener the ability to determine the direction of sound sources, and a more spatial experience of the soundscape.

As the system described in this paper is a prototype, there are many opportunities for further work. The prototype should be systematically validated, including listening tests and comparing the computed sound levels throughout the geometry with those found by noise mapping tools. Furthermore, source sound materials for a larger number of vehicles should be recorded, including various light and heavy vehicles and perhaps also trains. Synthesising the sound of road vehicles [13] is worth looking into. Realising the full augmented reality system as described in Section 1.3 is also a long-term aim in the continuation of this project.

6. ACKNOWLEDGEMENTS

We are grateful to our colleagues Odd Pettersen, Femke Gelderblom, and Rolf Tore Randeberg for helpful comments and/or assistance throughout the project. We are especially grateful to our now-retired engineer Asbjørn Ustad who performed the car recordings as part of an earlier project.

7. REFERENCES

[1] Birger Plovsing, "Nord2000. Comprehensive Outdoor Sound Propagation Model. Part 1: Propagation in an Atmosphere without Significant Refraction," Tech. Rep. AV 1849/00, DELTA, 2006.

[2] Erik Salomons, Dirk van Maercke, Jerome Defrance, and Foort de Roo, "The Harmonoise sound propagation model," *Acta Acustica United with Acustica*, vol. 97, no. 1, pp. 62–74, 2011.

[3] Mendel Kleiner, Bengt-Inge Dalenbäck, and U. Peter Svensson, "Auralization — an overview," *Journal of the Audio Engineering Society*, vol. 41, no. 11, pp. 861–875, 1993.

[4] Eveline Maris, *The social side of noise annoyance*, Ph.D. thesis, Leiden University, Leiden, 2008.

[5] Birger Plovsing, "Nord2000. Comprehensive Outdoor Sound Propagation Model. Part 2: Propagation in an Atmosphere with Refraction," Tech. Rep. AV 1851/00, DELTA, 2006.

[6] Birger Plovsing, "Proposal for Nordtest Method: Nord2000 — Prediction of Outdoor Sound Propagation," Tech. Rep. AV 1106/07, DELTA, 2010.

[7] Markus Noisternig, Alois Sontacchi, Thomas Musil, and Robert Holdrich, "A 3d Ambisonic Based Binaural Sound Reproduction System," in *Proceedings of the 24th AES International Conference*, Banff, 2003.

[8] Ravish Mehra, Nikunj Raghuvanshi, Lakulish Antani, Anish Chandak, Sean Curtis, and Dinesh Manocha, "Wave-based sound propagation in large open scenes using an equivalent source formulation," *ACM Transactions on Graphics*, vol. 32, no. 2, pp. 19:1–19:13, 2013.

[9] Hengchin Yeh, Ravish Mehra, Zhimin Ren, Lakulish Antani, Dinesh Manocha, and Ming Lin, "Wave-ray coupling for interactive sound propagation in large complex scenes," *ACM Transactions on Graphics*, vol. 32, no. 6, pp. 165:1–165:11, 2013.

[10] Jens Forssén, Tomasz Kaczmarek, Jesper Alvarsson, Peter Lund, and Mats E. Nilsson, "Auralization of traffic noise within the LISTEN project — preliminary results for passenger car pass-by," in *Euronoise 2009*, Edinburgh, Scotland, Oct. 2009.

[11] Peter Lundén, Marja Gustin, Mats E. Nilsson, Jens Forssén, and Björn Hellström, "Psychoacoustic evaluation as a tool

for optimization in the development of an urban soundscape simulator," in *Proceedings of the 5th Audio Mostly Conference: A Conference on Interaction with Sound*, 2010, p. 6.

[12] Marja Gustin, *Outdoor Auralization*, Master's thesis, KTH, Stockholm, 2010.

[13] Chinmay Pendharkar, *Auralization of road vehicles using spectral modeling synthesis*, Master's thesis, Chalmers University of Technology, Gothenburg, 2012.

[14] Dick Botteldooren, Bert De Coensel, Timothy Van Renterghem, Luc Dekoninck, and Dominique Gillis, "The urban soundscape: a different perspective," in *Sustainable mobility in Flanders: The livable city*, chapter 8, pp. 177–204. Ghent University, 2008.

[15] Yuliya Smyrnova and Jian Kang, "Determination of perceptual auditory attributes for the auralization of urban soundscapes," *Noise Control Engineering Journal*, vol. 58, no. 5, pp. 508–523, 2010.

[16] Paul Richmond, Yuliya Smyrnova, Steve C. Maddock, and Jian Kang, "Audio-Visual Animation of Urban Space," in *Theory and Practice of Computer Graphics*, 2010, pp. 183–190.

[17] J. Blauert, *Spatial Hearing - Revised Edition: The Psychophysics of Human Sound Localization*, The MIT Press, Cambridge, Massachusetts London England, 1997.

[18] Stephen Perrett and William Noble, "The effect of head rotations on vertical plane sound localization," *The Journal of the Acoustical Society of America*, vol. 102, no. 4, pp. 2325–2332, 1997.

[19] D. R. Begault, E. M. Wenzel, and M. R. Anderson, "Direct comparison of the impact of head tracking, reverberation, and individualized head-related transfer functions on the spatial perception of a virtual speech source," *Journal of the Audio Engineering Society*, vol. 49, no. 10, pp. 904–916, 2001.

[20] Thomas Funkhouser, Ingrid Carlbom, Gary Elko, Gopal Pingali, Mohan Sondhi, and Jim West, "A beam tracing approach to acoustic modeling for interactive virtual environments," in *Proceedings of the 25th annual conference on computer graphics and interactive techniques*. 1998, pp. 21–32, ACM.

[21] J. Daniel, S Moreau, and R. Nicol, "Further investigations of high-order Ambisonics and wavefield synthesis for holophonic sound imaging," in *AES 114th Convention*. 2003, Audio Engineering Society.

[22] B. Rafaely, "Plane-wave decomposition of the sound field on a sphere by spherical convolution," *The Journal of the Acoustical Society of America*, vol. 116, no. 4, pp. 2149–2157, 2004.

[23] E. G. Williams, *Fourier Acoustics: Sound radiation and nearfield acoustical holography*, Academic Press, 1999.

[24] M. A. Poletti, "Three-dimensional surround sound systems based on spherical harmonics," *Journal of the Audio Engineering Society*, vol. 53, no. 11, pp. 1004–1025, 2005.

[25] Jakob Vennerød, *Binaural reproduction of higher order Ambisonics: A real-time implementation and perceptual improvements*, Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, 2014.

[26] B. Rafaely and M. Kleider, "Spherical microphone array beam steering using Wigner-D weighting," *IEEE Signal Processing Letters*, vol. 15, pp. 417–420, 2008.

[27] M. A. Blanco, M. Florez, and M. Bermejo, "Evaluation of the rotation matrices in the basis of real spherical harmonics," *Journal of Molecular Structure: THEOCHEM*, vol. 419, no. 1, pp. 19–27, 1997.

[28] C. H. Choi, J. Ivanic, M. S. Gordon, and K. Ruedenberg, "Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion," *The Journal of Chemical Physics*, vol. 111, no. 19, pp. 8825, 1999.

[29] Darren B. Ward and T. D. Abhayapala, "Reproduction of a plane-wave sound field using an array of loudspeakers," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 6, pp. 697–707, 2001.

[30] Douglas Brungart, Alex J. Kordik, and Brian D. Simpson, "Effects of headtracker latency in virtual audio displays," *Journal of the Audio Engineering Society*, vol. 54, no. 1/2, pp. 32–44, 2006.

SWING RATIO ESTIMATION

Ugo Marchand
STMS IRCAM-CNRS-UPMC
1 pl. Igor Stravinsky, 75004 Paris, France
ugo.marchand@ircam.fr

Geoffroy Peeters
STMS IRCAM-CNRS-UPMC
1 pl. Igor Stravinsky, 75004 Paris, France
geoffroy.peeters@ircam.fr

ABSTRACT

Swing is a typical long-short rhythmical pattern that is mostly present in jazz music. In this article, we propose an algorithm to automatically estimate how much a track, a frame of a track, is swinging. We denote this by swing ratio. The algorithm we propose is based on the analysis of the auto-correlation of the onset energy function of the audio signal and a simple set of rules. For the purpose of the evaluation of this algorithm, we propose and share the "GTZAN-rhythm" test-set, which is an extension of a well-known test-set by adding annotations of the whole rhythmical structure (downbeat, beat and eight-note positions). We test our algorithm for two tasks: detecting tracks with or without swing, and estimating the amount of swing. Our algorithm achieves 91% mean recall. Finally we use our annotations to study the relationship between the swing ratio and the tempo (study the common belief that swing ratio decreases linearly with the tempo) and the musicians. How much and how to swing is never written on scores, and is therefore something to be learned by the jazz-students mostly by listening. Our algorithm could be useful for jazz student who wants to learn what is swing.

1. INTRODUCTION

1.1. Swing

Music is not always played exactly as written on the score. These deviations from the score constitute the musician personal interpretation and are precisely chosen by the musician to make music more lively or to convey an emotion.

In this paper we focus on timing deviations. We distinguish different types of systematic timing deviation. The *rubato* refers to a rhythmic freedom, in which tempo is sped up then slowed down to make music more expressive. It was usual in Romantic period. The *notes inégales* is a performance practice of the Baroque and Classical period in which some notes with equal durations are performed with unequal durations. Finally the *swing* is often found in jazz music but also other music styles as we will see in part 3. Our work focuses on the swing estimation.

Swing is present at a specific time level. It describes the specific interaction between note durations at the eight-note level. When swing is present, two consecutive eight-notes are played following a long-short pattern: the first eight-note is lengthened, while the second is shortened proportionally.

The swing ratio is defined by the ratio between the duration of the long eight note and the short one. Common swing ratios found in jazz music are 1:1 (no swing at all), 2:1 (triple feel), 3:1 (hard swing). These three example are presented in Figure 1. It should be noted that the swing ratio value is not limited to the values 2 or 3, but it can take all floating values between 1 and 3.5.

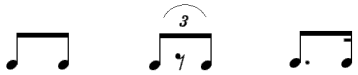


Figure 1: *Different swing ratio. From left to right : 1:1 (no swing), 2:1 (triple feel), 3:1 (hard swing)*

Swing ratios are not written on the score, they are implicit. Thus, swing ratio can vary considerably between two musicians or even inside a single music piece.

1.2. Related works

On swing. In [1], Friberg et al. study the exact duration ratio between the long eight-note and the short one in recordings. The author shows that the swing ratio varies linearly with the tempo. At slow tempi, the swing ratio can reach 3.5:1. At fast tempi, it goes down to 1:1. The authors also show that the minimum absolute duration of the short eight-note in the long-short pattern is around 100ms. This suggests a physical limit to swing ratio, maybe due to perceptual factors.

In [2], Honing et al. show that professional jazz drummers have an extremely precise control over the swing ratio they want to achieve. The authors also found no evidence that the swing ratio scale linearly with tempo, as was suggested previously.

On swing ratio estimation. In [3], Gouyon et al. introduced a swing modification tool. The swing ratio is estimated by two methods. The first one is based on the estimation of the position of the second peak d_s in the inter-onset-histogram which is supposed to corresponds to the duration of the short eight-note. The swing ratio is then $\frac{d_e+d_s}{d_e-d_s}$ where d_e is the theoretical duration of a non-swinging eight-note and d_s is the estimated duration of the second peak of the histogram. The second method compares the inter-onset-histogram to several predefined histogram models (representing different swing ratio), and choose the model that best represent the inter-onset-histogram.

In [4], Laroche presents a joint estimation of the tempo, the downbeat and the swing ratio. For this, he first estimates the time-positions where the energy in a given frequency band grows quickly. Then he exhaustively tests all the triplet (tempo, downbeat time, swing ratio) and keeps the one that have the best likelihood.

1.3. Paper overview and organization

In this paper, we propose a novel algorithm for estimating the swing ratio of a track (part 2). We then evaluate its performance for a task of finding music tracks with/without swing. For this, we present the test-set we created for the purpose of swing detection

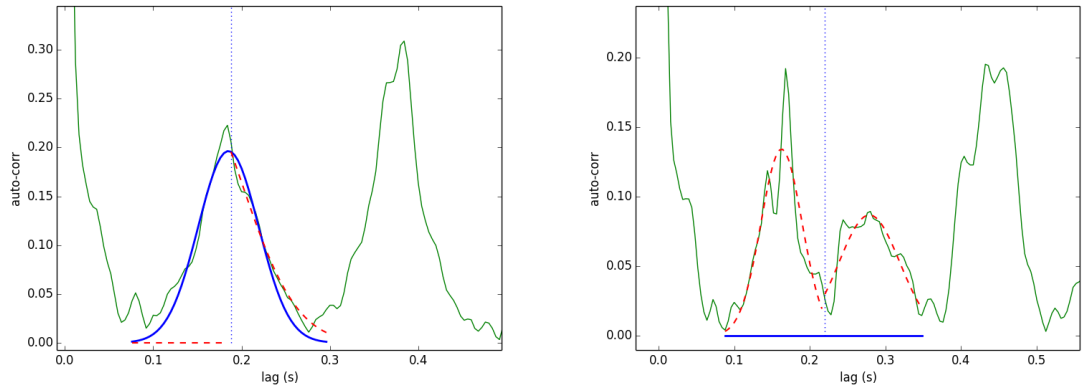


Figure 2: *Auto-correlation function for [Left] a signal without swing ('disco.00020.wav', first frame), [Right] a signal with swing ('jazz.00099.wav', first frame). The auto-correlation of the onset-energy-function (OEF) of the signal is the green thin line. The dotted vertical line (around 0.2 s.) corresponds to the theoretical duration of an eight-note. The thick blue line corresponds to the fitting of the peak of a non-swinging eight-note. The two red dotted lines correspond to the fitting of the peaks of the swing pattern (short and long eight-note).*

(part 3). We present our experiments and discuss their results in part 4.

2. SWING RATIO COMPUTATION

To estimate the swing ratio, we first compute an onset-energy-function[1] OEF (part 2.1). The auto-correlation function of the OEF allows highlighting the various metrical level of a rhythm pattern therefore the specific irregularities of the eight-note due to the swing (part 2.2). From the positions of the peaks in the auto-correlation function we then estimate the swing ratio (part 2.3).

2.1. Signal pre-processing

We calculate the onset-energy function $o(t)$ using the method proposed by Ellis [5][2].

$o(t)$ is then analyzed using a frame-analysis with a window length of 16s[3], and a hop-size of 1s[4]. For each frame, we then compute the normalized (at lag zero) auto-correlation of $o(t)$:

$$r(\tau) = \frac{1}{r(0)} \sum o(t)o(t-\tau) \quad (1)$$

2.2. Auto-correlation

The auto-correlation function of $o(t)$ allows highlighting the various metrical level of a rhythm pattern. Therefore it highlights the specific irregularities of the eight-note due to the swing. We illustrate this in Figure 2. The two panels illustrate the auto-correlation function for a signal without and with swing.

In a signal without swing (Figure 2 [Left]), there is one peak representing the tactus (the duration of a quarter-note) at 0.40s and one representing the duration of the eight-note at 0.20s. For a signal with swing (Figure 2 [Right]), the peak representing the quarter-note is still present at 0.45s, but the peak representing the duration of the eight-note is split into two peaks : one for the duration of the short eight-note at 0.15s and one for the duration of the long eight-note at 0.30s.

The goal of swing detection is to distinghish both cases.

2.3. Peak finding

2.3.1. Eight-note duration estimation

In order to distinguish the two cases we need to know the theoretical position of the eight-note without swing. This position is derived from an estimation of the tempo (the tactus-level). For this we use the joint estimation of beat, downbeat and tempo algorithm of Peeters et al. [6]. The tempo value is hypothesised to be constant over the whole track duration. If we denote by T the estimated tempo in bpm (beat per minute), the theoretical duration (in second) of the eight-note d_e is then computed as $d_e = \frac{1}{2}\frac{60}{T}$ (half of the duration in second of a quarter-note : $\frac{60}{T}$).

[1] A onset-energy-function is a function taking high values when an onset is present and low values otherwise.

[2] The OEF is computed as follow: first, the audio is resampled to 8kHz, then the short-term Fourier transform (STFT) magnitude (spectrogram) using 32 ms windows and 4 ms step between frames is computed. Then it is converted to 40 Mel bands. The Mel spectrogram is converted to dB, and the first-order difference along time is calculated in each band. Negative values are set to zero (half-wave rec-tification), then the remaining, positive differences are summed across all frequency bands. This signal is passed through a high-pass filter with a cutoff around 0.4 Hz to make it locally zero-mean, and smoothed by convolving with a Gaussian enveloppe about 20 ms wide.

[3] A long window is required because there is not always a swinging eight-note around each beat. For some tracks, the swing information can be very sparse. Thus it is necessary to have a long window.

[4] A 1s hop-size was arbitrarily chosen. Taking a smaller hop-size wouldn't make sense considering that there are, at most, 4 beats per 1 second in our database.

Table 1: *Distribution into swing, noSwing and ternary of the proposed GTZAN-rhythm test-set.*

	blues	classical	country	disco	hip-hop	jazz	metal	pop	reggae	rock	**Total**
# Track with swing	44	1	15	0	0	45	0	0	25	6	**136**
# Track with noSwing	52	92	80	98	100	54	94	99	74	92	**835**
# Track with ternary	4	7	5	2	0	1	6	1	1	2	**29**

2.3.2. Eight-note detection in the auto-correlation

The values of the eight-note duration can be derived from the peak positions in the auto-correlation function of OEF. We also use the height of the peaks and the width of the corresponding lobes to have information on the reliability of these peaks. Because peak *finding* algorithms (such as finding local maxima) do not lead to good results (since the auto-correlation is often noisy), we propose to use a peak *fitting* algorithm [7, 8]. In this a predefined shape is fitted locally to a signal. This shape is defined by a Gaussian function (this leads to better results than the use of a 2nd order polynomial function).

In the absence of swing a peak must be present at d_e, in the presence of swing this peak is split in two peaks in the two intervals $\left[\frac{d_e}{2}, d_e\right]$ and $\left[d_e, \frac{3d_e}{2}\right]$ respectively. We therefore looks for peaks in these three intervals.

- The first interval $\left[\frac{d_e}{2}, d_e\right]$[5] corresponds to the short eight-note.
- The second interval $\left[d_e, \frac{3d_e}{2}\right]$ corresponds to the long eight-note.
- The third interval $d_e \pm \frac{d_e}{2}$ correspond to a non-swinging eight-note (this interval is here only for illustration purpose, it is not used by our algorithm).

In order to estimate the peak within each interval, we use a non-linear least squares algorithm to fit a Gaussian function. Thus we get 3 parameters for each peak: the amplitude A, the standard deviation σ and the mean μ (μ defines the estimated peak position). The parameters of the short and the long eight-notes are denoted A_s, σ_s, μ_s and A_l, σ_l, μ_l respectively.

2.3.3. Swing detection

Given the 6 parameters described above, we propose the following set of rules to decide if there is swing or not. The Gaussian functions representing the short and the long eight-notes must comply with the following rules:

- positive amplitudes $A_s > 0$, $A_l > 0$
- small widths $\sigma_s < \frac{d_e}{4}$, $\sigma_l < \frac{d_e}{4}$
- positions in the right interval $\frac{d_e}{2} < \mu_s < d_e$ and $d_e < \mu_l < \frac{3d_e}{2}$

If all the conditions are met, the frame is classified as swing.

Illustrations: Swing detection is illustrated in Figure 2. On both figures, we show the auto-correlation of the OEF of the signal (green thin line). The theoretical duration of the eight-note d_e is given by the vertical dotted line. We also show the 3 peaks that were fitted as previously described: the thick blue line is the Gaussian fit for the non-swinging eight-note, and the two red dashed lines correspond to the fit of the short and the long eight-note of the swing pattern. Given the parameters of these fitted Gaussian functions and using the proposed rules described above, our algorithm decide that there is no swing on the Left panel of Figure 2 while there is swing on the Right panel.

2.3.4. Swing ratio estimation

For each frame classified as swing, its swing ratio s_r is finally computed as $s_r = \frac{\mu_l}{\mu_s}$.

Illustrations: For the previous example (Figure 2 [Right]), the swing ratio would be $s_r = \frac{0.28}{0.16} = 1.75$.

3. THE GTZAN-RHYTHM TEST-SET

There is currently no test-set on which the swing ratio is annotated. It is therefore difficult to compare numerically our results to previous works.

We therefore created our own test-set by extending the widely used GTZAN test-set for music genre[6]. While there exist controversies related to the use of the GTZAN test-set for music genre classification[9], its audio content is however represensative of real commercial music of various music genre. Also, this test-set has a good balancing between tracks with swing (blues and jazz music) and without swing.

Annotations: Each track of the test-set has been annotated into downbeat, beat/tactus (quarter-note) and eight-note positions[7]. In order to take into account tempo or rhythm pattern variations (the swing ratio is never constant over time) the annotation is performed over the whole duration of each track. A time region is said to contain swing if the eight-note positions are not exactly at the middle between the two adjacent quarter-notes.

The distribution into swing, non-swing and ternary is shown in Table 1. As can been seen, most of the swinging tracks are jazz (45 tracks) or blues (44 tracks). Swing is also present in country (15 tracks), reggae (25 tracks) and rock (6 tracks). The classical track classified as swing has a "march" rhythm. In total the test-set contains 136 swinging tracks, 835 non-swinging tracks and 29 ternary tracks.

4. EVALUATION

We evaluate our proposed algorithm for two tasks.

[5]The search ranges imply that the swing ratio is comprised between 1 and 3.

[6]The GTZAN test-set is made of 1000 audio excerpts of 30 s. duration evenly distributed in 10 music genres (blues, classical, country, disco, jazz, hip-hop, metal, pop, reggae, rock).

[7]We only annotated the eight-note position when swing exist or in case of ternary rhythm

Table 2: *Results of swing detection. Each column represents the results on a subset of the GTZAN-rhythm test-set. For this, each genre is subdivided into frames with swing and without. For each subset, we indicate its number of frames (# frame), the percentage of correct recognition of the automatic tempo estimation (Tempo-Estimation) and the two recalls for the classes swing and noSwing obtained using our algorithm (Exp-estimatedTempo and Exp-annotatedTempo).*

	blues		**classical**		**country**		**disco**		**hiphop**	
	swing	*noSwing*	*swing*	*noSwing*	*swing*	*noSwing*	*swing*	*noSwing*	*swing*	*noSwing*
# Frame	686	714	92	1308	294	1106	28	1372	0	1400
Tempo-Estimation (%)	63	74	41	63	61	72	100	96	0	97
Exp-estimatedTempo (recall %)	63.8	93.6	57.6	99.6	52.7	98.7	100	99.3	0	100
Exp-annotatedTempo (recall %)	95.3	94.1	92.4	99.6	84.4	97.6	100	99.3	0	100
	jazz		**metal**		**pop**		**reggae**		**rock**	
	swing	*noSwing*	*swing*	*noSwing*	*swing*	*noSwing*	*swing*	*noSwing*	*swing*	*noSwing*
# Frame	630	770	84	1316	14	1386	364	1036	112	1288
Tempo-Estimation (%)	37	56	83	64	100	87	57	75	62	84
Exp-estimatedTempo (recall %)	23.8	96.4	61.9	99.8	100.0	100	54.4	91.4	48.2	98.2
Exp-annotatedTempo (recall %)	60.2	97.4	76.2	99.1	100.0	99.6	93.4	95.3	83.0	99.1

In part 4.1, we test the ability of our algorithm (part 2.3.3) to correctly detect the tracks with swing and the ones without. Considering that the swing presence can vary over time, this is performed on a frame basis.

In part 4.2, we use our swing ratio annotation (part 2.3.4) for two experiments. First we test the commonly accepted belief that the swing factor decrease linearly with the tempo. Then, we test if the swing factor can be used to recognize the musicians.

4.1. Swing detection

4.1.1. Exp-estimatedTempo

The algorithm we proposed in part 2.3.3 allows deciding if a frame of a given track has swing or not. We test our algorithm on each frame of each track of the GTZAN-rhythm test-set. Because, the method proposed in 2.3.3 uses a 16 s. window (for the computation of the auto-correlation), we compare our estimation to the corresponding local annotation. The corresponding local annotation is computed as follow: a 16-sec frame is said to be swing if it contains at least a single swing annotation.

In our experiment we assimilated ternary frames to swinging frames. This is because our method can not distinguish a ternary frame from a swinging jazz frame with the 'triple feel' (swing ratio of 2:1).

Results: The results are presented in Table 2. The first, second and third rows indicate the frame repartition in genre and in swing/noSwing classes. For example, we see that there are much more swinging frames in jazz (630) and blues (686) than in hiphop (0) or pop (12). The results of Exp-estimatedTempo are presented in terms of Recall[8] for each class (in %).

As one can see, the Recall for the noSwing classes are all very good (from 91.4% to 100%). In the opposite, the Recall for swing is not that good and strongly depends on the genre: in jazz, it is only 23.8%, in rock is is 48.2%, country 52.7% and reggae 54.4%. It is slightly better for blues (63.8%), classical (57.6%) and metal (61.9%). The Recall is perfect for disco and pop. However the number of swing frames to be detected is very small for these two classes.

The summary of the results over the genres is indicated in Table 3. The noSwing Recall is very high (98%) which means that almost all noSwing frames have been correctly detected. While the swing Recall is low (49.5%) the swing Precision is rather high (84%) which means that in most cases when a frame have been detected as swing it is indeed annotated as swing.

Table 3: *Confusion matrix for the estimation of the classes noSwing and swing for Exp-estimatedTempo.*

Annotated / Detected	noSwing	Swing	Recall
noSwing	**11479**	217	98.14%
Swing	1162	**1142**	49.57%
Precision	90.91%	84.03%	

4.1.2. Tempo-Estimation

Since our swing detection algorithm relies on a previous tempo estimation (the one provided by the algorithm of [6]), we test if the bad results obtained for the swing detection (low Recall) are due to wrong tempo estimations. For this, we compare the tempo estimation T_e to our manually annotated tempo T_a. An estimation is said to be correct if $\frac{|T_a - T_e|}{|T_e|} < 4\%$.

Results: The detailed results are presented in Table 2. The lowest results are obtained for genre for which tempo variation or timing can be large (*rubato* in classical music, expressive music timings in jazz). Highest results are obtained for genre with a steady tempo (hip-hop, pop and disco). Our tempo estimation is based on the assumption that the tempo is constant over a track, so tracks with a lot of tempo changes are poorly classified.

Overall, 75.1% of the tracks have their tempo correctly estimated. This value falls down to 56.4% if we consider only the tracks that have swing. This can explain the bad results obtained

[8]The recall is the number of items detected for a class divided by the total number of annotated item for that class.

for the swing detection (Recall=49.5%) but the good results for the noSwing detection (98%).

4.1.3. Exp-annotatedTempo

Because of this, we redid the swing detection evaluation using manually annotated tempo instead of the automatically estimated one. The manually annotated tempo is derived from the beat annotations of our GTZAN-rhythm.

Results The detailed results are presented in Table 2. Using annotated tempo always improves the results for the swing class. The Recalls of blues, classical, country jazz, reggae and rock are 32 to 40% higher. The Recall for the class metal/swing is 15% higher, and the other Recalls were already perfect in the previous experiment (disco, pop).

Concerning the noSwing classes, the mean-Recall are now all comprised between 94.1 and 100%, which is better than in Exp1.

The summary of the results over the genres is indicated in Table 4. Using the annotated tempo improved a lot the results: the global mean-Recall increases from 73.9% to 90.6%.

These experiments show that we can estimate the swing knowing the 8th-note tactus. However deciding which level is the 8th-note remains a far more complicated problem.

Table 4: *Confusion matrix for the estimation of the classes noSwing and swing for Exp-annotatedTempo.*

Annotated / Detected	noSwing	Swing	Recall
noSwing	**11514**	182	98.44%
Swing	399	**1905**	82.68%
Precision	96.65%	91.27%	

4.2. Swing ratio as a function of tempo and artist

In this part, we use the annotation of the GTZAN-rhythm into swing ratio (part 2.3.4) for two experiments. First we test the commonly accepted belief that the swing factor decrease linearly with the tempo. Then, we test if the swing factor can be used to recognize the musicians.

4.2.1. Swing ratio as a function of tempo

In Figure 3, we illustrate the annotated swing ratio as a function of the tempo for the subset of our test-set corresponding to blues and jazz music (where swing is often present).

We test the commonly accepted belief that swing is a linearly decreasing function of tempo. This was proposed by the experiment of Friberg [1]. We superimposed to Figure 3, the linear regression (dotted line) used in Figure 1 of Friberg [1]. We can see that in our experiment the swing ratio does not scale linearly with the tempo. Our observed swing ratios are all far from the linear assumption proposed by Friberg (dotted line) at tempi < 130, and they does not show the same trend. Our results are therefore in agreement with the results of Honing et al. [2] saying that there may be no correlation at all between tempo and swing ratio. We also see that there may be a preference for the triple feel swing ratio (2:1) in our test-set, as about a third of our examples have a swing ratio around 2 (comprised between 1.9 and 2.1).

4.2.2. Swing ratio as a function of an artist

In Figure 3, we also indicates the artist[9] [10] corersponding to each swing ratio.

We test if there is a relationship between the artist and the swing ratio. We see that some artists may tend to play with low swing ratio (Robert Johnson) or high swing ratio (Magic Slim). Tracks from Stevie Ray Vaughan share the same swing factor but also the same tempo. (it is therefore difficult to say if the swing is characteristic of the artist or of the tempo). Except these examples predicting the musicians from its swing ratio and tempo seems difficult.

4.3. Applications

Our swing ratio estimation could be useful to jazz students who want to learn swing and to musicologists who would benefit from a swing estimation tool to study jazz music/ musicians.

How much to swing is often learned by students by listening to a lot of jazz recordings. It is not something that can be learned from books, and it requires a lot of practice to master. One application of this swing estimation could be a system helping jazz students to learn how much to swing :

- A jazz teacher plays a jazz piece and its swing ratio is estimated.
- The student plays the same piece and the computer says if the student swings enough or too much (compared to the teacher's reference), allowing the student to learn the right swing ratio.

It should be noted that we do not indicate what is the ideal swing ratio for a music piece.

5. CONCLUSION AND FUTURE WORKS

In this article, we presented an algorithm to automatically detect the swing presence in an audio music track and estimate its swing ratio. For the purpose of the evaluation of this algorithm, we proposed and shared the "GTZAN-rhythm" test-set, which is an extension of a well-known test-set by adding annotations of the whole rhythmical structure (downbeat, beat and eight-note positions). We showed that the algorithm allows a mean-Recall of 74% for the recognition of swing/ noSwing. By analysing the errors, we discovered that most of the errors are due to wrong tempo estimation which is used as input parameter of our algorithm. Using annotated tempo as input of our algorithm, allowed to reach 91% mean-Recall. We finally used our annotations to study the relationship between the swing ratio and the tempo and the musicians. Future works will concentrate on the development of a swing ratio estimation algorithm which does not necessitate this tempo estimation. Finally, given that swing ratio is a rhythmic descriptor, it could be used as input, among other audio descriptors, to machine learning algorithms to perform other Music Information Retrieval tasks.

[9] We are grateful to Bob Sturm for sharing the artist information of the GTZAN tracks.

[10] Given that the various tracks of an artist are coming from the same album we suppose that the musicians are the same for a given artist.

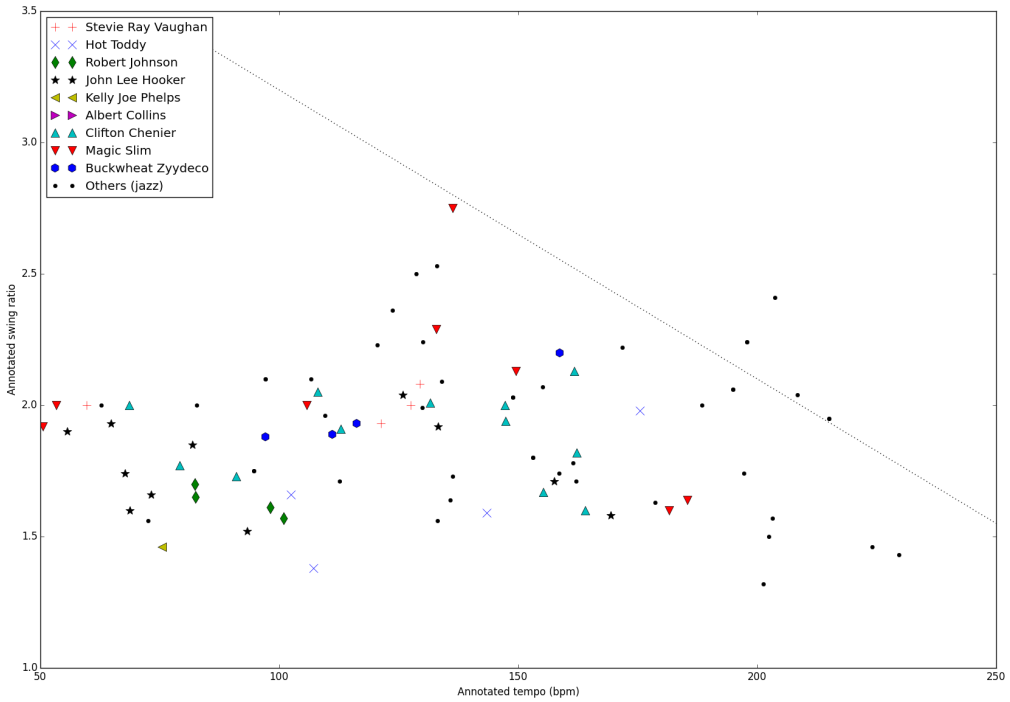


Figure 3: *Swing ratio as a function of tempo. Each type of point marker correspond to a different artist. The dotted line correspond to the swing as a function of tempo from the experiment of Friberg [1].*

Downloading the GTZAN-rhythm test-set.

We deeply thank Quentin Fresnel for helping us with the annotations of the "GTZAN-rhythm" test-set. The "GTZAN-rhythm" test set can be downloaded at `http://anasynth.ircam.fr/home/media/GTZAN-rhythm/`.

Acknowledgements

This work was founded by the French government Programme Investissements d'Avenir (PIA) through the Bee Music Project.

6. REFERENCES

[1] Anders Friberg and Andreas Sundström, "Swing ratios and ensemble timing in jazz performance: Evidence for a common rhythmic pattern," *Music Perception*, vol. 19, no. 3, pp. 333–349, 2002.

[2] Henkjan Honing and W Bas De Haas, "Swing once more: Relating timing and tempo in expert jazz drumming," 2008.

[3] Fabien Gouyon, Lars Fabig, and Jordi Bonada, "Rhythmic expressiveness transformations of audio recordings: swing modifications," in *Proc. Digital Audio Effects Workshop (DAFx)*, 2003.

[4] Jean Laroche, "Efficient tempo and beat tracking in audio recordings," *Journal of the Audio Engineering Society*, vol. 51, no. 4, pp. 226–233, 2003.

[5] Daniel PW Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.

[6] Geoffroy Peeters and Helene Papadopoulos, "Simultaneous beat and downbeat-tracking using a probabilistic framework: theory and large-scale evaluation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 6, pp. 1754–1769, 2011.

[7] Kenneth Levenberg, "A method for the solution of certain non–linear problems in least squares," 1944.

[8] Donald W Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial & Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[9] Bob L Sturm, "The gtzan dataset: Its contents, faults, and their effects on music genre recognition evaluation," *IEEE Transactions on Audio, Speech and Language Processing*, 2013.

WAVELET SCATTERING ON THE PITCH SPIRAL

*Vincent Lostanlen, Stéphane Mallat**

Department of Computer Science, École normale supérieure
Paris, France
vincent.lostanlen@ens.fr

ABSTRACT

We present a new representation of harmonic sounds that linearizes the dynamics of pitch and spectral envelope, while remaining stable to deformations in the time-frequency plane. It is an instance of the scattering transform, a generic operator which cascades wavelet convolutions and modulus nonlinearities. It is derived from the pitch spiral, in that convolutions are successively performed in time, log-frequency, and octave index. We give a closed-form approximation of spiral scattering coefficients for a nonstationary generalization of the harmonic source-filter model.

1. INTRODUCTION

The spectro-temporal evolution of harmonic spectra conveys essential information to audio classification, blind source separation, transcription, as well as other processing tasks. This information is however difficult to capture in time-varying, polyphonic mixtures. On one hand, spectrogram-based pattern recognition algorithms [1] are exposed to detection errors as they enforce strong constraints on the shape of harmonic templates. On the other, time-varying generalizations of matrix factorization [2] are underconstrained and thus may fail to converge to a satisfying solution. In this article, we address the characterization of harmonic structures without any detection nor training step.

Wavelets have long proven to provide meaningful, sparse activations as long as they operate on a dimension on which the signal has already some regularity. Although a single sine wave draws a regular edge on the time-frequency plane, a harmonic comb is made of distant sharp peaks over the log-frequency axis, an irregular pattern that is hard to characterize globally. This irregularity weakens the discriminative power of existing wavelet-like representations, such as Mel-frequency cepstral coefficients (MFCC).

To recover regularity across partials within a wavelet framework, we capitalize on the fact that power-of-two harmonics are exactly one octave apart. By rolling up the log-frequency axis into a spiral, such that octave intervals correspond to full turns, these partials get aligned on a radius. Consequently, introducing the integer-valued octave variable reveals harmonic regularity that was not explicit in the plane of time and log-frequency.

Once specified the variables of time, log-frequency, and octave index, our representation merely consists in cascading three wavelet decompositions along them and applying complex modulus. Thus, the constant-Q scalogram is "scattered" into channels over which main factors of time variability are disentangled and regularized, yet harmonicity is preserved.

* This work is supported by the ERC InvariantClass 320959. The source code to reproduce figures and experiments is available at www.github.com/lostanlen/scattering.m.

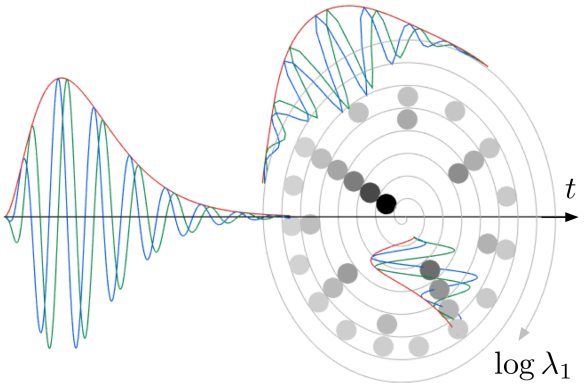


Figure 1: The spiral wavelet is a product of wavelets along time, log-frequency, and octave index. Blue and green oscillations represent the real and imaginary parts. The red envelope represents the complex modulus. Partials of an hypothetical harmonic sound are marked as thick dots.

Section 2 gives a formal definition of the spiral scattering transform. Section 3 introduces a nonstationary formulation of the source-filter model relying on time warps, and shows that its variabilities in pitch and spectral envelope are jointly linearized by the spiral scattering transform. Section 4 provides a visual interpretation of the spiral scattering coefficients of a nonstationary musical note.

2. FROM TIME SCATTERING TO SPIRAL SCATTERING

This section builds the spiral scattering transform progressively as a cascade of wavelet transforms along time, log-frequency, and octave index. All three variables share the same framework.

2.1. Time scattering

An analytic "mother" wavelet is a complex filter $\psi(t)$ whose Fourier transform $\widehat{\psi}(\omega)$ is concentrated over the dimensionless frequency interval $[1 - 1/2Q, 1 + 1/2Q]$, where the quality factor Q is in the typical range 12–24. Dilations of this wavelet define a family of bandpass filters centered at frequencies $\lambda_1 = 2^{j_1 + \frac{\chi_1}{Q}}$, where the indices $j_1 \in \mathbb{Z}$ and $\chi_1 \in \{1 \dots Q\}$ respectively denote octave and chroma:

$$\widehat{\psi}_{\lambda_1}(\omega) = \widehat{\psi}(\lambda^{-1}\omega) \quad \text{i.e.} \quad \psi_{\lambda_1}(t) = \lambda_1 \psi(\lambda_1 t). \qquad (1)$$

The wavelet transform convolves an input signal $x(t)$ with the filter bank of ψ_{λ_1}'s. We denote convolutions along time by the operator $\overset{t}{*}$. Applying complex modulus to all wavelet convolutions results in the "scalogram" matrix

$$x_1(t, \log \lambda_1) = |x \overset{t}{*} \psi_{\lambda_1}| \quad \text{for all } \lambda_1 > 0, \tag{2}$$

whose frequential axis is uniformly sampled by the binary logarithm $\log \lambda_1$. The scalogram x_1 localizes the energy of $x(t)$ around frequencies λ_1 over durations $2Q/\lambda_1$, trading frequency resolution for time resolution.

The constant-Q transform (CQT) $S_1 x$ corresponds to a low-pass filtering of x_1 with a window $\phi_T(t)$ of size T:

$$S_1 x(t, \log \lambda_1) = x_1 \overset{t}{*} \phi_T = |x \overset{t}{*} \psi_{\lambda_1}| \overset{t}{*} \phi_T. \tag{3}$$

To recover the amplitude modulations lost when averaging by ϕ_T in Equation (3), the time scattering transform also convolves x_1 with a second filterbank of wavelets ψ_{λ_2} and applies complex modulus to get

$$x_2(t, \log \lambda_1, \log \lambda_2) = |x_1 \overset{t}{*} \psi_{\lambda_2}| = ||x \overset{t}{*} \psi_{\lambda_1}| \overset{t}{*} \psi_{\lambda_2}|. \tag{4}$$

The wavelets $\psi_{\lambda_2}(t)$ have a quality factor in the range 1–2, though we choose to keep the same notation ψ for simplicity. Like in Equation (3), averaging in time creates invariance to translation in time up to T, yielding

$$S_2 x(t, \log \lambda_1, \log \lambda_2) = x_2 \overset{t}{*} \phi_T = ||x \overset{t}{*} \psi_{\lambda_1}| \overset{t}{*} \psi_{\lambda_2}| \overset{t}{*} \phi_T. \tag{5}$$

Due to the constant-Q property, $S_1 x$ and $S_2 x$ are stable to small time warps of $x(t)$ as long as they do not exceed Q^{-1}, i.e. one semitone. This implies that small modulations, such as tremolo and vibrato, are accurately linearized [3].

2.2. Joint time-frequency scattering

The time scattering transform defined in Equation (4) decomposes each frequency band separately, and so cannot properly capture the coherence of time-frequency structures, such as those induced by pitch contour. To remedy this, Andén et al. [4] have redefined the wavelets ψ_{λ_2}'s as functions of both time and log-frequency, indexed by pairs $\lambda_2 = (\alpha, \beta)$, where α is a modulation frequency in Hertz and β is a frequency along log-frequencies in cycles per octaves. The joint wavelets $\psi_{\lambda_2}(t, \log \lambda_1)$ factorize as

$$\psi_{\lambda_2}(t, \log \lambda_1) = \psi_\alpha(t) \times \psi_\beta(\log \lambda_1). \tag{6}$$

We write $\overset{\chi_1}{*}$ to denote convolutions along the log-frequency axis, i.e. along chromas. Wavelet scattering is extended to two-dimensional convolutions by plugging Equation (6) into the definition of x_2 in Equation (4):

$$x_2(t, \log \lambda_1, \log \lambda_2) = |x_1 \overset{t,\chi_1}{*} \psi_{\lambda_2}| = |x_1 \overset{t}{*} \psi_\alpha \overset{\chi_1}{*} \psi_\beta|. \tag{7}$$

The joint time-frequency scattering transform corresponds to the "cortical transform" introduced by Shamma and his team to formalize his findings in auditory neuroscience [10].

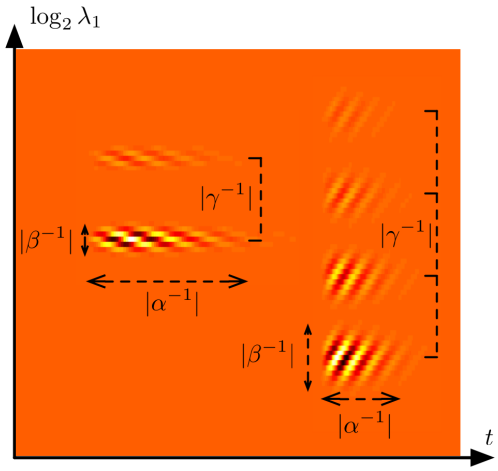


Figure 2: Two spiral wavelets $\psi_{\lambda_2}(t, \log \lambda_1)$ in the time-frequency plane, with different values of $\lambda_2 = (\alpha, \beta, \gamma)$. Left: $\alpha^{-1} = 120$ ms, $\beta^{-1} = -0.25$ octave, $\gamma^{-1} = +2$ octaves. Right: $\alpha^{-1} = 60$ ms, $\beta^{-1} = +0.5$ octave, $\gamma^{-1} = -4$ octaves. Darker color levels corresponds to greater values of the real part.

2.3. Spiral scattering

The time-frequency scattering transform defined in Equation (7) provides template-free features for pitch variability along time. However, it is unaware of the harmonic structure of voiced sounds, such as vowels or musical notes. The temporal evolution of this structure yields relevant information about attack transients and formantic changes, almost independently from the pitch contour.

In order to capture this information, we extend the joint time-frequency scattering transform to encompass regularity in time across octaves at fixed chroma, in conjunction with regularity along neighboring constant-Q bands. Just like wavelet filterbanks along time and log-frequency have been defined in the two previous subsections, we capitalize on harmonicity by introducing a third wavelet filterbank.

We roll up the log-frequency variable $\log \lambda_1$ into a pitch spiral making one full turn at each octave (see Figure 1). Since a frequency interval of one octave corresponds to one unit in binary logarithms $\log \lambda_1$, pitch height and pitch chroma in the spiral correspond to integer part $\lfloor \log \lambda_1 \rfloor$ and fractional part $\{\log \lambda_1\}$:

$$\log \lambda_1 = \lfloor \log \lambda_1 \rfloor + \{\log \lambda_1\} = j_1 + \frac{\chi_1}{Q}. \tag{8}$$

In this setting, the fundamental frequency f_0 is aligned with its power-of-two harmonics $2f_0$, $4f_0$, $8f_0$ and so forth. Likewise, the perfect fifth $3f_0$ is aligned with $6f_0$. As the number of harmonics per octave increase exponentially, the alignment of upper harmonics — $5f_0$, $7f_0$, and so forth — in the spiral is less crucial, because it can also be recovered with convolutions along chromas for β^{-1} of the order of a few semitones.

The wavelet ψ_{λ_2} is now defined as a product between wavelets in time, log-frequency, and octave index:

$$\psi_{\lambda_2}(t, \log \lambda_1) = \psi_\alpha(t) \times \psi_\beta(\log \lambda_1) \times \psi_\gamma(\lfloor \log \lambda_1 \rfloor). \tag{9}$$

Examples of the "spiral wavelet" ψ_{λ_2} are shown in Figure 2 for different values of α, β and γ. To ensure invertibility and energy conservation, the quefrencies β and γ must take negative values, including zero. We adopt the shorthand notation

$$\log \lambda_2 = (\log \alpha, \log |\beta|, \operatorname{sign} \beta, \log |\gamma|, \operatorname{sign} \gamma) \qquad (10)$$

to specify their indexing. In the special case $\beta = 0$, ψ_β is no longer a wavelet but a low-pass filter whose support covers one octave. By convention, the corresponding log-quefrency index is $\log |\beta| = -\infty$. The same remark applies to ψ_γ for $\gamma = 0$, which covers six octaves. Since its Fourier transform $\widehat{\psi}_{\lambda_2}$ is centered at (α, β, γ), the spiral wavelet ψ_{λ_2} has a pitch chroma velocity of α/β and a pitch height velocity of α/γ, both measured in octaves per second.

We write $\overset{j_1}{*}$ to denote convolutions across neighboring octaves. The definition for x_2 is comparable to Equations (4) and (7):

$$\begin{aligned} x_2(t, \log \lambda_1, \log \lambda_2) &= |x_1 \overset{t,\chi_1,j_1}{*} \psi_{\lambda_2}| \\ &= |x_1 \overset{t}{*} \psi_\alpha \overset{\chi_1}{*} \psi_\beta \overset{j_1}{*} \psi_\gamma|. \end{aligned} \qquad (11)$$

Rolling up pitches into a spiral is a well-established idea in music, if only because of circularity of musical pitch classes. It has been studied by Shepard [5], Risset [6], and Deutsch [7] to build paradoxes in perception of pitch, and is corroborated by functional imaging of the auditory cortex [8].

3. DEFORMATIONS OF THE SOURCE-FILTER MODEL

A classical model for voiced speech production consists in the convolution of a harmonic glottal source $e(t)$ with a vocal tract filter $h(t)$. Introducing independent deformations to both components brings realistic variability to pitch and spectral envelope. This section studies the decomposition of the deformed source-filter model in the spiral scattering transform.

3.1. Overview

Let $e(t) = \sum_n \delta(t-n)$ be a harmonic signal and $t \mapsto \theta(t)$ a time warp function. We define a warped source as $e_\theta(t) = (e \circ \theta)(t)$. Similarly, we compose a filter $h(t)$ and a warp $t \mapsto \eta(t)$ to define $h_\eta(t) = (h \circ \eta)(t)$. The warped source-filter model is the signal

$$x_{\theta,\eta}(t) = (e_\theta \overset{t}{*} h_\eta)(t) \qquad (12)$$

Observe that $\dot\theta(t)$ induces a change of fundamental frequency, whereas $\dot\eta(t)$ accounts for a local dilation of the spectral envelope $|\widehat{h}|(\omega)$. We show in this section that, for $\dot\theta(t)$ and $\dot\eta(t)$ reasonably regular over the support of first-order wavelets, the local maxima of x_2 are clustered on a plane in the (α, β, γ) space of scattering coefficients. This plane satisfies the Cartesian equation

$$\alpha + \frac{\ddot\theta(t)}{\dot\theta(t)}\beta + \frac{\ddot\eta(t)}{\dot\eta(t)}\gamma = 0. \qquad (13)$$

In a polyphonic context, this result means that harmonic sounds overlapping both in time and frequency could be resolved according to their respective source-filter velocities.

Our proof is driven by harmonicity and spectral smoothness properties — Equation (18) — and derives Equation (13) from the computation of wavelet ridges on the pitch spiral [9].

3.2. Source-filter factorization in the scalogram

Given λ_1 near the p^{th} partial $p\dot\theta(t)$ where $p \in \mathbb{N}$, we linearize $\theta(t)$ and $\eta(t)$ over the support of the first-order wavelet $\psi_{\lambda_1}(t)$. We work under the following assumptions:

(a) Q large enough to discriminate the p^{th} partial: $Q > 2p$,

(b) slowly varying source: $\|\ddot\theta/\dot\theta\|_\infty \ll \lambda_1/Q$,

(c) slowly varying filter: $\|\ddot\eta/\dot\eta\|_\infty \ll \lambda_1/Q$, and

(d) spectral smoothness: $\|\mathrm{d}(\log|\hat h|)/\mathrm{d}\omega\|_\infty \times \|1/\dot\eta\|_\infty \ll Q/\lambda_1$.

According to (a), partials $p' \neq p$ have a negligible contribution to the scalogram of the source at the log-frequency $\log\lambda_1$. For lack of any interference, this scalogram is constant through time, and we may drop the dependency in t:

$$|e \overset{t}{*} \psi_{\lambda_1}| \approx |\widehat{\psi}_{\lambda_1}(p)|. \qquad (14)$$

According to (b), the scalogram of the warped source $e_\theta(t)$ can be replaced by the scalogram of the original source translated along the log-frequency axis at the velocity $\log\dot\theta(t)$:

$$|e_\theta \overset{t}{*} \psi_{\lambda_1}|(t) = |e \overset{t}{*} \psi_{\lambda_1}|(\theta(t)) \approx |\widehat{\psi}_{\lambda_1}(p\dot\theta(t))|. \qquad (15)$$

According to (c), we linearize $\eta(t)$ over the support of $\psi_{\lambda_1}(t)$. According to (d), we approximate $\hat h(\omega)$ by a constant over the frequential support of the wavelet and factorize the filtering as a product:

$$(h_\eta * \psi_{\lambda_1})(t) \approx \hat h\left(\frac{\lambda_1}{\dot\eta(t)}\right) \times \psi_{\lambda_1}\left(\frac{\eta(t)}{\dot\eta(t)}\right). \qquad (16)$$

By plugging Equation (15) into Equation (16), the scalogram of the deformable source-filter model appears as a separable product:

$$|x_{\theta,\eta} \overset{t}{*} \psi_{\lambda_1}|(t) = |\widehat{\psi}_{\lambda_1}(p\dot\theta(t))| \times \left|\hat h\left(\frac{\lambda_1}{\dot\eta(t)}\right)\right|. \qquad (17)$$

3.3. Harmonicity and spectral smoothness properties

The second step in the proof consists in showing that the convolution along chromas with ψ_β only applies to $e_{1,\theta}$, whereas the convolution across octaves with ψ_γ only applies to $h_{1,\eta}$. Indeed, all wavelets are designed to carry a negligible mean value, i.e. convolving them with a constant yields zero. Therefore, the harmonicity and spectral smothness properties rewrite as

$$\left||e_\theta \overset{t}{*} \psi_{\lambda_1}| \overset{j_1}{*} \psi_\gamma\right| \approx 0 \quad \text{and} \quad \left||e_\theta \overset{t}{*} \psi_{\lambda_1}| \overset{\chi_1}{*} \psi_\beta\right| \approx 0. \qquad (18)$$

Gathering Equations (17) and (18) into the definition of spiral scattering yields

$$\begin{aligned} & x_{\theta,\eta} \overset{t,\chi_1,j_1}{*} \psi_{\lambda_2} \\ &= \left[\left(|e_\theta \overset{t}{*} \psi_{\lambda_1}| \overset{\chi_1}{*} \psi_\beta\right) \times \left(|h_\eta \overset{t}{*} \psi_{\lambda_1}| \overset{j_1}{*} \psi_\gamma\right)\right] \overset{t}{*} \psi_\alpha, \end{aligned} \qquad (19)$$

where the superscripts t, χ_1, and j_1 denote convolutions along time, chromas and octaves respectively.

3.4. Extraction of instantaneous frequencies

As a final step, we state that the phase of $(|e_\theta \overset{t}{*} \psi_{\lambda_1}| \overset{\chi}{*} \psi_\beta)$ is $\beta \times (\log \lambda_1 - \log p\dot{\theta}(t))$. By differentiating this quantity along t for fixed $\log \lambda_1$, we obtain an instantaneous frequency of $-\beta\ddot{\theta}(t)/\dot{\theta}(t)$. Similarly, the instantaneous frequency of $(|h_\eta \overset{t}{*} \psi_{\lambda_1}| \overset{j_1}{*} \psi_\gamma)$ is $-\gamma\ddot{\eta}(t)/\dot{\eta}(t)$. As long as

$$\alpha \geq \left|\frac{\ddot{\theta}(t)}{\dot{\theta}(t)}\beta\right| \quad \text{and} \quad \alpha \geq \left|\frac{\ddot{\eta}(t)}{\dot{\eta}(t)}\gamma\right|, \tag{20}$$

the envelopes of these two convolutions are almost constant over the support of $\psi_\alpha(t)$ [9]. We conclude with the following approximate closed-form expression for the spiral scattering coefficients of the deformed source-filter model:

$$x_2(t, \log\lambda_1, \log\lambda_2) = \left||e_\theta \overset{t}{*} \psi_{\lambda_1}| \overset{\chi_1}{*} \psi_\beta\right| \times \left||h_\eta \overset{t}{*} \psi_{\lambda_1}| \overset{j_1}{*} \psi_\gamma\right| \times \left|\widehat{\psi}_\alpha\left(-\frac{\ddot{\theta}(t)}{\dot{\theta}(t)}\beta - \frac{\ddot{\eta}(t)}{\dot{\eta}(t)}\gamma\right)\right|. \tag{21}$$

The Fourier spectrum $|\widehat{\psi}_\alpha(\omega)|$ of $\psi_\alpha(t)$ is a bump centered at the frequency α. Equation (13) follows immediately from the above formula. The same result holds for the averaged coefficients $S_2x = x_2 * \phi_T$ if the velocities $\ddot{\theta}(t)/\dot{\theta}(t)$ or $\ddot{\eta}(t)/\dot{\eta}(t)$ have small relative variations:

$$\left|\frac{\dddot{\theta}(t)}{\ddot{\theta}(t)} - \frac{\ddot{\theta}(t)}{\dot{\theta}(t)}\right| \ll T^{-1} \quad \text{and} \quad \left|\frac{\dddot{\eta}(t)}{\ddot{\eta}(t)} - \frac{\ddot{\eta}(t)}{\dot{\eta}(t)}\right| \ll T^{-1}. \tag{22}$$

In the example of a trombone signal, glissando can be modeled by $\ddot{\theta}(t)/\dot{\theta}(t)$ in the source-filter model, whereas the brassiness profile induces a timbral velocity $\ddot{\eta}(t)/\dot{\eta}(t)$. Figure 3 illustrates that these two velocities are stably disentangled and characterized.

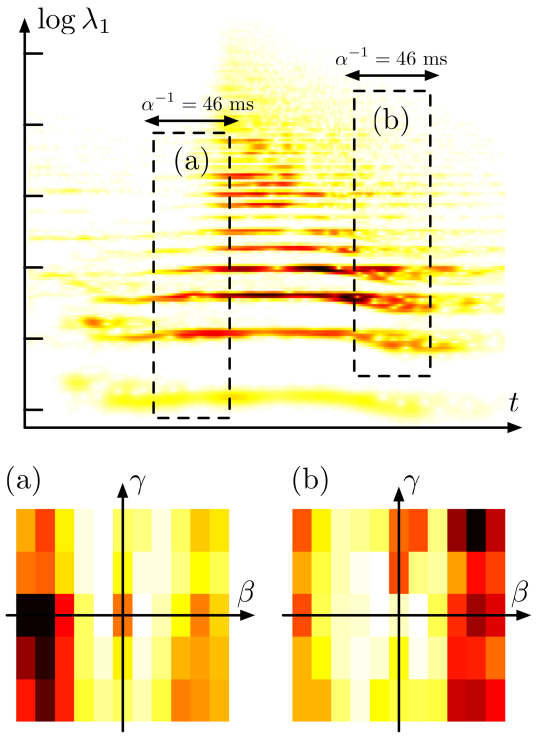


Figure 3: Top: scalogram of a musical note from Berio's *Sequenza V* for trombone, around 3'45". Observe that the attack part (a) has increasing pitch and increasing brightness, whereas the release part (b) has decreasing pitch and decreasing brightness. Bottom: spiral scattering coefficients for t and $\log\lambda_1$ specified by (a) and (b), α^{-1} fixed at 46 ms, β^{-1} ranging from -1 octave to $+1$ octave, and γ^{-1} ranging from -4 octaves to $+1$ octaves. As expected, highest values are concentrated in the bottom left corner for (a) and in the top right corner for (b).

4. CONCLUSIONS

The spiral model we have presented is well-known in music theory and experimental psychology [5, 6, 7]. However, existing methods in audio signal processing do not fully take advantage from its richness, because they either picture pitch on a line (e.g. MFCC) or on a circle (e.g. chroma features). In this article, we have shown how spiral scattering can represent the transientness of harmonic sounds.

5. REFERENCES

[1] C. Kereliuk and P. Depalle, "Improved Hidden Markov Model Partial Tracking Through Time-frequency Analysis," in *Proc. DAFx*, 2008.

[2] R. Hennequin, R. Badeau, and B. David, "NMF with Time-frequency Activations to Model Nonstationary Audio Events," *IEEE Trans. Audio, Speech, Language Process.*, vol. 19, no. 4, pp. 744–753, 2011.

[3] J. Andén and S. Mallat, "Scattering Representation of Modulated Sounds," *Proc. DAFx*, , no. 3, 2012.

[4] J. Andén, V. Lostanlen, and S. Mallat, "Joint Time-frequency Scattering for Audio Classification," *Proc. IEEE MLSP*, 2015.

[5] R. Shepard, "Circularity in Judgments of Relative Pitch," *J. Acoust. Soc. Am.*, vol. 36, no. 12, pp. 2346, 1964.

[6] J.-C. Risset, "Pitch control and pitch paradoxes demonstrated with computer-synthesized sounds," *J. Acoust. Soc. Am.*, vol. 46, no. 1A, pp. 88–88, 1969.

[7] D. Deutsch, K. Dooley, and T. Henthorn, "Pitch circularity from tones comprising full harmonic series.," *J. Acoust. Soc. Am.*, vol. 124, no. 1, pp. 589–597, 2008.

[8] J. Warren, S. Uppenkamp, R. Patterson, and T. Griffiths, "Analyzing Pitch Chroma and Pitch Height in the Human Brain," *Ann. N. Y. Acad. Sci.*, vol. 999, no. 17, pp. 212–214, 2003.

[9] N. Delprat, B. Escudié, P. Guillemain, R. Kronland-Martinet, P. Tchamitchian, and B. Torrésani, "Asymptotic Wavelet and Gabor Analysis: Extraction of Instantaneous Frequencies," *IEEE Trans. Inf. Theory*, vol. 38, pp. 644–664, 1992.

[10] K. Patil, D. Pressnitzer, S. Shamma, and M. Elhilali, "Music in our ears: the biological bases of musical timbre perception," *PLoS Computational Biology*, vol. 8, no. 11, 2012.

ANALYSIS/SYNTHESIS OF THE ANDEAN QUENA VIA HARMONIC BAND WAVELET TRANSFORM

Aldo Díaz, * *Rafael Mendes*

Department of Computer Engineering and Industrial Automation - DCA,
University of Campinas
Campinas, Brazil
aldodiaz64@gmail.com rafael@dca.fee.unicamp.br

ABSTRACT

It is well known that one of the challenges in musical instruments analysis is to obtain relevant signal characteristics and information for sound description and classification. In this paper we study the Peruvian quena flute by means of the Harmonic Band Wavelet Transform (HBWT), a convenient representation for the sound content based on its $1/f$ fractal characteristics. In order to identify a relationship between fractal characteristics of musical sounds, we developed two sound transformations and establish a comparison between quena, a recorder and melodica wind instruments. The sound transformations implemented were noise filtering and pitch-shifting while the sound classification was focused on the γ_p fractal attribute. Our work led us to the conclusion that the HBWT quena representation favored the implementation of sound transformations and that the γ_p fractal feature had great potential in musical instruments recognition and classification applications.

Keywords: sound fractal analysis; quena; $1/f$ noise; noise filtering; pitch-shifting.

1. INTRODUCTION

Fractal sound analysis/synthesis are techniques that aim to develop sound decomposition and reconstruction using a minimum set of relevant fractal attributes. Recently, fractal sound analysis techniques have obtained improved results in musical instruments recognition and classification compared to traditional state of the art methods [1, 2, 3].

Despite most of the fractal techniques use the *box counting* method to analyze the fractal characteristics of musical sound [4], a method called the Harmonic Band Wavelet Transform (HBWT) proposed a different approach inspired on Multirate Filter Banks (MFB) and Perfect Reconstruction (PR) digital signal processing techniques [5]. The HBWT provides a formal representation of musical sound in terms of harmonic noise sources with $1/f$ fractal properties that accurately follow the spectral behavior of a sound signal in the frequency domain. Moreover, the HBWT is based in solid PR and MFB techniques that use overlapping filters in order to achieve distortion-free signal reconstruction [6].

Despite the HBWT was applied as a relevant synthesis tool in audio coding and compression, our purpose was to contribute with the study of the Peruvian *quena* flute (Fig. 1) in the areas of fractal sound analysis and sound transformations. The implementation of two sound transformation techniques were described and discussed: the Harmonic band $1/f$ noise filtering and the pitch-shifting. Additionally, we performed a sound comparison between the quena, recorder and melodica wind instruments based on γ_p fractal parameter in order to illustrate the potential of our methodology in sound content description, musical instruments recognition and classification problems.

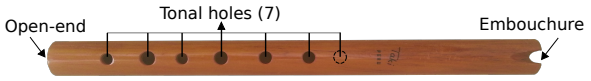


Figure 1: Traditional Peruvian quena with tuning in G major.

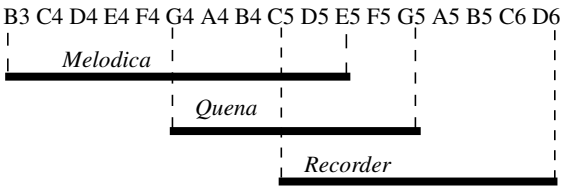


Figure 2: Pitch ranges of the analysis instruments where the overlap between them can be seen.

2. METHOD

It was generated a database with sound of quena, recorder, and melodica. The dataset was conformed by individual notes of one second duration played with *mezzo-forte* intensity, mono channel, sample rate $f_s = 44100$ Hz and 16 bits of resolution. The dataset characteristics are depicted in Fig. 2. A total of 27 sounds samples were collected: 8 for quena, 9 for recorder and 10 for melodica.

2.1. Sound decomposition

The HBWT belongs to the family of Spectral Modeling Synthesis (SMS) techniques [7]. Related to the representation of relevant sound components, the sound noise is indispensable in the perception of musical sound and inherent to the sound of musical instruments (see Fig. 4). Nevertheless, SMS contributions showing an explicit model for the stochastic component of sound (noise) are scarce [8]. In contrast, the HBWT provides a specific *pseudo-periodic* $1/f$ *noise model* for noise characterization [9]. A block

* The author is grateful to the founding of the Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq - Brazil (grant number 160186/2013-7) for funding this project.

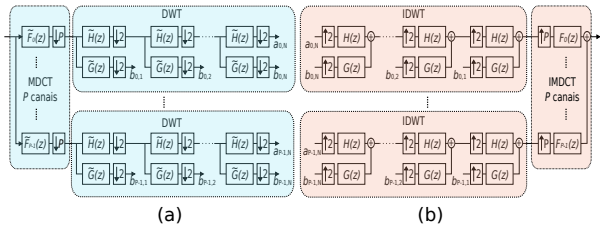


Figure 3: Structure of filters implementing the Harmonic Band Wavelet Transform. (a) Analysis bank. (b) Synthesis bank.

diagram of HWBT implementation is depicted in Fig. 3. As illustrated in Fig. 4, the HBWT model matches the sound signal fractal characteristics by decomposing the frequency harmonics in separated sidebands.

2.2. HBWT Analysis

The HBWT analysis structure is composed by perfect reconstruction filter banks of Modified Discrete Cosine Transform (MDCT) [6] and Discrete Wavelet Transform (DWT) [10]. In Fig. 5 it is shown the frequency response characteristic of an 8-channel MDCT and DWT with Daubechies order 11 filters.

The purpose of MDCT filters is to decompose the signal harmonics into two sidebands, each one containing half of a band that approximates the $1/f$ noise behavior. For MDCT implementation it was used cosine modulated type IV bases [11]:

$$g_{p,r}(k) = g_{p,0}(k - rP) \tag{1}$$

$$g_{p,0}(k) = w(k)\cos\left[\left(k - \frac{P+1}{2}\right)\left(p + \frac{1}{2}\right)\frac{\pi}{P}\right] \tag{2}$$

$$w(k) = \sin\left[\left(k + \frac{1}{2}\right)\frac{\pi}{2P}\right], \tag{3}$$

where $p = 0, \ldots, P-1$; $r \in \mathbb{Z}$; $k = 0, \ldots, 2P-1$. Considering an input signal $x(k)$ with fundamental frequency f_0 and average period P measured in number of samples then, the number of MDCT channels is equal to $P = f_s/f_0$. For instance, in Fig. 4 it is represented the spectrum of the first three harmonics of an A4 quena sound with $f_0 = 444.8$ Hz and the MDCT channels synchronized to the average period of the signal $P = 99$.

Each sideband is immediately decomposed by a bank of the Discrete Wavelet Transform (DWT). In wavelet theory, the wavelet representation bases execute shift and scaling operations of a wavelet function $\psi(t)$ defined as:

$$\psi_{\alpha,\tau}(t) = \frac{1}{\sqrt{\alpha}}\psi\left(\frac{t-\tau}{\alpha}\right), \tag{4}$$

where τ is called the shifting factor and $\alpha > 0$ is the scaling factor. Following the recommendations in [9], the DWT filter bank was implemented using Daubechies wavelets of order 11. The number of wavelet analysis levels N varies according to the instrument. In case of the quena, recorder and melodica, it was found $N = 4$ to be a proper number of levels for analysis.

The overall HBWT analysis procedure was implemented as follows:

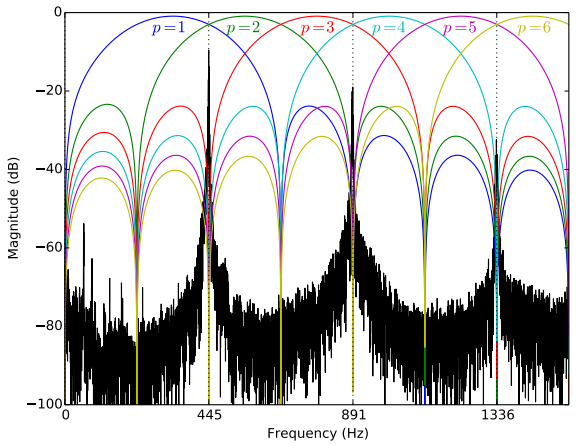


Figure 4: Quena A4 signal and MDCT channels synchronized to the average period P.

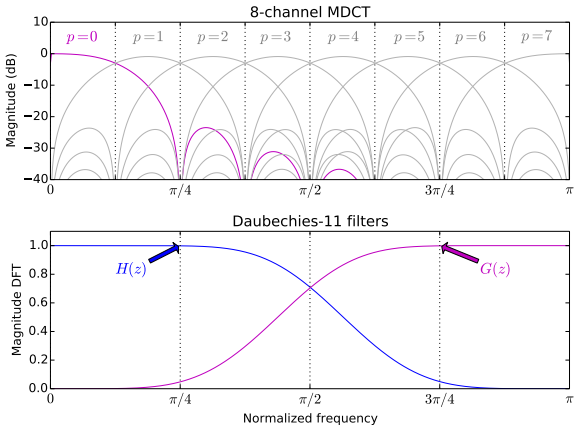


Figure 5: Frequency responses of MDCT and DWT filter banks.

Step 1
Decomposition of input signal $x(k)$ via a P-channel MDCT filter bank. Each MDCT filter $\tilde{F}_p(z)$ is band-pass type with bandwidth equals to π/P.

Step 2
Decimation bank of factor P[1].

Step 3
Wavelet decomposition via a P-channel DWT filter bank at the output of the decimation bank.

As a result, it is obtained the decomposition of the sidebands of the harmonics into two components: A deterministic component, represented by the scale coefficients $a_{p,N}$ and a stochastic component, represented by the wavelet coefficients $b_{p,n}$, where p

[1]The P decimation rate does not cause aliasing since the MDCT channels have a bandwidth equals to π/P.

is the MDCT channel index, n the DWT level index, and N the DWT total number of levels.

2.3. HBWT Synthesis

Step 1
Wavelet Reconstruction via a P-channel filter bank implementing the inverse of the DWT (IDWT)[2].

Step 2
Expansion bank of factor P[3].

Step 3
Output signal $\hat{x}(k)$ reconstruction via a P-channel IMDCT (inverse of MDCT)[4].

By means of HBWT decomposition and reconstruction, a sound signal is modeled by a set of periodic $1/f$ noise-like stochastic processes. A formal definition of $1/f$ noise-like processes can be found in [12]. One of the benefits of using the $1/f$ noise-like model is that only two parameters are needed to describe the complete behavior of a sound signal: The parameter σ_p^2 that controls the amplitude of the $1/f$ spectrum and the parameter γ_p that controls the slope of the pseudo-periodic $1/f$ noise-like sidebands of the spectrum.

The discrete-time harmonic band wavelets (DT-HBW), as defined in [13], are:

$$\xi_{n,m,p}(k) = \sum_{r=-\infty}^{\infty} \psi_{n,m}(r) g_{p,r}(k) \tag{5}$$

$$\zeta_{N,m,p}(k) = \sum_{r=-\infty}^{\infty} \varphi_{N,m}(r) g_{p,r}(k), \tag{6}$$

where $n = 1, 2, \ldots, N$; $m \in \mathbb{Z}$; $p = 0, 1, \ldots, P-1$; $\psi_{n,m}$ and $\varphi_{N,m}$ are the discrete-time ordinary wavelets and the corresponding scale residue function, respectively; $g_{p,r}$ are the MDCT functions of Eq. 2. Therefore, a signal $x(k) \in l^2$ can be expanded on a discrete-time harmonic band wavelet set according to:

$$x(k) = \sum_{p=1}^{P} \left(\sum_{n=1}^{N} \sum_{m=-\infty}^{\infty} b_{p,n}(m) \xi_{n,m,p}(k) + \sum_{m=-\infty}^{\infty} a_{p,N}(m) \zeta_{N,m,p}(k) \right), \tag{7}$$

where the $b_{p,n}(m)$'s and $a_{p,N}(m)$'s are the expansion coefficients and the corresponding harmonic-band scale residue coefficients at scale N, respectively. From *Proposition 3.4* in [13] it was deducted the resultant energy for the $b_{p,n}(m)$ analysis coefficients:

$$Var\{b_{p,n}(m)\} = \sigma_p^2 2^{n\gamma_p} \tag{8}$$

Considering the logarithm of the energies of each n-subband of a single p-sideband it was found a linear relationship for γ_p ($1/f$ noise slope) at each harmonic band wavelet analysis level n:

$$\log_2(Var\{b_{p,n}[m]\}) = \gamma_p n + const, \tag{9}$$

[2]This step performs the reconstruction of the sidebands.

[3]This step returns the properly bandwidth for the reconstructed spectrum of each sideband.

[4]The IMDCT filter bank selects the appropriate frequency range for the sidebands on each channel.

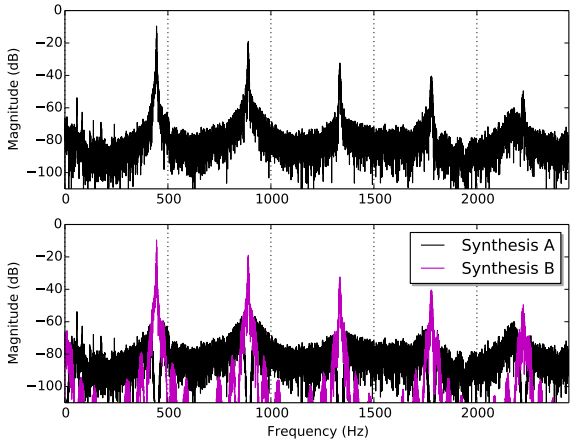


Figure 6: Above: First five harmonics of quena A4 sound. Below: Synthesis results of experiments A and B.

where the spectral component γ is directly related to the self-similarity attribute H_p (Hurst exponent) according to Eq. 10.

$$\gamma_p = 2H_p + 1 \tag{10}$$

In case of $1/f$ noise signals, or fractional Brownian motion processes (fBm), the typical values for the Hurst exponent are in the interval $0 < H < 1$, for a corresponding $1 < \gamma < 3$ [12].

3. RESULTS

3.1. Harmonic band $1/f$ noise filtering

We performed two sets of experiments in order to filter the harmonic band $1/f$ noise. The results are shown in Fig. 6. In the first experiment, called *Synthesis A*, the wavelet expansion coefficients $b_{p,n}$ were used as input to the synthesis bank with up to $N = 3$ scale levels. Separately, in experiment *Synthesis B*, the wavelet scale coefficients $a_{p,N}$ ($N = 3$) were used as input to the synthesis bank. *Synthesis A* resulted in the isolation of the harmonic band $1/f$ noise content associated to the sidebands of harmonics. In *Synthesis B*, the result was the $1/f$ noise-free harmonic signal (purely deterministic content).

These experiments benefited the study of the sound signal fluctuations related to the $1/f$ noise. It was found that the harmonic band $1/f$ noise content provided an important contribution to the sound in terms of perception. For instance, in *Synthesis A*, it was found that the harmonic band $1/f$ noise was related to the action mechanism of the instrument. In case of the quena, from the $1/f$ signal it was the retrieved the sound of the blowing air.

3.2. Pitch-shifting via HBWT

In sound synthesis, pitch-shifting is a technique that consists in modification (shift) of the fundamental frequency f_0 (pitch) of a sound. In terms of HBWT decomposition, pitch-shifting was interpreted as a frequency modulation process: The frequency content extracted in the initial signal analysis was transferred to new frequency bands in the synthesis.

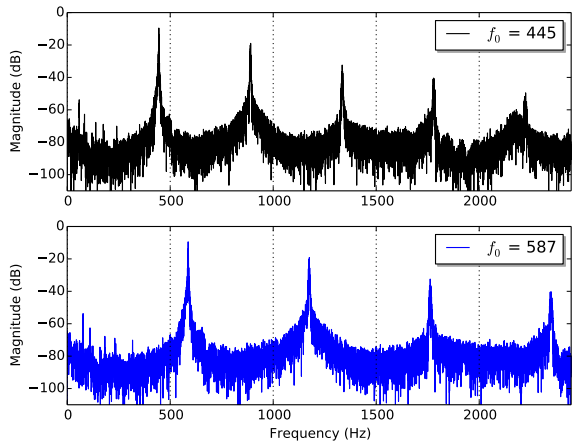


Figure 7: Pitch-shifting of A4 note (above) to D5 (below).

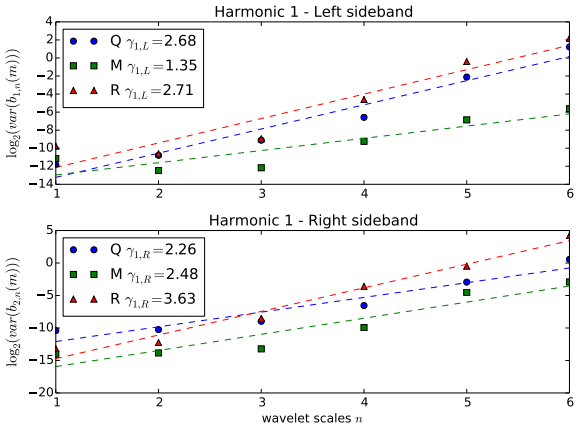


Figure 8: Quena (Q), melodica (M), and recorder (R) analysis of the left and right sidebands (channels $p = 1, 2$) for the first harmonic of C5 note with six subbands (wavelet scales $n = 1, \ldots, 6$).

The pitch-shifting implementation via HBWT consisted of two main steps: **Step 1:** Design of the MDCT synthesis filter bank (Eq. 2) with an appropriate number of channels equals to $P_2 = \left\lfloor \frac{f_s}{f_2} \right\rfloor$. This step allowed the modulation of the original signal pitch to a new $f_0^{'}$ frequency. **Step 2:** Apply the original HWBT analysis coefficients as input to the synthesis bank. In Fig. 7, it can be appreciated the modification of the A4 quena note ($f_0 = 445$ Hz) to D5 ($f_0^{'} = 587$ Hz). This technique resulted in high-quality pitch-shifted signals that preserved the acoustical characteristics of the original quena sound.

3.3. Quena fractal analysis and comparison

The comparison between wind instruments quena, melodica and recorder was established by analysis of the fractal parameter γ_p ($1/f$ slope). In the experiments we compared the fractal characteristics of the frequency harmonics ($1/f$ noise) for all instruments playing the same note. The parameters γ_p were computed by linear regression according to Eq. 9.

In Fig. 8 are shown the results for the f_0 harmonic of C5 note ($f_0 = 523.25$ Hz). The results showed a strong $1/f$ fractal behavior for the main harmonics of all instruments. This fact was corroborated by a strong Pearson correlation coefficient (greater than the 80%) between the variables of the linear regression.

By including the analysis of the Hurst exponent, we found between 7 to 12 fractal harmonics on the quena, 12 to 19 fractal harmonics on the recorder and 20 to 40 fractal harmonics on the melodica. Additionally, it was found that the quena fractal slopes γ_p were closer to the recorder rather than the melodica. The latter was supported by the fact that the quena and recorder shared similar physical characteristics like the typical resonator tube of the flutes. In contrast, the melodica analysis returned greater differences with respect to the quena that were probably caused due to the differentiated free-reeds action mechanism and other distinctions in execution like the blowing air pressure or articulations. Based on the $1/f$ fractal analysis, these findings supported our initial hypotheses that γ_p fractal parameter was a useful attribute for the description of musical sound with potential as a feature for musical instrument recognition and classification applications.

4. CONCLUSION

In this article it was presented an analysis of the quena by means of the *pseudo-periodic* $1/f$ *noise model* by using the *Harmonic Band Wavelet Transform*. The method and analysis described in this paper were performed for the first time on the andean wind instrument *quena*. By means of HBWT, it was obtained a suitable quena sound representation that enabled the development of interesting sound transformations and sound analysis applications.

The two sound transformation techniques we presented were: The *harmonic band* $1/f$ *noise filtering*, applied to the $1/f$ noise isolation or to the pure harmonic content reconstruction and the *pitch-shifting*, a modulation technique used to create new sound signals with a different fundamental frequency based on the analysis of a previous source. Both transformation techniques demonstrated acoustically interesting results. The last experiment was a comparison between the quena, recorder and melodica in terms of the fractal parameter γ_p. The results of the analysis pointed out that the γ_p fractal parameter was a useful attribute for sound description and characterization and a potential tool for musical instruments recognition and classification applications.

The results presented in this paper were part of a research project focused on the quena signal analysis and characterization based on distinctive fractal attributes. The methodology presented in this paper is scalable to the analysis of other types of musical instruments, thus we considered the study of additional andean instruments for the future. The source code of our Python implementation of the sound analysis/synthesis method described in section 2 and the sound examples to the experiments described in section 3 are available at the URL: `https://sites.google.com/site/aldodiazsalazar/`.

5. REFERENCES

[1] A. Zlatintsi and P. Maragos, "Multiscale Fractal Analysis of Musical Instrument Signals With Application to Recogni-

tion," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 4, pp. 737–748, Apr. 2013.

[2] S. Gunasekaran and K. Revathy, "Fractal dimension analysis of audio signals for Indian musical instrument recognition," in *2008 International Conference on Audio, Language and Image Processing*. July 2008, pp. 257–261, IEEE.

[3] A. Das and P. Das, "Fractal Analysis of different eastern and western musical instruments," *Fractals*, vol. 14, no. 3, pp. 165–170, 2006.

[4] P. Maragos, "Fractal signal analysis using mathematical morphology," in *Advances in Electronics and Electron Physics*, vol. 88, pp. 199–246. 1994.

[5] Pietro Polotti and Gianpaolo Evangelista, "Fractal additive synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 105–115, Mar. 2007.

[6] P. P. Vaidyanathan, *Multirate Systems And Filter Banks*, Prentice Hall, New Jersey, first edition, 1993.

[7] X. Serra and J. Smith, "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12, Jan. 1990.

[8] G. Evangelista, "Pitch-synchronous wavelet representations of speech and music signals," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3313–3330, 1993.

[9] P. Polotti and G. Evangelista, "Fractal Additive Synthesis via Harmonic-Band Wavelets," *Computer Music Journal*, vol. 25, no. 3, pp. 22–37, Mar. 2001.

[10] S. Mallat, *A Wavelet Tour of Signal Processing, The Sparse Way*, Academic Press, third edition, 2008.

[11] T. Nguyen and R. Koilpillai, "The theory and design of arbitrary-length cosine-modulated filter banks and wavelets, satisfying perfect reconstruction," *IEEE Transactions on Signal Processing*, vol. 44, no. 3, pp. 473–483, Mar. 1996.

[12] G. Wornell, "Wavelet-based representations for the 1/f family of fractal processes," *Proceedings of the IEEE*, vol. 81, no. 10, pp. 1428–1450, 1993.

[13] P. Polotti, *Fractal additive synthesis: spectral modeling of sound for low rate coding of quality audio*, Ph.D. thesis, École Polytechnique Fédérale de Lausanne, 2003.

Author Index