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`

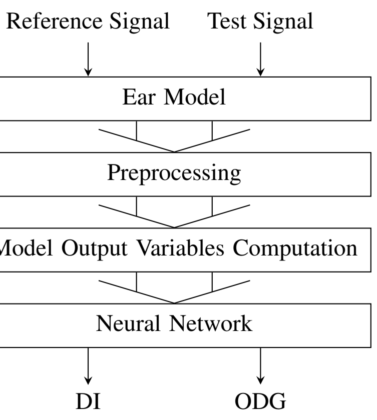

Figure 1: *Outline of the PEAQ algorithm.*

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

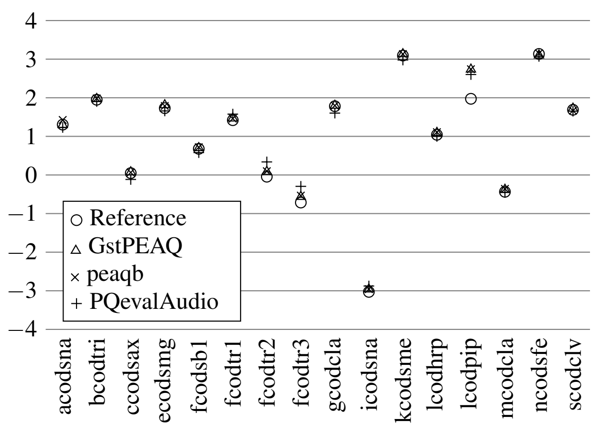

Figure 2: *Reference DI values (basic version) and DI values computed by the three implementations for the 16 test items.*

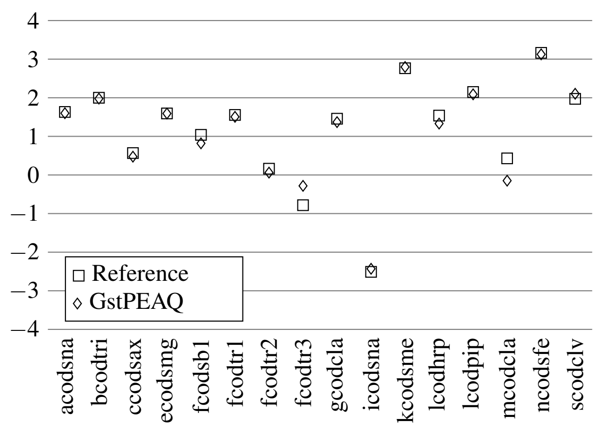

Figure 3: *Reference DI values (advanced version) and DI values computed by GstPEAQ for the 16 test items.*

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)

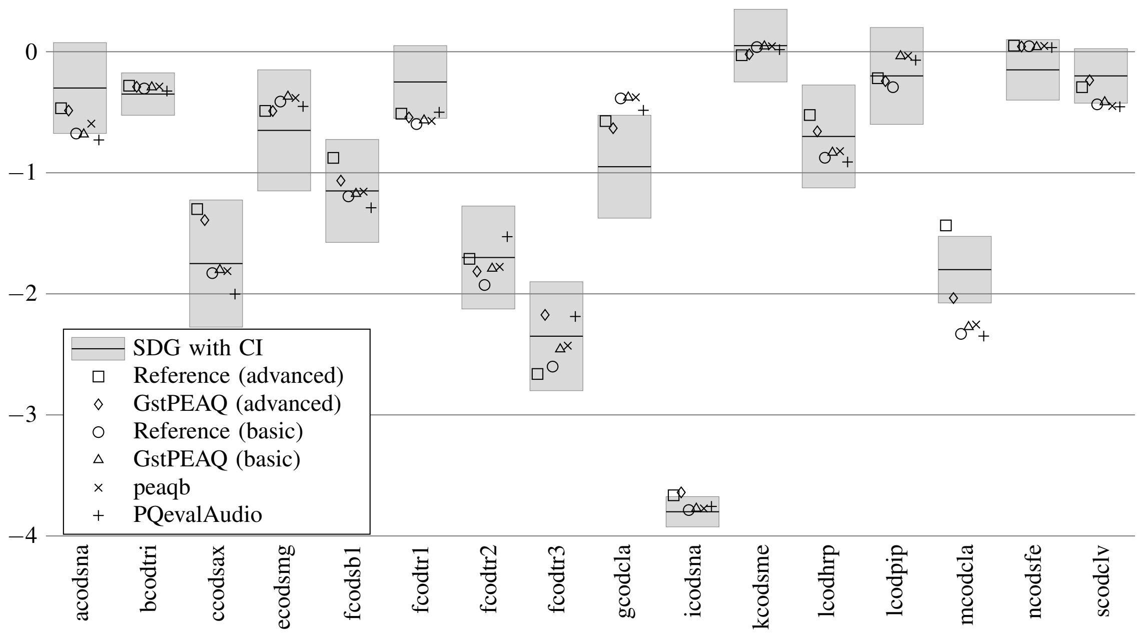

Figure 4: *SDG values with 95 % confidence interval (CI), reference ODG values, and ODG values computed by the three implementations for the 16 test items.*

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.