PRESIOUS

Predictive digitization, restoration and degradation assessment of cultural heritage objects

D4.1 – State of the Art Report on Object Matching, Reassembly and Completion Methods

Project Ref. No.	FP7-ICT-2011-9 – FP7-600533
Project Acronym	PRESIOUS
Project Start Date (duration)	1 Feb 2013 (36M)
Document Due Date	31 Jul 2013 (M6)
Actual Delivery Date	31 Jul 2013
Deliverable Leader	Tobias Schreck (UKON)
Document Status	final
Dissemination Level	PU

Address
Sem Sælands vei 5, Gamle fysikk 3rd floor
Gløshaugen, NTNU, NO-7491 Trondheim,
Norway

http://presious.eu

Contact person
Catalina Hellesø
catalina.helleso@ime.ntnu.no
+ 47 73 59 1452

Deliverable Identification Sheet

Project Ref. No.	FP7-ICT-2011-9 – FP7-600533	
Project Acronym	PRESIOUS	
Document Name	PRESIOUS-D4.1-31072013-v1.0	
Contractual Delivery Date	31 Jul 2013 (M6)	
Deliverable Number	D4.1	
Deliverable Name	State of the Art Report on Object Matching, Reassembly and Completion Methods	
Type	Document	
Deliverable Version	1.1	
Status	final	
Associated WP / Task	WP4 / T4.1, T4.2	
Author(s)	Anthousis Andreadis	AUEB-RC
	Pavlos Mavridis	AUEB-RC
	Georgios Papaioannou	AUEB-RC
	Robert Gregor	UKON
	Tobias Schreck	UKON
Other Contributors	Georgios Tsatiris	AUEB-RC
Project Officer	Philippe Gelin	
Abstract	Deliverable D4.1 is a State of the Art Report (STAR) on 3D object matching, reassembly and completion methods. In part A, we focus on 3D descriptor-based search methods for structured 3D objects, as a basis to retrieve similar 3D objects from a 3D object repository as input to repair and completion operations. Then, we review object repair methods classified by the size of the necessary repair operations, ranging from small-scale (e.g., hole filling, connectivity repair) to complex or large-scale repair (e.g., shape recombination or model-based shape completion). In part B, the report will review the current status of methods related to 3D puzzling algorithms for surface and volume data and reassembly algorithms, stochastic methods involved in the optimization of solutions for these two classes of problems and matching and retrieval methods. For this last part, this report will focus on matching and retrieval methods related to 3D data possessing structural information, such as connectivity and neighborhood information (meshes, regular/sparse volumes, hierarchical structures) and strongly irregular and uneven sampling (i.e. with no particular sampling assumptions).	
Keywords	3D object retrieval, 3D descriptors, interest point detectors, mesh repair, puzzle solving, surface alignment, surface matching, reassembly methods, complementary matching, shape repair, interpolation and completion.	
Sent to Internal Reviewer	10 Jun 2013 (PART B), 03 Jul 2013 (PART A)	
Internal Review Completed	02 Jul 2013 (PART B), 15 Jul 2013 (PART A)	

Circulated to Participants	21 Jul 2013
Read by Participants	30 Jul 2013
Approved by General Assembly	31 Jul 2013

Document Revision History

Date	Version	Author/Editor/Contributor	Summary of Changes
21/5/2013	0.1	Anthousis Andreadis, Pavlos Mavridis,	First draft of Part B: Segmentation, 3D matching and reassembly algorithms in 2D/2.5D/3D, GPU techniques and architectures.
24/5/2013	0.2	Anthousis Andreadis, Pavlos Mavridis, Georgios Tsatiris (external contributor)	Added ICP-based alignment methods.
27/5/2013	0.3	Georgios Papaioannou, Anthousis Andreadis, Pavlos Mavridis	Edited text, and corrections - final version (pre-review) of Part B.
8/7/2013	0.4	Anthousis Andreadis, Pavlos Mavridis	Edited text for reviewer comments integration and corrections
31/5/2013	0.1	Robert Gregor, Tobias Schreck	First outline and reference draft for Part A: Some parts in full text. Global and partial retrieval methods; defects, basic and complex shape repair.
10/6/2013	0.2	Robert Gregor, Tobias Schreck	Intermediate extended version of Part A
3/7/2013	0.3	Robert Gregor, Tobias Schreck	Further extended version of Part A. Sent for review.
19/7/2013	0.4	Robert Gregor, Tobias Schreck	Incorporated partner peer review comments to part A. Finalized the text.
21/7/2013	1.0	Georgios Papaioannou, Anthousis Andreadis, Pavlos Mavridis, Robert Gregor, Tobias Schreck	Merged final Parts A and B into joint document.
30/7/2013	1.1	Theoharis Theoharis	Updated document template

Table of Contents

D4.1 – STATE OF THE ART REPORT ON ... 1
OBJECT MATCHING, REASSEMBLY AND COMPLETION METHODS ... 1
DELIVERABLE IDENTIFICATION SHEET ... 2
DOCUMENT REVISION HISTORY ... 3
TABLE OF CONTENTS ... 4
REPORT D4.1 IN CONTEXT OF WORK PACKAGE 4 AND PRESIOUS ... 7
PART A ... 8
1. DESCRIPTORS FOR RETRIEVAL OF WHOLE 3D OBJECTS ... 8
1.1. 3D Descriptor Extraction Pipeline ... 8
1.2. Image-based Descriptors ... 9
1.3. Surface and Volume Descriptors ... 10
1.4. Structure Descriptors ... 12
1.5. Similarity Functions, Query Modalities, and Benchmarking ... 12
1.6. Conclusion on Global Retrieval Methods ... 13
2. PARTIAL 3D RETRIEVAL ... 15
2.1. 3D Local Feature Detection for Partial Retrieval ... 16
2.2. 3D Local Description for Partial Retrieval ... 17
2.3. Performance Assessment of Detection and Description Approaches ... 18
2.3.1. Repeatability and Description Stability ... 18
2.3.2. Quality of Interest Point Detection ... 19
2.4. Conclusion on Partial 3D Retrieval and Starting Points for PRESIOUS Task 4.2 ... 20
3. 3D MESH DEFECTS AND FLAWS ... 22
3.1. Introduction and Structure of the Problem ... 22
3.2. Defects of the Originating Physical Object ... 23
3.3. Local Connectivity Defects ... 24
3.4. Global Topology ... 26
3.5. Geometric Defects ... 27
4. BASIC SHAPE REPAIR ... 29
4.1. Local Approaches ... 29
4.2. Global Approaches ... 37
5. COMPLEX SHAPE REPAIR ... 41
5.1. Mesh Completion and Inpainting ... 41
5.2. Assembly-Based 3D Modeling ... 44
5.3. Symmetry-driven Segmentation and Part Repair ... 49
6. CONCLUSIONS ON SHAPE REPAIR ... 52
6.1. A Possible Approach for Reassembly and Repair of Fragmented and Incomplete Cultural Heritage Objects ... 52
6.2. Evaluation ... 53

APPENDIX OF PART A: ADDITIONAL REFERENCES TO MESH REPAIR 55
PART B 60
7. REASSEMBLY OF OBJECTS FROM FRAGMENTS 60
7.1. PROBLEM STATEMENT 60
7.2. COMPUTATIONAL COMPLEXITY 61
7.3. GENERAL SOLVING STRATEGY 61
8. THE ICP AGORITHM AND ITS VARIANTS 63
8.1. THE BESL-MCKAY METHOD 63
8.2. THE CHEN AND MEDIONI METHOD 64
8.3. *K*-D TREES AND OUTLIER HANDLING: THE ZHANG METHOD 66
8.4. EFFICIENT VARIANTS AND OPTIMIZATION 67
8.5. CONCLUSION 73
9. FACET EXTRACTION AND LABELING 75
9.1. MATHEMATICAL FORMULATION OF SEGMENTATION METHODS 75
9.1.1. Objectives of segmentation methods for 3D object reassembly 76
9.1.2. Related Problems 76
9.2. GENERAL CATEGORIZATION 77
9.3. SEGMENTATION CRITERIA 78
9.3.1. Planarity of Various Forms 78
9.3.2. Fitting to simple primitives 78
9.3.3. Difference in geometric normals 79
9.3.4. Curvature 79
9.3.5. Average Geodesic distance 79
9.3.6. Shape Diameter Function 80
9.4. CLASSIFICATION OF SEGMENTATION METHODOLOGIES 80
9.4.1. Region Growing 80
9.4.2. Multi-Region Growing 82
9.4.3. Hierarchical Clustering 83
9.4.4. Iterative Clustering – Classification 84
9.4.5. Contour Extraction 85
9.4.6. Spectral Analysis 88
9.4.7. Implicit Methods 89
9.4.8. Critical Points based 90
9.5. DISCUSSION AND COMPARISON 91
9.5.1. Post Processing 91
9.5.2. Performance on noisy data 92
9.5.3. Massively Parallel Implementation 92
9.5.4. Comparison Table 92
10. PAIRWISE MATCHING METHODS 94
10.1. JIG-SAW MATCHING 94

10.1.1. A Probabilistic Image Jigsaw Puzzle Solver *95*
10.1.2. Jigsaw Puzzles with Pieces of Unknown Orientation *96*
10.2. TWO-DIMENSIONAL MATCHING 98
10.2.1. A Multiscale Method for the Reassembly of Two-Dimensional Fragmented Objects *99*
10.2.2. Archaeological Fragment Reconstruction Using Curve-Matching *100*
10.2.3. Globally Consistent Reconstruction of Ripped-Up Documents *101*
10.2.4. A Texture Based Matching Approach for Automated Assembly of Puzzle *103*
10.2.5. Automatic Color Based Reassembly of Fragmented Images and Paintings *106*
10.3. RESTRICTED THREE-DIMENSIONAL (2.5D) MATCHING 108
10.3.1. Bayesian assembly of 3D axially symmetric shapes from fragments *109*
10.3.2. A System for High-Volume Acquisition and Matching of Fresco Fragments: Reassembling Theran Wall Paintings *112*
10.4. THREE-DIMENSIONAL MATCHING 113
10.4.1. On the Automatic Assemblage of Arbitrary Broken Solid Artefacts *114*
10.4.2. Reassembling Fractured Objects by Geometric Matching *118*
10.4.3. Pairwise Matching of 3D Fragments Using Cluster Trees *123*
10.5. METHOD COMPARISON 125
10.5.1. Discussion *129*
11. MULTI-PIECE MATCHING (ASSEMBLY) METHODS 130
11.1. A GLOBAL APPROACH TO AUTOMATIC SOLUTION OF JIGSAW PUZZLES 131
11.2. REASSEMBLING FRACTURED OBJECTS BY GEOMETRIC MATCHING 132
11.3. ARCHAEOLOGICAL FRAGMENT RECONSTRUCTION USING CURVE-MATCHING 133
11.4. GLOBALLY CONSISTENT RECONSTRUCTION OF RIPPED-UP DOCUMENTS 134
11.5. JIGSAW PUZZLES WITH PIECES OF UNKNOWN ORIENTATION 135
11.6. AUTOMATIC RECONSTRUCTION OF ARCHAEOLOGICAL FINDS – A GRAPHICS APPROACH 136
11.7. METHOD COMPARISON 137
11.7.1. Discussion *138*
12. CURRENT ADVANCES IN GENERAL PURPOSE GPU ARCHITECTURES AND TECHNIQUES 139
12.1. A BRIEF GPU HISTORY 139
12.2. PROGRAMMABLE SHADERS 140
12.3. STREAM PROCESSING MODEL 141
12.4. RESOURCE UTILIZATION AND LATENCY HIDING 142
12.5. COMMON IMPLEMENTATION CHARACTERISTICS 142
12.6. CONDITIONAL STATEMENTS AND DIVERGENT BRANCHES 142
12.7. MEMORY ARCHITECTURE 143
12.7.1. Integrated Architectures and shared memory model *143*
12.7.2. Growth of memory access speed and processing power *144*
12.8. GENERAL STRATEGIES FOR SYNERGISTIC CPU AND GPU ALGORITHMS 144
12.9. DISCUSSION 145
13. ACKNOWLEDGEMENTS 146

Report D4.1 in Context of Work Package 4 and PRESIOUS

The goal of work package 4 in project PRESIOUS is to develop semi-automatic techniques for assisting the reassembly of fragmented, partially acquired 3D objects to form complete, plausible 3D objects. It is based on two subtasks. In task 4.1, clustering[1] of sets of fragments which belong to the same input shape is considered. In that task the goal is to research how incrementally larger fragments can be assembled by complementing fragments along contact surfaces. Once the clustering process has reassembled the available input fragments as good as possible, technology developed within task 4.2 considers the completion of the partially provided shape to a plausible model. To that end, in the PRESIOUS approach the partially reconstructed model is to be matched against a repository of whole template shapes, and missing shape parts and in addition, surface details are to be transferred. The shape repository also serves as input to task 4.1, where the information about existing full shapes will be exploited to guide the clustering process. From the finished 3D model, missing parts can be extracted for use during reconstruction of the original object, potentially including physical reconstruction, e.g., using a 3D printer.

Part A of this report first surveys the state of the art in 3D object retrieval methods for structured 3D object data, where structured refers to the 3D models being represented typically by polygon meshes. Structured models typically occur at later stages in the 3D digitization and/or restoration process. Note that report D2.1 surveys search methods for unstructured 3D data, typically relating to point cloud data as occurring in earlier stages of the digitization process. Note also that despite this distinction, methods exist to convert between structured and unstructured representations, and 3D search methods can often be interchangeably applied, at least within certain restrictions. Based on this survey, appropriate search methods will be pinpointed for implementation in PRESOUS, aiming to retrieve candidate example models from a repair repository, based on querying with an incomplete intermediate object. This part also reviews object repair methods which will allow to repair both small and large object defects based on the given object alone (called simple object repair methods) as well as more complex repair methods that employ external template models, application specific assumptions and self-symmetry. Part B of this report reviews state of the art technology for assembly of 3D object fragments. It starts by addressing the general object reassembly from fragments solving strategy. It then details the ICP registration method family, segmentation methods, and fragment matching methods, focusing on pairwise and multi-piece matching, respectively. It also reviews accelerating geometric computation using GPU hardware.

[1] Clustering here refers to ’puzzling together’ of 3D fragments and includes the description of the topological relationship between them.

Part A

3D Descriptors for Structured 3D Models and 3D Repair Methods

1. DESCRIPTORS FOR RETRIEVAL OF WHOLE 3D OBJECTS

Methods for 3D object retrieval can be distinguished according to the scope of the objects they apply to. Global methods provide similarity functions to compare 3D objects as a whole. Typically, a descriptor (e.g., feature vector, histogram, graph or other abstract data structure) is extracted which characterizes properties of the whole model in a single, aggregate representation. Then, two 3D objects are compared by comparing the global descriptors with each other. This is different from local methods which are based on comparing local properties of a pair of 3D objects. Although in PRESIOUS, the retrieval problem we are looking at in WP4 is inherently local (as we expect objects to be missing substantial parts), the study of global methods is important because in many cases the proposed features and object normalization techniques can be adapted to work also in conjunction with local methods. This section covers approaches for retrieval of whole (complete) 3D objects, and in Section 2, we will address partial methods.

1.1. 3D Descriptor Extraction Pipeline

Descriptor-based 3D search approaches often implement a processing scheme such as depicted in Figure 1 (taken from [2]). Briefly, the process can be distinguished into preprocessing, object abstraction, numeric transformation, and descriptor generation. In preprocessing, objects are normalized prior to descriptor extraction, in order to help the comparability of features. Preprocessing may include translating, rotating and scaling the objects into a canonical coordinate frame, or normalizing the level of detail of the objects by smoothing, etc. (see also Figure 2). Object abstraction then selects one of several important object aspects to base the descriptor extraction on. These regularly include surface properties, views, volumes, or structures. Numeric transformation can be applied to sample and encode the identified properties, and subsequently represent them as a descriptor of a certain form, e.g., vector, histogram, or graph.

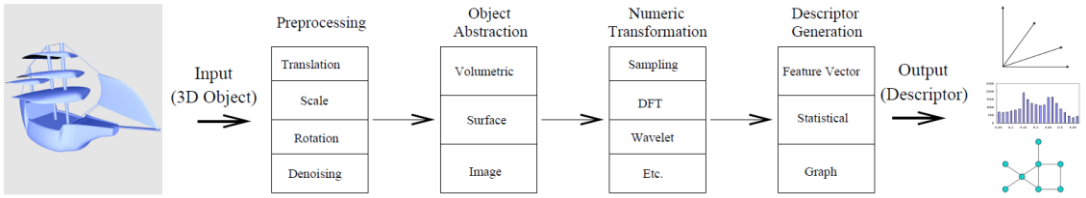


Figure 1: Typical 3D descriptor extraction pipeline (Figure taken from [2]).

A number of introductory surveys on 3D similarity search methods exist including those of Bustos et al. [3,2], Tangelder and Veltkamp [4], Iyer et al. [5], and Funkhouser et al. [6]. Important object

[2] B. Bustos, D. Keim, D. Saupe, and T. Schreck. Content-based 3D object retrieval. IEEE Computer Graphics and Applications, Special Issue on 3D Documents, 27 (4):22–27, 2007.

[3] B. Bustos, D. Keim, D. Saupe, T. Schreck, and D. Vranic. Feature-based similarity search in 3D object databases. ACM Computing Surveys, 37:345–387, 2005.

[4] Johan W. Tangelder and Remco C. Veltkamp. A survey of content based 3d shape retrieval methods. Multimedia Tools Appl., 39(3):441–471, September 2008.

[5] Natraj Iyer, Subramaniam Jayanti, Kuiyang Lou, Yagnanarayanan Kalyanaraman, and Karthik Ramani. Three-dimensional shape searching: state-of-the-art review and future trends. Comput. Aided Des., 37(5):509–530, April 2005.

processing steps relevant to 3D retrieval are also described in recent textbooks edited by Dugelay et al. [7] or Pears et al. [8]. We next review important classes of descriptors.

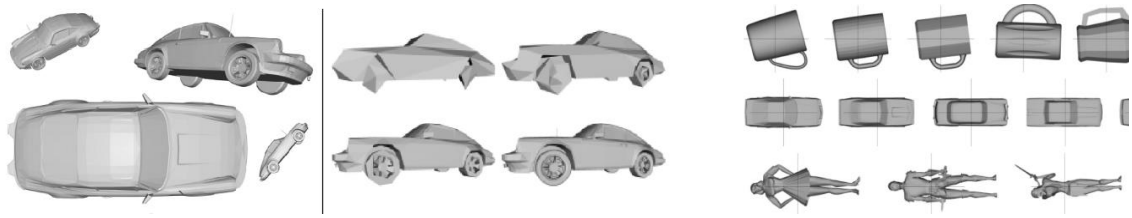


Figure 2: 3D preprocessing for feature extraction regularly includes normalization for different scale and resolution (left) and orientation (right) (Figure taken from [3]).

1.2. Image-based Descriptors

Image-based methods provide 3D description based on 2D views of objects. The general idea is to first render a set of 2D views from a 3D object, and then encode properties of the resulting images by image descriptors. Rendering methods include using flat (silhouette) images, depth images, or even perspective images including shading and texture (if available). The description of the images can potentially be done by any 2D shape descriptor e.g., based on boundary representation such as centroid distance vector, chain code, or features based on shape area such as statistical moments [9].

The first view-based approaches focused on generating a smaller number of images from each object and concatenating the image descriptors in a joint vector to represent a global 3D model. The standard Silhouette descriptor [2] uses three silhouettes rendered by orthogonal projection along the principal axes of a given 3D model. The Silhouette may be encoded by a centroid descriptor in spatial or frequency domain representation. The standard Depth Buffer descriptor [2] renders six depth maps along principal axes (two per axis) of the models, encoding the distance from the view plane to the model surface by a gray value. The 2D Fourier transform is applied to each of the depth images, and magnitudes of a larger set of low-frequency Fourier coefficients are concatenated to form the final descriptor. Figure 3 (left) illustrates.

View-based methods have shown good effectiveness for generic model retrieval, are robust with respect to model defects, and have subsequently been improved a lot in the literature. The Lightfield method [10] extended the view-based method by rendering a set of images using a system of projection planes surrounding the object (typically, 12 or 20 cameras are used). Features from both silhouettes and depth images are used to compute an aggregate distance value for each possible alignment of the camera system for a pair of objects, taking the minimum aggregate distance as the final similarity score. Figure 3 (right) illustrates.

The PANORAMA method [11] proposed to use, instead of discrete views, a single cylinder projection that renders one panoramic image of the whole model, aligned with the principal axes of the given

[6] Thomas Funkhouser, Michael Kazhdan, Patrick Min, and Philip Shilane. Shape-based retrieval and analysis of 3d models. Commun. ACM, 48(6):58–64, June 2005.

[7] Jean-Luc Dugelay, Atilla Baskurt, Mohamed Daoudi, and Edward Delp. 3D Object Processing: Compression, Indexing and Watermarking. Wiley, 2008.

[8] N. Pears, Y. Liu, and P. Bunting. 3D Imaging, Analysis and Applications. Springer, 2012.

[9] Sven Loncaric. A survey of shape analysis techniques. Pattern Recognition, 31(8):983–1001, 1998.

[10] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3d model retrieval. Comput. Graph. Forum, 22(3):223–232, 2003.

[11] Panagiotis Papadakis, Ioannis Pratikakis, Theoharis Theoharis, and Stavros Perantonis. Panorama: A 3d shape descriptor based on panoramic views for unsupervised 3d object retrieval. Int. J. Comput. Vision, 89(2-3):177–192, September 2010.

model. Using Fourier and Wavelet descriptors in combination with a custom feature vector coding resulted in improved retrieval effectiveness of this method, as compared to Lightfield and other image-based descriptors, as shown in the paper.

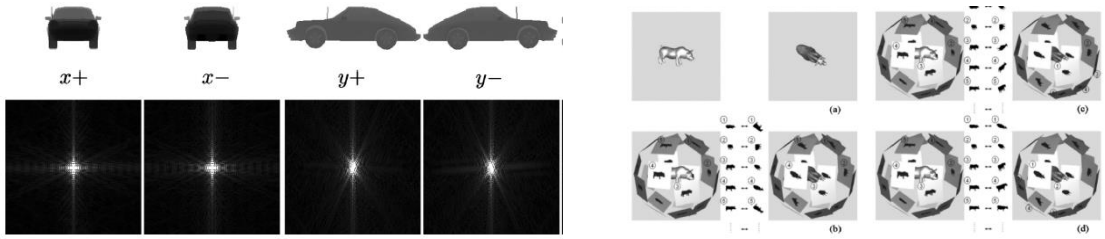


Figure 3: View-based methods: Example depth images and Fourier image representation (left, taken from [**2**]) and Light Field descriptor array alignment (right, taken from [**10**]).

More recently, additional methods based on view descriptors for 3D retrieval have been proposed. The idea of the view context method [12] is to compare two 3D views not only directly against each other based on descriptors of the given view, but also, take into account the differences of views from adjacent perspectives. Such an approach can potentially improve the descriptive power because it may compensate for variations of perspective due to differences in the object normalization stage. Furthermore, it may be advantageous to not encode an object with a number of views which is fixed a priori, but determine the number of views in a data-dependent way. To that end, the viewpoint entropy clustering approach [13] suggests to compute, for each model, the number of relevant views by an analysis function heuristically scoring the information contained in each view, and using just a set of relevant views. Yet another possible view selection function could search for those views which are most robust with respect to small variations of the viewing direction. Such approaches can potentially improve the descriptive power as they may exclude irrelevant views from the similarity computation, which would otherwise just add as noise to the comparison result.

1.3. Surface and Volume Descriptors

Another class of descriptors can be derived by analysis of object surface, typically given as point clouds or triangular meshes. Classic descriptors include Gaussian Images [14] or the Shape Spectrum descriptor [15]. The Gaussian Image descriptor is a histogram of surface normals formed over a binning of the centered bounding sphere of a given 3D object. The Shape Spectrum descriptor computes for points on the model surface the shape index, which is a function of the two principal curvatures at a given point, indicating the degree to which a basic curvature configuration is given at the point (Figure 4 top illustrates). A histogram then represents the distribution of the shape index for the whole model. While that method in one of our previous evaluations has performed rather average on generic retrieval tasks, it was well able to distinguish classes of articulated objects, such as human models in different poses, from each other [16]. Another work proposed a catalogue of distance and

[12] Bo Li and Henry Johan. Sketch-based 3d model retrieval by incorporating 2d-3d alignment. Multimedia Tools Appl., pages 363–385, 2013.

[13] Bo Li, Yijuan Lu, and Henry Johan. Sketch-based 3d model retrieval by viewpoint entropy-based adaptive view clustering. In Proc. EG Workshop on 3D Object Retrieval, pages 49–56, 2013.

[14] B. Horn. Extended gaussian images. Proceedings of the IEEE, 72(12):1671–1686, 1984.

[15] T. Zaharia and F. Preteux. 3D shape-based retrieval within the MPEG-7 framework. In Non-linear Image Processing and Pattern Analysis, volume 4304, pages 133–145, January 2001.

[16] Benjamin Bustos, Daniel A. Keim, Dietmar Saupe, Tobias Schreck, and Dejan Vranic. An Experimental Effectiveness Comparison of Methods for 3D Similarity Search. International Journal on Digital Libraries, Special Issue, 6(1):39–54, 2006.

angle-based measurements which can be sampled from the model surface, to yield a histogram descriptor [17]. Specifically, a rather simple function (the Euclidean distance between two random points) performed well in a study reported in this article regarding retrieval effectiveness. As another well-known example, the Spin Image method [18] describes a 3D model by a set of 2D histograms of local point distributions, sampled for a number of points of a model. Spin Images have been used in many applications to date and can be employed, like many other surface based descriptors, both for global and local similarity search (see also next section).

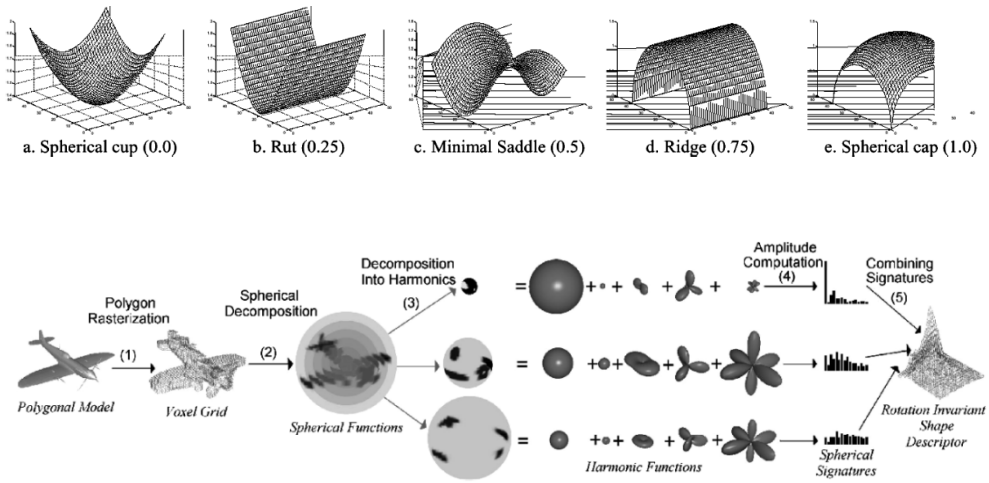


Figure 4: Shape index scores for a set of basic surface configurations (top, taken from [**15**]). The bottom image shows the decomposition of a 3D volumetric representation into a number of Spherical Harmonics functions (taken from [**19**]).

It is also possible to extract 3D description from volumetric representations. 3D objects can come either already in a volumetric representation (e.g., stemming from a solid modelling process), or a volume can be constructed from a 3D mesh as post-processing (e.g., using ray casting or distance transform approaches). A standard volume-based descriptor can be obtained by considering the voxel cell grid of a volumetric descriptor and forming a vector representation by enumerating occupation values of the grid as components in a feature vector. Occupation values can then be set e.g., in a binary way as to whether the voxel cell is intersected by the model or not [19], or in a continuous way by the fraction of overall object surface [3] intersected. An alternative is the use of the distance transform [20]. It is also possible to obtain descriptors from numeric transforms of the volume representation. Examples include the decomposition into Spherical Harmonics descriptors which provide implicit rotation invariance [19] (Figure 4 bottom illustrates); extracting gradient histograms from the volume

[17] Robert Osada, Thomas A. Funkhouser, Bernard Chazelle, and David P. Dobkin. Shape distributions. ACM Trans. Graph. , 21(4):807–832, 2002.

[18] Andrew Edie Johnson and Martial Hebert. Surface matching for object recognition in complex three-dimensional scenes. Image Vision Comput., 16(9-10):635–651, 1998.

[19] Thomas A. Funkhouser, Patrick Min, Michael M. Kazhdan, Joyce Chen, J. Alex Halderman, David P. Dobkin, and David Pokrass Jacobs. A search engine for 3d models. ACM Trans. Graph., 22(1):83–105, 2003.

[20] Maximilian Scherer, Michael Walter, and Tobias Schreck. Histograms of oriented gradients for 3D model retrieval. In Proc. Int. Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, pages 41–48. University of West Bohemia, Plzen, 2010.

[20]; or Heat Kernel Signatures (HKS) obtained from an interpolation of Laplace-Beltrami Eigenfunctions computed on the surface of the model [21].

1.4. Structure Descriptors

Briefly, structure-based descriptors analyse a 3D shape for structural properties of the model, in the sense of object components and their topological and/or spatial relationships to each other. A natural encoding for these properties are annotated graph structures. A landmark paper of Hilaga et al. [22] proposed the use of topological graphs (Reeb graphs) to encode object structure. A hierarchic Reeb graph is constructed for a selected analysis function evaluated on the surface of the model (sums of geodesic distances are used in their paper). A custom matching function allows to compare models invariant with respect to deformation. A model skeleton is computed by Sundar et al. in [23]. Based on a volumetric representation, thinning operations reduce the object to a skeletal line graph, clustered components of which are annotated by local features and input to a graph matching scheme. Also, Sfikas et al. [24] introduced a string representation of graph structures obtained from connecting object segments identified by conformal factor analysis. While these methods compute a graph from scratch for a given input model, in some cases a meaningful graph representation may already be stemming from the object modeling phase. E.g, during the CAD modeling process functional components may be combined and their relationships be captured in a scene graph. Then, usage of this knowledge for similarity computation becomes possible.

1.5. Similarity Functions, Query Modalities, and Benchmarking

Extraction of descriptors is the main prerequisite for implementing descriptor-based 3D retrieval systems, and the type of descriptors in the system is decisive to the resulting retrieval performance, in terms of relevance of the retrieved objects. But besides the descriptors, the applied similarity function plays and important role, as it maps pairs of descriptors to distance scores which in turn result in the rankings. Basic options for similarity functions include the well-known Minkowski Lp distance functions. It is defined as $L_p = (\sum_{i=1}^{n} |q_i - o_i|^p)^{\frac{1}{p}}$ and gives the distance between two vectors q and o of dimensionality n. Parameter p gives different instantiations of the distance (e.g., Manhattan (p=1), Euclidean (p =2) or Maximum (p = Inf) distance). Also more complex schemes such as Quadratic Forms or Mahalanobis distances can be used. Using a matrix of weights they allow to scale the influence of the different descriptor components according to some weighting scheme. Also, there exist a number of combination (or fusion schemes) which we can resort to merge sets of descriptors into a joint similarity measure. Early fusion approaches merge the descriptors and then compute an aggregate distance score. Late fusion (or rank aggregation methods) produce a similarity ranking first for each descriptor in a set of input descriptors, and then merge the resulting lists. The type of distance function and merging scheme chosen are tuning parameters of the system.

Querying is usually done according to the query-by-example approach, where the user provides (e.g., by marking from a list of existing objects) one query object which is taken as the query. In recent years, a number of works have focused on sketch-based approaches to 3D retrieval [25, 26, 27]. A main

[21] Raif M. Rustamov. Interpolated eigenfunctions for volumetric shape processing. The Visual Computer, 27(11):951–961, 2011.

[22] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, SIGGRAPH '01, pages 203–212, New York, NY, USA, 2001. ACM.

[23] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton-based shape matching and retrieval. In Proceedings of the Shape Modeling International 2003, SMI '03, pages 130–, IEEE Computer Society 2003.

[24] Konstantinos Sfikas, Theoharis Theoharis, and Ioannis Pratikakis. Non-rigid 3d object retrieval using topological information guided by conformal factors. The Visual Computer, 28(9):943–955, 2012.

[25] Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, and Marc Alexa. Sketch-based shape retrieval. ACM Transactions on Graphics (Proceedings SIGGRAPH), 31(4):31:1–31:10, 2012.

problem to solve in sketch-based approaches relates to the types of descriptors to employ. Because 3D object and user sketch are typically of different level of abstraction, detail and quality, one needs to find a scheme that makes both query and target objects structurally comparable. As recent work shows, using view-based gradient descriptors relying on non-photorealistic renderings can be beneficial to this kind of query approach, but these are only first solutions to tackle the problem.

Evaluation of 3D retrieval methods requires appropriate benchmark data sets to test for the quality of the answer sets that can be retrieved by alternative methods. Like in other retrieval domains, 3D retrieval benchmarks exist which allow to compare the quality of rankings on defined query classes. Well-known retrieval benchmarks include the Princeton Shape Benchmark (PSB) [28] which contains about 1800 generic 3D objects classified in a four-level hierarchy. The Purdue Engineering Shape Benchmark (ESB) [29] defines a number of equivalence classes over a database of about 900 mechanical part models representing content from the engineering domain. Also, there exists a data set of architectural building and illustration models described in [30]. The Shape Retrieval Contest (SHREC) is an established evaluation forum that defines benchmarks and challenges the 3D object retrieval community to compare their retrieval algorithms on. The contest is run yearly dating back to 2006. It provides numerous benchmarks according to the type of 3D domain (e.g., generic, or domain specific such as human faces, architecture data, CAD data, and protein models), according to data acquisition (e.g., range scan data benchmarks) or invariance properties (rigid transformation, watertight models). The benchmarks also distinguish between whole retrieval and partial retrieval. To date, there does not exist a benchmark that would be geared toward the Cultural Heritage domain.

Benchmarks can be used to tune the performance of a 3D retrieval system, if the application data corresponds to one of the existing benchmarks. If no comparable benchmark exists, it is also possible to include the user into the search optimization by means of relevance feedback [31]. The basic idea in relevance feedback is that the users distinguish relevant and irrelevant objects from an initial answer set. The retrieval system then considers this information as training data to improve the search function, e.g., by optimizing the search parameters to more closely map to the relevant objects while excluding irrelevant objects, aiming to retrieve more relevant objects on the side. Machine Learning techniques such as classification approaches [32] can be employed to this end.

1.6. Conclusion on Global Retrieval Methods

The current section could only list an illustrative number of descriptors in the typical classes of descriptors, based on views, volume, surface and structure. More methods are described in surveys listed in Section 1.1 and also, in PRESIOUS Deliverables D2.1 and part B of this report. In our opinion, many of the different descriptors can be seen as variants of a smaller number of descriptors, or special cases. The methods discussed are global but are instructive because they hint at possible

[26] B. Li, T. Schreck, B. Bustos, A. Godil, M. Alexa, T. Boubekeur, J. Chen, M. Eitz, T. Furuya, K. Hildebrand, S. Huang, H. Johan, A. Kuijper, R. Ohbuchi, R. Richter, J. Saavedra, M. Scherer, T. Yanagimachi, G. Yoon, and S. Yoon. SHREC'12 Track: Sketch-Based 3D Shape Retrieval. In Proc. EG Workshop on 3D Object Retrieval, pages 109–118. Eurographics Association, 2012.

[27] Sang Yoon, Maximilian Scherer, Tobias Schreck, and Arjan Kuijper. Sketchbased 3D model retrieval using diffusion tensor fields of suggestive contours. In Proc. ACM Multimedia, pages 193–200. ACM, 2010.

[28] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The Princeton shape benchmark. In Proc. Shape Modeling International, June 2004.

[29] Subramaniam Jayanti, Yagnanarayanan Kalyanaraman, Natraj Iyer, and Karthik Ramani. Developing an engineering shape benchmark for cad models. Computer-Aided Design, 38(9):939–953, 2006.

[30] Raoul Wessel, Ina Blümel, and Reinhard Klein. A 3d shape benchmark for retrieval and automatic classification of architectural data. In Proc. EG Workshop on 3D Object Retrieval, pages 53–56, 2009.

[31] D. Giorgi, P. Frosini, M. Spagnuolo, and B. Falcidieno. 3d relevance feedback via multilevel relevance judgements. Vis. Comput., 26(10):1321–1338, October 2010.

[32] R.O. Duda, P.E. Hart, and D.G. Stork. Pattern Classification. John Wiley & Sons, 2001.

features to extract also for local areas of interest. So it is useful to have at hand a toolbox of different methods to choose from.

Identification of the best descriptors (in terms of effectiveness of the similarity functions) can typically not be done theoretically, but relies on test data. While many methods have been evaluated and compared on a subset of available benchmarks and against a subset of the competing methods, to date there is no global comparison of the wealth of methods across different data sets. Experimentation is generally needed to identify the most appropriate descriptors and parameterization thereof to adapt to a given data set and retrieval task. Even descriptors which perform well on average over a benchmark, may fail to provide good retrieval performance on specific model classes, different levels of noise or resolution of models, different 3D representations, etc. To that end, we consider it important to have a toolbox of different descriptors available to choose from (feature selection). It is in many cases also fruitful to consider combinations of descriptors, as these can compensate the disadvantages of individual descriptors for certain model classes and in sum provide improved performance [[33]] . Also, it is difficult to provide an objective run time comparison between the methods, because run time results are often not reported in the literature, or the hardware/software environments are not comparable across different works. We observe that most of the described descriptors have acceptable run time for feature extraction. Typically, feature extraction is considered a one-time task which can be performed offline. However, in the PRESIOUS approach, due to a gradual repair process an iterative search may be needed over increasingly complete models. To that end it may be required to recompute features on the fly in this process. As this however, will only be required for a single model under consideration at a time, we do not expect this to be problematic in practice.

[33] Benjamin Bustos, Daniel A. Keim, Dietmar Saupe, Tobias Schreck, and Dejan Vranic. Automatic Selection and Combination of Descriptors for Effective 3D Similarity Search. In Proc. IEEE International Workshop on Multimedia Content-based Analysis and Retrieval, pages 514–521. IEEE Computer Society, 2004.

2. PARTIAL 3D RETRIEVAL

In the previous section, we have discussed approaches to global similarity search in 3D object databases. We will here focus on methods for partial retrieval. Solving this problem is important for PRESIOUS to retrieve repair models for completion and repair of partially reconstructed 3D models of Cultural Heritage objects. Generic approaches to solve this problem include either computation of local correspondences, or aggregating local descriptors to compute an overall measure of similarity between two objects. The former approach has the advantage to make explicit the points of correspondences and can also be used for alignment purposes which are important if the completion of a partially given model by parts from its best matching object is desired. The latter approach has the advantage to operate on a single descriptor and does not need explicit correspondence computation; however, alignment then needs to be done in a post-processing step, using e.g., Iterative Closest Points (ICP)-type methods (see also Section 8 in part B of this report). We will focus on methods for computation of correspondences and the generic process to compute these can be described as follows:

1. **Detection**. First, for each model, a set of local points of interest needs to be identified. These interest points should indicate relevant model information that is meaningful in some sense. The identification can result in discrete points, or in regions. To this end, a number of interest-point detectors are candidates for application (Figure 5 (left) illustrates). Also, segmentation methods as described in Section 9 in Part B of this report are basically applicable.
2. **Description**. Each local interest point or area is then described using a local descriptor method. A simple description is already implicitly given by the coordinate(s) of the identified points or areas. Additional description can be obtained by 3D descriptors extracted for the local areas of interest. Options include basically any 3D descriptor that can be defined. Either, global 3D descriptors are extracted for a local segment or neighborhood around the identified interest point (see Section Descriptors for Retrieval of Whole 3D Models above). Also, a number of descriptors are local in nature and will be reviewed below.
3. **Matching**. After identification of local points/areas of interest and their description, a matching is computed which establishes a set of correspondences between the set of local descriptors in two models to be considered [34]. In case of description using 3D coordinates (actually, clouds of interest points), the ICP method family can be used (see also in part B of this report in Section 8). Graph-matching approaches such as the Hungarian Method [35] can take into account also similarities between local descriptors modeled as edge weights, in computing a correspondence. In combination with appropriate local descriptions, graph matching is a flexible scheme. Based on the descriptor definition, it can weigh flexibly between the notions of similarity of the local shapes and topological/proximity constraints imposed on the matching. Figure 5 (right) illustrates a matching of local points on a 3D object).

[34] U. Castellani. 3D Shape Registration. In: 3D Imaging, Analysis and Applications (Ed. Pears and Y. Liu and P. Bunting), Springer, 2012.

[35] H. W. Kuhn and Bryn Yaw. The Hungarian method for the assignment problem. Naval Res. Logist. Quart, pages 83–97, 1955.

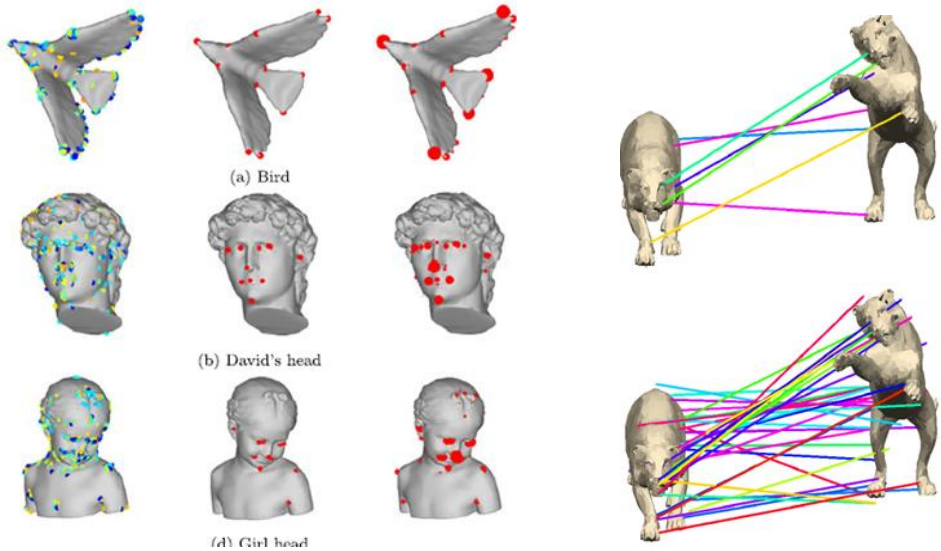


Figure 5: Detection of local 3D interest points (left, taken from [36]), and illustration of a matching of interest points (right, taken from [37]).

Appropriate implementations then need to be evaluated for performance. Again, here many different criteria are possible including, e.g., accuracy of the retrieval results, precision of the established correspondences, or run time requirements of the method. Of particular interest are two important properties: 1) The robustness (stability) of the interest point detection, and 2) the robustness (stability) of the description. 1) refers to the property that the set of identified interest points remains positional stable for a set of transformations of the models (e.g., rescaling, resampling, noise, similar versions of the same model class, etc.) 2) refers to the fact that also the local descriptors should remain stable with small changes of the local surface properties, that is, the descriptor changes should not be abrupt. These criteria have been studied experimentally for a selection of local interest point detectors and descriptors by Boyer et al. [38] as discussed in Section 2.3.1.

We next review some important candidate detectors and local descriptors. The selection is based largely on the excellent discussion of Bronstein et al. given in [39] and [38].

2.1. 3D Local Feature Detection for Partial Retrieval

Feature detectors aim to identify points or regions of interest for local feature extraction. As a most simple case, the detection is based on regular sampling which does not involve any analysis but produces a number of equally-spaced (along the surface, or along a volumetric representation) interest points. Such dense approaches are always possible but produce a large number of points, not all of them potentially meaningful but adding noise to the description, and also, can make the problem

[36] Helin Dutagaci, Chun Pan Cheung, and Afzal Godil. Evaluation of 3d interest point detection techniques via human-generated ground truth. The Visual Computer, 28(9):901–917, 2012.

[37] Jan Knopp, Mukta Prasad, and Luc J. Van Gool. Automatic shape expansion with verification to improve 3d retrieval, classification and matching. In Proc. EG Workshop on 3D Object Retrieval, pages 1–8, 2013.

[38] Edmond Boyer, Alexander M. Bronstein, Michael M. Bronstein, Benjamin Bustos, Tal Darom, Radu Horaud, Ingrid Hotz, Yosi Keller, Johannes Keustermans, Artiom Kovnatsky, Roee Litman, Jan Reininghaus, Ivan Sipiran, Dirk Smeets, Paul Suetens, Dirk Vandermeulen, Andrei Zaharescu, and Valentin Zobel. Shrec '11: Robust feature detection and description benchmark. In Proc. EG Workshop on 3D Object Retrieval, pages 71–78, 2011.

[39] A. Bronstein, M. Bronstein, and M. Ovsjanikov. 3D features, surface descriptors, and object descriptors. In: 3D Imaging, Analysis and Applications (Ed. Pears and Y. Liu and P. Bunting), Springer, 2012.

computationally more expensive. Detectors in the general case base their detection on a local analysis function and identify an interest point to fulfill some defined interestingness criterion. Parameters of the method involve the definition of the interest function, the definition of the sampling scheme, and a threshold or other decision criterion based on the analysis function. The following detectors have been particularly considered in the case for partial 3D retrieval and we briefly review them next, following the taxonomies presented in [38] and [39].

- The Harris 3D detector proposed in [40] is an adaption of the 2D Harris operator to 3D mesh models. Its analysis function is defined as the local auto correlation of the mesh obtained by considering small shifts of a considered local surface patch. A number of points showing the highest Harris value are chosen as the local interest points.
- The Difference of Gaussians-based (DOG) framework considers a scale space analysis of shape properties. The basic idea is to consider the differences of some analysis function for points on the shape, when gradually changing the scale of the shape. Points showing a local extreme of these differences are detected. In the Mesh-DOG approach [41], Gaussian kernels were used to generate the scale sequence, and curvature was suggested as the analysis function.
- Besides filtering the values of the analysis function, it is also possible to filter the underlying shape prior to computation of the analysis function. In the Mesh-SIFT approach [42], binomial filtering (as an approximation of Gaussian filtering) of the input mesh was applied and analysis functions considered were mean curvature and principal coordinates in curvature space.
- An approach filtering the shape and considering the offset of mesh vertices as the analysis function has been suggested in [43].

Besides such point detectors, also shape segmentation methods can be applied to detect regions of interest for subsequent descriptor extraction (see also Section 9 in Part B of this report).

2.2. 3D Local Description for Partial Retrieval

The identified local interest points or regions may be characterized by different descriptor types. Here we present just a brief overview over some of the recently applied methods. Additional local descriptors are also reviewed in Report D2.1 Part B.

- In [44], Oriented Gradient features are computed for a neighborhood around each detected interest point. A histogram captures the distribution of gradient orientations for similarity computation.

[40] Ivan Sipiran and Benjamin Bustos. A robust 3d interest points detector based on harris operator. In Proc. EG Workshop on 3D Object Retrieval, pages 7–14, 2010.

[41] Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu Horaud. Surface feature detection and description with applications to mesh matching. In Proc. IEEE Conference on Computer Vision and Pattern Recognition, pages 373–380, 2009.

[42] C. Maes, T. Fabry, J. Keustermans, D. Smeets, P. Suetens, and D. Vandermeulen. Feature detection on 3d face surfaces for pose normalisation and recognition. In Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on, pages 1–6, 2010.

[43] Umberto Castellani, Marco Cristani, Simone Fantoni, and Vittorio Murino. Sparse points matching by combining 3d mesh saliency with statistical descriptors. Comput. Graph. Forum, 27(2):643–652, 2008.

[44] Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu Horaud. Surface feature detection and description with applications to mesh matching. In Proc. IEEE Conference on Computer Vision and Pattern Recognition, pages 373–380, 2009.

- Spin images, originally proposed in [18] have been applied extensively in the related work and exist in different variants. They capture in a 2D histogram the spatial distribution of points in a certain neighborhood around detected points.
- Also, view-based descriptors can be applied for local description. Basically, one needs to define a view point and local area around an interest point. Then, e.g., depth images can be generated, which in turn can be used in raw form, or by another image-based feature extraction step for description [13].

2.3. Performance Assessment of Detection and Description Approaches

In the literature a wealth of detectors and local descriptors has been proposed to date, with different method classes and variants existing. The comparison of the methods by experiments is helpful to decide which one to start with in developing 3D retrieval systems. We next review two of the most encompassing experimental studies which hint at successful methods.

2.3.1. Repeatability and Description Stability

To provide stable shape retrieval under different transformations that shapes may undergo (including noise and object deformations) the detection and description of local interest points should be robust with respect to such transformations. In [38], a study was performed that systematically evaluates how stable the detection and description of 3D interest points is. The authors compiled, based on the TOSCA dataset[45] a benchmark consisting of a human shape that is transformed in eleven different ways: Isometry-preserving articulation, resampling, occlusion artifacts, introduction of small and big holes, random noise and shotnoise (offset vertices), partial and view-based representation, rescaling, and affine transformation (see Figure 6). These representative transformations are varied for strength in five different levels. The evaluation was conducted such that candidate methods were to provide a set of interest points detected on the untransformed shape, as well as all transformed shapes.

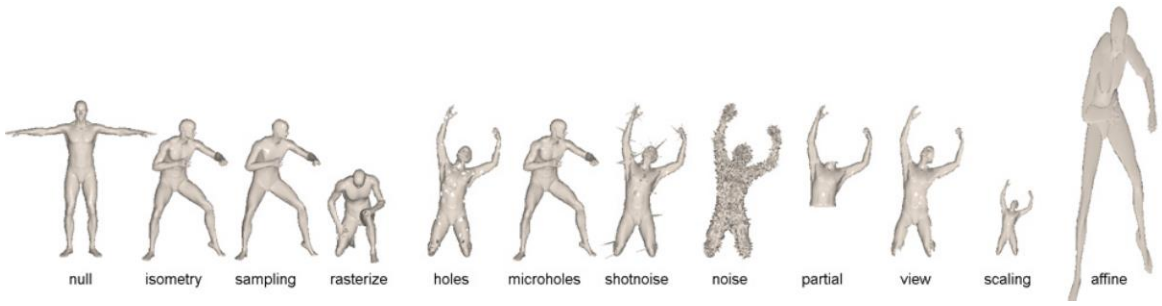


Figure 6: Transformations considered in the stability and repeatability experiment (taken from [38]).

The repeatability test compared whether the interest points detected on the transformed shapes, fall into a geodesic sphere around the interest points of the original shape. This comparison is possible as the data set provides a correspondence of surface positions across all test shapes. The more points detected in the neighborhood of the original detected points, the more repeatable a detector is. The fraction of repeatedly found interest points constitutes the repeatability score of a method. Higher repeatability is considered better as it allows detecting the same interest points in variations of the objects within a class. The descriptor stability test was performed by computing the L2 distance between the descriptors of corresponding interest points, giving a measure of how much the local descriptors change when considering approximately the same interest points across the transformed shapes. Higher stability is better, as stable descriptors do not change the resulting similarity measure abruptly, when a given detected point is somewhat offset due to a model transformation. The studied

[45] http://tosca.cs.technion.ac.il/book/resources_data.html

detectors included the HARRIS detector (identifying outlying mesh points with respect to a locally fitted quadratic surface), Mesh-DOG (identifying filtered scale space extremes measured regarding mean and Gaussian curvature), Mesh-SIFT (minimal and maximal curvature), and Mesh-Scale DOG (spatial differences maxima). The evaluation also included one region detector, Shape MSER (Maximally stable extreme regions). The considered local descriptors included Mesh-HoG (local gradient histograms), Scale invariant Spin Image, Local DepthSIFT (2D Sift features extracted from depth image constructed for tangent plane at interest point), and a Heat Kernel Signature (HKS) descriptor. The results of exhaustive tests can be summarized as:

- **Repeatability** The best methods in this experiment include the Mesh Scale-DoG (more than 90% average repeatability), and Harris 3D (around 90% average repeatability). MSER region detector performs in the middle ground (around 70-80% average repeatability), while Mesh-SIFT performs average in this study (around 50% average repeatability).
- **Stability** The Mesh-HOG approach provides the best performance (average normalized L2 distance around 0:5). The Spin Image approach provides middle ground performance (around 0:6), while the depth SIFT method scores least (at around 0:75). Results for the Heat Kernel Signature descriptor are provided only partially but indicate moderate to low performance in the experiment.

2.3.2. Quality of Interest Point Detection

Besides the repeatability and descriptor stability, the question arises as to how intuitive or semantically meaningful the detected local points are. The study of Dutagaci et al. [36] addresses this question by an experiment where manually annotated interest points on given shapes are compared to the detection results of six different detectors. The studied detectors include two Mesh Saliency detectors (scale space approach based on local curvature and vertex position), 3D Harris, 3D SIFT, Scale-dependent corner detector in a 2D embedding, and maxima of Heat Kernel Signatures. The authors collected, via a web-based interface, interest points for different 3D models from a number of users. From the manually obtained interest points, representative interest points were generated by grouping the sets of points by geodesic distance, accepting a final interest point if the group consists of a minimum number of points pairwise not more distant than a maximum threshold. The central point (in terms of geodesic distances) is selected as the representative point. Each representative interest point is weighted by the number of users assigning that point.

The evaluation against the automatic detectors follows a neighborhood-based approach, where a match is declared if the distance between automatically and interactively determined interest points is below a given threshold. The study included about 40 models annotated by about 20 users. From the manual annotation, it stands out that the number of marked interest points differs strongly between users. Interest points tend to be annotated in corners, edges, or symmetrical regions, yet some users also annotated points in planar areas without specific geometric features (with the center of the planar area as a tendency). For simple and artificial models, the annotations are more consistent, and all of them in general include extremities. Along linear features such as ridges, there is no consistent point annotation as there is no singular feature available. Concavities get less often chosen than convexities. For human and animal models, facial parts tend to be of interest. When comparing the automatic detection results to the ground truth, it is observed that when relaxing the correspondence criterion (in terms of distance between true and detected interest point), the Harris 3D and Mesh Saliency (mean curvature) improve the fastest among all methods, indicating that these methods pick up the ground truth points the best as compared to the user annotated ground truth. The corner detector picks up relevant interest points, but at a larger tolerance radius. The Voxel-based 3DSIFT Method does perform rather average in comparison. On the other hand, Mesh Saliency and corner detector generate also many irrelevant interest points, yielding a high false positive rate. The Heat Kernel Signature (HKS) method shows a different performance. It detects only few interest points, but most of them are relevant regarding the ground truth, being extreme points (which have the highest consensus among the manual users). In addition to these average results for all models, an analysis for different models was performed but did not consistently outperform any one algorithm. Further insights from the study include that the HKS method is particularly insensitive to local perturbations, as it considers global

model structure. There is no consistent relationship between the type of interest points and the detection or miss-detection by the algorithms, other than this. Most methods report interest points which are stable across different scales; the corner detector reports interest points also on individual scales.

2.4. Conclusion on Partial 3D Retrieval and Starting Points for PRESIOUS Task 4.2

In this section we have reviewed current approaches for partial 3D object retrieval. The framework includes the steps of detection of local interest points, description of local interest points, and matching. Modular workflows can be set up by choosing different instantiations for the three steps in this framework. So far, many different approaches to these steps have been proposed so the design space to choose from is quite large. It can be expected that a combination of methods appropriate to the Cultural Heritage objects considered in PRESIOUS can be found. Research is needed as so far, experimental comparison of proposed methods has been conducted only for discrete points in this design space, and evaluation has to date not considered supporting Cultural Heritage objects in particular. Judging from the state of the art, we identify the following starting points for PRESIOUS task 4.2:

- Based on the repeatability study reported in Section 2.3.1 the Mesh ScaleDoG and Harris 3D detectors provide good repeatability and are therefore candidates for local detectors.
- In the same study, the Mesh-HOG and the Spin Image approaches provided good stability of the description, which makes them also candidate descriptors to consider for our purposes. A previous study in [46] also supports that the combination of Harris detection with Spin Image description is a starting point to implement partial similarity search systems.
- According to the analysis reported in Section 2.3.2, also Heat Kernel Signature based detection has useful properties, as it yields small numbers of interest points as compared to other detectors, and is insensitive to local perturbations. It is therefore also a candidate for application.
- Regarding the similarity computation, correspondence-based approaches are promising for our purposes, as for the shape completion problem we expect to require an alignment of shapes, which is fostered by correspondence methods. Bag-of-words approaches do not provide an initial alignment. However they could be suited as a fast filtering step to generate a small number of candidate objects from a larger repository, for which alignment is then computed by a post-processing operation using e.g., an ICP approach. We want to consider Bag-of-Words representations a secondary option because of this reason.
- As different detectors and descriptors are applicable at various levels to different data, also combinations should be considered. Detectors can be combined by a voting scheme. Descriptors can be combined by e.g., early fusion approaches. It should also be considered to use combined global and local similarity computation approaches to try to arrive at stable results. Appropriate combination approaches exist [47,48] and are candidates for improved results.

The following evaluation criteria can be thought of to assess devised solutions:

[46] I. Sipiran, R. Meruane, Benjamin Bustos, Tobias Schreck, H. Johan, B Li, and Y. Lu. SHREC13 track: Large-scale partial shape retrieval using simulated range images. In Proc. EG Workshop on 3D Object Retrieval, 2013.

[47] B. Bustos, T. Schreck, M. Walter, J. Barrios, M. Schaefer, and D. Keim. Improving 3D similarity search by enhancing and combining 3D descriptors. Springer Multimedia Tools and Applications, 58(1):81–108, 2012.

[48] Tobias Schreck, Maximilian Scherer, Michael Walter, Benjamin Bustos, Sang Yoon, and Arjan Kuijper. Graph-based combinations of fragment descriptors for improved 3D Object Retrieval. In Proc. ACM Multimedia Systems Conference, pages 23–28, 2012.

- Established ground-truth-based evaluation methods are possible; however they require the definition of a benchmark data set to provide the class labels. Existing benchmarks currently do not support Cultural Heritage models. Manual annotation of scan data from PRESIOUS could be an approach to evaluate the search methods in terms of precision and recall.
- The retrieval in our case is complicated by the existence of cracks and broken surfaces for the search model. These artifacts can potentially misguide the interest point detection and description. One could, in an experiment, vary the fraction of cracks and broken surfaces and observe the change to the interest point detection and quantitatively compare resulting retrieval precision rates.
- If available, texture information should also be considered for the retrieval. To that end, experiments could compare how much, in terms of retrieval precision, can be gained by adding texture information to the shape-based partial retrieval. To date, only limited studies in the area of shape and texture based retrieval exist. One of these experiments includes [49] and a similar evaluation methodology could be followed.

[49] Andrea Cerri, Silvia Biasotti, Mostafa Abdelrahman, Jesús Angulo, K. Berger, Louis Chevallier, Moumen T. El-Melegy, Aly A. Farag, F. Lefebvre, Andrea Giachetti, H. Guermoud, Y.-J. Liu, Santiago Velasco-Forero, Jean-Ronan Vigouroux, C.-X. Xu, and J.-B. Zhang. Shrec'13 track: Retrieval on textured 3d models. In Proc. EG Workshop on 3D Object Retrieval, pages 73–80, 2013.

3. 3D MESH DEFECTS AND FLAWS

3.1. Introduction and Structure of the Problem

The following sections provide an overview of the current state of the art in repairing various aspects of 3D mesh models. We primarily focus on triangular meshes as they are the most prevalent from of representation for 3D models. In recent years dedicated hardware acceleration for visualization of triangular meshes has become almost ubiquitous. This also induced that for all other representations (such as point-clouds, volumetric or implicit representations) there are well-known algorithms to convert them to triangular meshes. While this is often also true for the inverse direction, it is however not always possible. In fact many algorithms presented in this section apply such conversions of representation to exploit particular properties of the different data models for efficient computation. Hence the possibilities of conversion many of the mentioned algorithms are also applicable to other forms of 3D model representation.

We structure the problem as follows. In the remainder of this section, we classify and illustrate defects that can commonly be encountered when working with mesh models that were acquired by digitizing real world cultural heritage objects. We also take into account defects that regularly result out of numerical instability during processing or conversions of the representation. In general, the notion of defect or flaw is used to denote a certain frailty or shortcoming of an object that prevents it from having certain qualities that are precondition for utilizing the object for a certain purpose. In the context of the digitization of cultural heritage objects, we distinguish between defects that are already inherent to a physical cultural heritage object (see Section 3.2) and defects that are introduced during acquisition, conversion or processing of its digital representation (i.e. a set of 3D Meshes with texture information) and that are thus not present in the real object (see Sections 3.3 - 3.5).

Based on the survey of defects, in Section 4 we then introduce a survey on basic mesh repair algorithms. Basic mesh repair algorithms are characterized in that they often address one or at most only a few types of defects. In general they avoid relying on domain or application specific assumptions and are more geared towards broad generic applicability than exploiting specific knowledge about a particular use case to generate a high quality result with maximum efficiency. For many applications they might reintroduce other defects and thus produce suboptimal results. However, they often can be adapted for many applications by tweaking their input parameters. Basic Shape repair algorithms are generally unaware of any semantic concepts that go beyond the mathematical notion of a manifold mesh itself (since these would be specific to the application domain). Almost all mentioned algorithms work solely on the geometric and connectivity information that is directly encoded by the mesh. Due to their often straight-forward workflow and the well-defined mathematical properties of a mesh, it is relatively easy to determine metrics to evaluate and compare the effectiveness of Basic Shape Repair algorithms. Their generic applicability and well-known characteristics lead to their use as building-blocks of more complex mesh processing workflows. Note that this section is largely based on a very recent survey by Attene et al. [50].

After considering basic repair, we discuss complex mesh repair algorithms in Section 5. They are characterized in that they focus to fit the requirements of certain application and thus often rely on domain specific assumptions and semantics either directly or indirectly by employing facilities to interactively gather input from users (that are likely domain experts). Since certain fields of application encompass numerous classes of defects, their workflow often also includes basic mesh repair algorithms. Besides the aforementioned explicit use of domain knowledge and the gathering of user input they often compare the input mesh to a repository of other models to augment the information that is present in the input mesh in order to yield a plausible result. For this they often rely on computing content or meta-data based similarity measures or might as well let the user provide additional cues to determine the relevance or plausibility of possible solutions. The evaluation and comparison of complex mesh repair approaches is non-trivial due to workflow complexity, the

[50] Marco Attene, Marcel Campen, and Leif Kobbelt. Polygon mesh repairing. ACM Computing Surveys, 45(2):1–33, February 2013.

involvement of the user and external information (through a repository of additional mesh models), as well as the gearing towards the specifics of a particular application.

Finally, in Section 6 we focus our conclusion on the applicability of the previously mentioned algorithms to the PRESIOUS use case and develop a possible workflow for the repair of fragmented cultural heritage objects. In addition we also observe a lack of evaluation for many of the mentioned approaches. Hence we derive certain key aspects that can constitute a more solid foundation for evaluating algorithms developed for object repair in PRESIOUS.

3.2. Defects of the Originating Physical Object

Given that the original cultural heritage object once served a certain purpose (e.g., as an article of daily use such as a jar, as a functional part of a device or building or as a cultural artifact symbolizing certain aspects of the culture of its time) it might have lost some of this qualities in the time between its creation and recovery. During digitization, this loss of quality is mapped to the obtained digital representation. An improvement in sensor precision or acquisition methodology cannot remove the presence of the defect but is inclined to accentuate it even more.

In the following, we are limiting our inquiry to a coarse classification of defects of solid cultural heritage objects that are made of robust materials. The classification is conducted according to the appearance of the defects in the digitized model. For more details on erosion please see deliverable report D3.1.

Small-Scale Decay

Small-scale decay involves defects which are typically very small as compared to the overall object scale. Depending on the nature of the decay source, the effects can occur only in discrete locations, or they can be distributed over the entire model, with varying densities. Examples of such processes are chemical weathering and small scale mechanical abrasion over long time periods. These processes can have strong impact on small scale features in the model such as the loss or introduction of small, high-frequency geometric details as well as color and patterns of texture.

As these defects are densely collocated and thus in total affect large areas of the object surface with continuously varying intensity, it is difficult to reconstruct or estimate the original surface structure in the digital model by relying solely on information from surrounding areas of the model surface. Instead a repository of similar un-weathered models, specific knowledge about the type of the digitized object or a detailed scientific model of the involved decay processes is required to estimate the initial appearance of the object.

Missing of Small Fragments

This category encompasses the missing of smaller fragments of the model such as chipped off slivers, fissures and larger scratches which reduce the volume of the object. These effects could be caused by physical weathering or other mechanical stress that was inflicted on the object before its digitization. In comparison to artifacts that are caused by defects of the previous category, these defects are usually less in number but cause a stronger effect on the surface geometry. They are likely to introduce new geometric features in previously regular surface areas and, depending on the material composition of the physical object, may change the affected areas' texture completely.

Due to their size and more sparse distribution on the surface these defects can be separated more easily from their surroundings. Their erratic intensity of effect on the surface and texture often leads to the circumstance that the initial appearance of the object can be estimated by using information from its immediate surroundings (i.e. 3D inpainting).

Residues on the Surface

Residues increase the volume of the object and can have a varying intensity which especially affects the texture of the object as well as small-scale geometric features of the surface. In addition the area of effect can be of larger variance. Depending on the composition of the material and erosive decay of the residues themselves new geometric and textural features can also be introduced. However,

practically, residues can often be removed before digitization. If this is not possible these defects appear similar to small-scale decay or chipped-off fragments in the digitized model. The original appearance of the object could be estimated by inpainting that relies on information obtained from adjacent areas, however if the fraction of affected surface area gets larger it might be difficult to judge which parts of the model surface reflect the actual geometric and textural features of the object and which parts are obfuscated by residues in the digitized model.

Missing of Large Parts

This category captures defects which lead to missing of large scale fragments important to the overall semantics of the digitized model. These defects can arise due to mechanical stress or as a long-term effect of physical and chemical weathering as well. Broken-off parts of the object could then be digitized separately and could thus to a certain extend be aligned and reassembled based on the geometric and/or textural details of their breaking edges. Or, in a second variant, some parts of the model are completely missing. Effectively there is no explicit information on the actual shape and texture of a missing fragment. As the parts are large and also capture a part of the semantic meaning of the original object it is very difficult to estimate its shape when the exploitable knowledge is restricted to the remaining digitized fragments. Missing larger parts could look alike to missing small scale fragments or residues in the digital representation, which further complicates estimation of the complete shape and its classification. For example a mug missing its handle could resemble a vase. A fragment of an embrasure could be misclassified as a decoration of a pillar. A statue with missing arm could have either depicted a person performing a gesture or holding an item of special symbolic meaning or not. A statue missing its head might not even be classified as being incomplete during automated processing if no additional knowledge about the concept of a statue is present. On the other hand, even for a domain expert it might be difficult to judge whether the statue actually depicted a historical person or not. This kind of information is often lost entirely but in many cases domain specific knowledge and assumptions could lead to at least a more complete estimation of the original appearance. In context of automated, machine-based processing of mesh models a repository of other models could be used in order to identify similar, but more complete models. Certain geometric and textural information from similar models could then be transformed to fit the breaking edges of an incomplete object. Albeit it seems to be a challenging if not impossible task to develop a standardized generic but also semantically meaningful measurement to support the evaluation of such approaches.

3.3. Local Connectivity Defects

Most formats that are commonly used for sharing 3D models encode surface meshes through indexed face sets. For efficiency and flexibility reasons the formats distribute mesh data over several blocks. Vertex coordinates and polygon topology are separated and cross-referenced by index. This structure itself does not enforce most constraints that must be met to conform to the mathematical notion of a piecewise linear 2-manifold in R3 that consists of homogeneous simplicial 2-complexes [51]. However this is a prerequisite for the applicability of the majority of mesh processing algorithms [50,52,53].

Local connectivity defects are the most simple family of defects that violate the above mentioned notion. In general each defect instance only involves very few elements of a mesh. These defects can be part of more complex defects that are mentioned in subsequent sections. According to [50,52] the following local connectivity defects can be distinguished:

Isolated Vertices

[51] I.e. sets of two dimensional simplices (in this case: sets of triangles).

[52] Marcel Campen, Marco Attene, and Leif Kobbelt. A Practical Guide to Polygon Mesh Repairing. Eurographics State of the Art Report, 2012.

[53] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. Polygon Mesh Processing. Ak Peters Ltd., Natick, Massachusetts, 2010.

Isolated vertices are not part of an edge of any triangle (or polygon) in the mesh. That is they are not an edge of any triangle (or polygon) in the mesh. Pragmatically isolated vertices are the vertices of a 3D model file that are not referenced by any facet or edge (depending on the file format). In most cases isolated vertices can be safely ignored or removed from the mesh.

Dangling Edges

Some file formats, e.g., OFF [54] and PLY [55] allow for explicit encoding of edges [56]. This can lead to dangling edges which are not an element of any facet of the mesh. In most cases dangling edges can be safely ignored or removed from the mesh. Alternatively their orientation might provide useful information for more complex repair algorithms. However the implementation of such algorithms is specific to just a fraction of model file formats which further lowers the potential benefit of such implementations for generic applications.

Singular Edges

These are edges that are element of more than two facets. As the previous two defect categories, singular edges violate the manifoldness of a mesh. Thus they are also referred to as complex or non-manifold edges and often result out of mergers of multiple 3D models. While singular edges are relatively easy to detect, their repair is not as trivial and does introduce entirely new primitives to the mesh. Some repair algorithms to address singular edges are mentioned in Section 4.1.

Singular Vertices

These are single vertices that violate the combinatorial manifoldness of the mesh; they might also be referred to as complex or non-manifold vertices. However a singular vertex is more difficult to detect than a singular edge as it is not sufficient to count the number of adjacent facets. Instead the adjacency of facets in the neighborhood of the vertex has to be considered. Informally there is no path [57] that contains exactly the adjacent facets of the vertex. Singular vertices might be caused by merging several meshes, topological noise (see below) or it might be a side effect of other mesh processing algorithms. Repair algorithms to address singular vertices are mentioned in Section 4.1.

Figure 7 illustrates a number of common defect types.

[54] Object File Format, see http://shape.cs.princeton.edu/benchmark/documentation/off_format.html

[55] Polygon File Format, see http://paulbourke.net/dataformats/ply/

[56] In contrast to implicit encoding where edges are derived from facets.

[57] In this case, a path denotes an ordered set of facets, where subsequent facets must be adjacent to each other by sharing an edge, note that each facet can only be contained at most once in the ordered set.

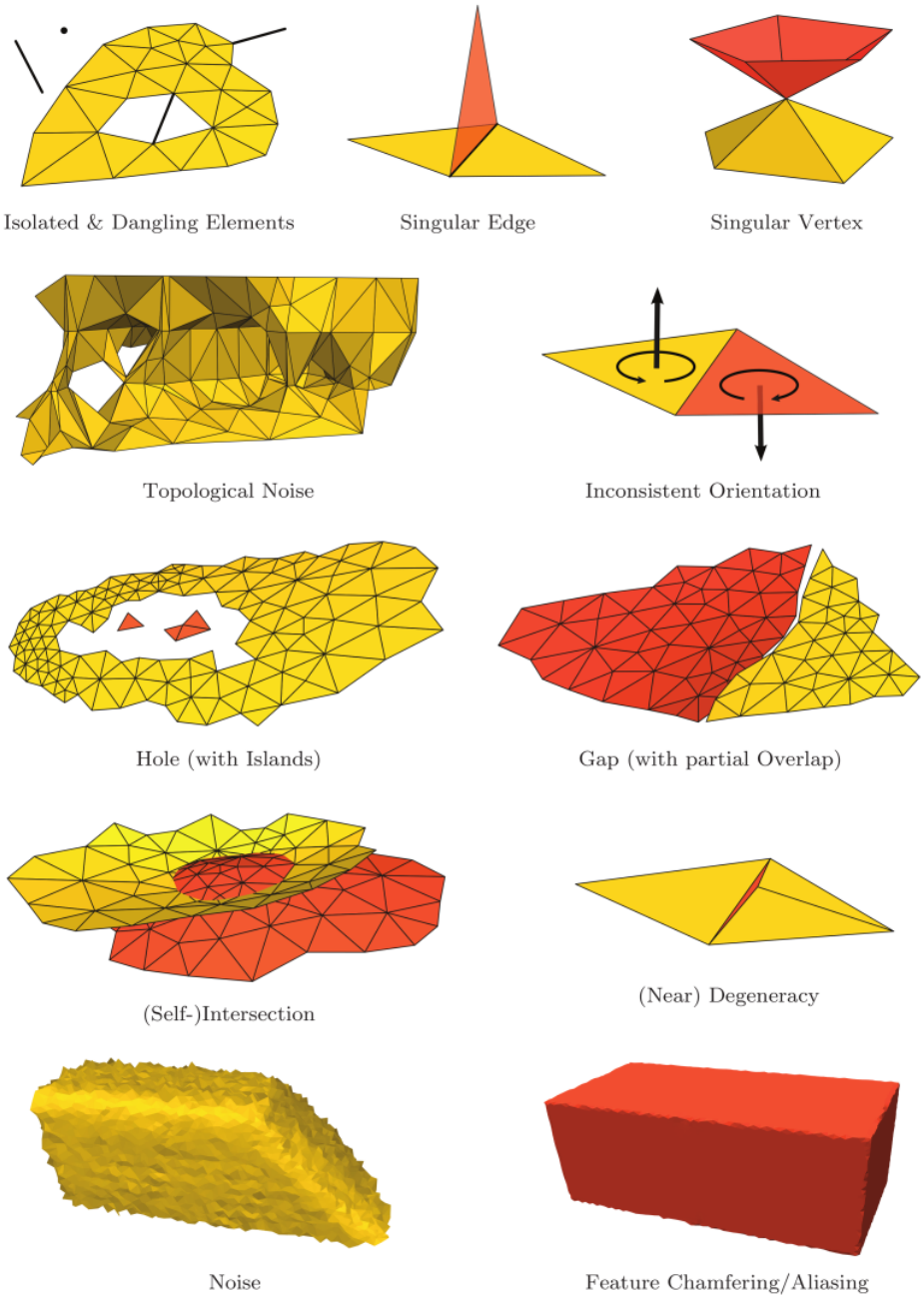


Figure 7: Common defect types of 3D Mesh Models (Figure taken from [50]).

3.4. Global Topology

Besides the family of local connectivity defects, where single mesh elements can break the manifoldness of the entire mesh, the family of global topology defects does not strictly violate the manifoldness of a mesh, are harder to detect and are caused by the combination of certain properties of several mesh elements. Most frequently they are introduced by conversion between different model representations and specifics of the model acquisition methodology when obtained by 3D scanners or cameras.

Topological Noise

Topological noise is commonly encountered when the model has been reconstructed from point clouds or depth images. Due to aliasing effects, tiny handles or tunnels are introduced that were not present in

the physical object which leads to an increase of the genus of the object. In [50] it is reported that a scan obtained from a Buddha [58] sculpture happens to have 104 handles while the originating object had only 6. Topological noise not only introduces qualitative topology issues but could complicate subsequent processing. Repair algorithms to address topological noise can be found in Section 4.1.

Orientation

In most rendering systems the orientation of a facet is used to improve the performance of the rendering system. If the angle between the viewpoint direction and the normal vector (i.e. the orientation of a facet) gets large enough, it can be skipped at rasterization (also known as Backface Culling). In most 3D model formats however, the orientation of a facet is often encoded implicitly through the ordering of its bounding vertices. While different rendering systems use opposite orientation [59] for implicitly coded normals, it is relatively easy to uniformly convert the orientation of all facets before using them in the rendering pipeline. However, problems do arise if the ordering of bounding edges is not consistent in the model itself.

3.5. Geometric Defects

As with global topology issues, geometric defects do not necessarily violate the manifoldness of the mesh. Defects of this category are caused by the position of a group of vertices and are thus difficult to detect.

Surface Holes and Gaps

When physical objects are digitized through 3D scanners usually some parts of the object remain occluded behind other parts of the object or the surrounding scene. Hence there is no explicit information on certain parts of the object surface. Depending on how the mesh model is created out of the sensor data (e.g., point clouds), areas surrounding those occluded areas do not get connected across the occluded area and thus form holes or gaps in the mesh. Informally, gaps are found between two isolated parts of the mesh whereas holes are surrounded by a single mesh fragment. A special variant of these occlusion induced defects are so called islands that are small isolated mesh fragments located within holes. Another way to introduce holes and gaps to meshes can be the merger of several models into a new model which is often the case in CAD related applications. To address these defects, a second step is required. However, the applicability of the different approaches greatly depends on the characteristics of the physical object, quality and origin of the data as well as the application domain. In some cases it is required to gather user input to distinguish between holes and gaps that were present in the originating object and those that were introduced during digitization. Depending on the extent of occlusion defects of this category can have an semantic impact that is similar to the missing of small to large parts of the physical object as discussed in Section 3.2. However occlusion defects can be detected more easily in most cases. Hole filling and gap closing algorithms are discussed in Sections 4.1 and 4.2.

Degenerate Elements

In the context of triangular meshes, degenerate triangles have an area of zero. In a wider sense, triangles that have a very large perimeter in proportion to their area could also be referred to as degenerate. Degenerate Elements cause numerical instabilities for many applications since a lot of properties of the facet cannot be computed (e.g., normal vectors, circumscribing circles, barycentric coordinates). For triangles with near zero area and relatively large perimeter the relative error of the computed values increases drastically [60]. Approaches that repair degenerate elements are discussed in Section 4.1.

[58] Provided in the Stanford 3D scan repository at http://graphics.stanford.edu/data/3Dscanrep/

[59] OpenGL uses counter-clockwise orientation whereas Direct3D uses clockwise orientation.

[60] In many performance critical applications single precision floating point arithmetic is used which furthermore increase the relative error by several orders of magnitude.

Self-Intersections

Self-intersections violate the geometrical manifoldness of a mesh. Self-intersections could result out of the merger of multiple 3D models or after the registration of several sensor shots of the same scene from different viewpoints. While self-intersections are relatively simple to detect there is no generic solution for repairing them with adequate quality. However several approaches to repair self-intersections are mentioned in Section 4.1.

Sharp Feature Chamfering

Depending on the sensor and methodology that is used during digitization of a physical object, samples of the object surface are often located on a specific line, curve or grid whose position and granularity are fixed. Often those patterns cannot be adjusted to exactly match the location of features of the physical object. Thus many corners and edges of the object are not sampled with high accuracy. This especially affects features of very high spatial frequency and amplitude and results in aliasing effects. Regular patterns on the originating object with a frequency that is slightly shifted from the frequency of the sampling pattern (or multiples and to some extent fractions thereof [61]) could appear to be completely irregular on the resulting model. However it is to some extent possible to repair such defects as mentioned in Section 4.1.

Data Noise

This type of noise is similar to Sharp Feature Chamfering in that it results out of the limited accuracy of the sensor used for digitization. To a lesser extent it could also result due to numerical instability during format conversion or mesh-processing. Each vertex of the mesh is slightly moved from the actual position of the sampled surface point while the amount of the shift varies individually for each vertex [62]. Practically, this leads to the introduction of high-frequency low amplitude features on the surface that can obfuscate other high-frequency features of the object. The difficulty in removing data noise is to retain the sharpness of high-frequency features, which is often only possible to a limited degree. In general it is easier to apply denoising to the mesh instead of the initial point-cloud as the implicit connectivity information can be exploited for algorithms such as Laplacian smoothing and bilateral denoising (see Section 4.1).

[61] Informally, if the feature frequency and sampling frequency have a relatively small greatest common divisor. See also the Nyquist-Shannon sampling theorem.

[62] Depending on the digitization methodology the direction could in some cases be strongly correlated for adjacent sampling points.

4. BASIC SHAPE REPAIR

After the discussion of typical model defects and flaws, we start the discussion of basic shape repair methods in this section by generic methods, most of which can be applied automatically without requiring additional user input at run time. The methods are targeted at defects which are of relatively small size, considering the overall given model on which the defect occurs. Example issues include holes, singular vertices, gaps and small overlaps, large overlaps, handles, and inconsistent orientation. Furthermore, they include complex edges, self-intersection, erosive decay of physical object, minor parts chipped off from physical object, and major parts broken-off from physical object. We first discuss local approaches for basic shape repair, followed by global approaches. Again note that this part is based largely on the recent survey article of Attene et al. [50].

4.1. Local Approaches

Local approaches operate locally on single defects; hence they are suitable when defects are sparsely distributed across the surface. In a first step defects are located on the surface and in a second step processed individually in isolation from each other. Local approaches are prone to introduce new defects of different nature since they have a limited scope when processing mesh data. When defects are located very closely or different defect types are collocated they often fail to achieve satisfactory results. That is they do not repair the targeted defect or introduce new defects of varying types. However, in comparison with global approaches they are often more easy to implement and can deliver results of better quality [63] if the mentioned preconditions are met.

Repairing Local Connectivity Defects

Meshes with local connectivity defects can be categorized depending on whether they circumscribe a so called regular set [50]. In other words, these meshes clearly define watertight volumes. If this is the case several algorithms exist that can convert such meshes to manifold meshes. According to [50] two algorithms described in [Guéziec et al. 2001] [64] and [Rossignac and Cardoze 1999] convert such non-manifold meshes into sets of manifold meshes that describe bound regular sets. While a closed surface mesh can represent a regular set only if it has no self-intersections, these two algorithms focus on the connectivity only and are still applicable. However the result is not a manifold mesh since it still has self-intersections. In both algorithms each singular edge that is incident to a number of 2k facets is split into k manifold edges that have 2 incident faces each. This potentially produces a number of additional problems. First, this is likely to produce new singular vertices and second could introduce new self-intersections [65]. Thus to address the former problem [Rossignac and Cardoze 1999] suggest a strategy to reduce the number of edge duplications while [Guéziec et al. 2001] introduces an additional operation that joins the boundary edges of the mesh along that was cut around the initial singular edge itself either by contracting the boundary loops (pinching) or by stitching nearly boundary curves effectively reducing the number of connected components (snapping).

There are extensions that work for higher dimensions and produce a so called initial quasi-manifold which is a slightly weaker condition than manifoldness [De Floriani et al. 2003]. [Attene et al. 2009] further proposes two algorithms for tetrahedral meshes. With the exception of the second variant in [Attene et al. 2009] all algorithms do not address the introduction of self-intersection through duplicated edges [66]. Table 1 summarizes the methods.

[63] I.e. less information of the initial surface mesh is lost.

[64] References in square brackets containing author names are taken from the survey of Attene et al. For the sake of conciseness and clarity we are listing them in block in the appendix of part A of this report.

[65] The duplicated edges do intersect since they have the same coordinates.

[66] Since in these sources a distinction is made between *combinatorial* and *geometric manifoldness* whereas the former is not violated by self-intersections, that in this case results out of quasi identical, duplicated edges that are adjacent to different facets.

algorithm	fixed defects	input constraints	optimal
[Rossignac and Cardoze 1999]	combinatorial singularities	no boundary	x
[Guéziec et al. 2001]	combinatorial singularities	no boundary	
[Attene et al. 2009] (combinatorial)	combinatorial singularities	no boundary	
[Attene et al. 2009] (geometric)	geometric and combinatorial singularities	no boundary, degenerate elements and self intersections	

Table 1: Algorithms to repair local connectivity defects, taken from [50]: Proposed algorithms for removal of singularities and their input constraints, while the proposal of [Rossignac and Cardoze 1999] computes an optimal solution, the output of the second algorithm from [Attene et al. 2009] is guaranteed to never contain geometric singularities (e.g., self-intersections) but imposes more requirements on its input.

Gap Closing

Gaps are encountered between two mesh boundaries that are close but not connected to each other. Usually these two boundaries are located on components that are also completely separated from each other i.e. they are not part of the same mesh segment. The boundary of a gap is formed by multiple disconnected chains of edges. Since these mesh boundaries have a very small Euclidean distance to each other, gaps can be detected by simply matching them by distance.

Thus, the simplest algorithm for gap closing is proposed in [Rock and Wozny 1992] where vertices within a configurable distance threshold are merged automatically. This is especially suitable for gaps that occur due to numerical instabilities and the merger of multiple models which could result within CAD applications or the registration of several surface models that were extracted from the same physical object but different perspective.

Other approaches proceed systematically along the boundary edges found in the input mesh to have a better control over the introduced topology changes. [Sheng and Meier 1995] as well as [Barequet and Kumar 1997] proceed on a per-edge basis and progressively merge them starting from those with the smallest distance among them. In [Turk and Levoy 1994] the authors also handle the special case of so called negative gaps which are boundaries at overlapping or intersecting but combinatorially disconnected mesh-patches and process them before boundary edges with lower distance. The pairwise processing of edges in these approaches avoids the introduction of singular edges. However this pairwise merging of boundary edges which is also referred to as zippering can leave certain gaps unclosed. To address this [Borodin et al. 2002] allows the merging of more than two edges. In a subsequent step, these singular edges are removed by edge splitting similar to the approaches for repairing local connectivity defects mentioned above. Thus this process adapts the mesh resolution which is especially useful if the disconnected mesh boundaries are composed of edges with greatly varying sizes where simply merging edges of very different length could lead to distortions, near degenerate faces or simply parts of a boundary edge chain not being repaired [67] or connected to very distant edges of the opposite boundary edge chain. This approach is also known as stitching. In extension to this [Patel et al. 2005] propose using a threshold to choose between these two approaches. If edge distances are below the threshold, they are simply merged through zippering and above threshold gaps are closed by stitching in additional facets through edge splitting and shifting.

All of the mentioned approaches are greedy algorithms that start at the boundaries with the least distance or most overlap. In contrast [Barequet and Sharir 1995] propose to establish globally consistent boundary correspondences first by heuristically matching the curvature of partial samplings

[67] This could also lead to the introduction of new holes.

of the (previously detected) boundary edge chains. However these algorithms only work for gaps between two boundary edge chains. Also all have problems dealing with large overlaps even if the gap is adjacent to two opposing boundary edge chains. More reliable algorithms can be found in Section 4.2, and Table 2 summarizes.

algorithm	input constraints	required parameters	possible new defects
[Rock and Wozny 1992]	very small gaps	max. gap width	self intersections, degeneracies, singularities
[Sheng and Meier 1995]	-	max. gap width	self intersections, degeneracies
[Barequet and Kumar 1997]	-	max. gap width	self intersections, degeneracies
[Turk and Levoy 1994]	overlapping parts (negative gaps)	gap threshold	self intersections, degeneracies
[Borodin et al. 2002]	-	-	self intersections, degeneracies
[Patel et al. 2005]	-	-	self intersections, degeneracies
[Barequet and Sharir 1995]	-	gap threshold	self intersections, degeneracies
[Bischoff and Kobbelt 2005]	-	gap width, resolution	degeneracies

Table 2: Gap closing algorithms, taken from [50]: All proposed algorithms guarantee gap-free output only if boundary edge distances are below a certain threshold. However using them with too high thresholds can yield arbitrary and unexpected results.

Hole Filling

In contrast to gaps, that might also result out of inaccurate merging or registration or multiple partial models, holes almost always correspond to missing surface parts. Thus, it is in general advisable to close them by stitching in new triangles (or polygons). Early algorithms simply search for closed loops of boundary edge chains. Detected holes are then filled in by triangulation of the boundary loop polygons. While [Bøhn and Wozny 1992], [Mäkelä and Dolenc 1993], [Varnuska et al. 2005] apply greedy strategies according to simple heuristics [Roth and Wibowoo 1997] use a more elaborate randomized triangulation method. However this method requires that the (previously detected) hole boundary is transformed to a parametrized representation over a plane that is itself fitted to the hole boundary. [Brunton et al. 2010] propose a simulated annealing-based boundary "unfolding" to increase the probability, that such a planar parametrization can be established for more complex boundary geometries with plausible results. [Pfeifle and Seidel 1996] heuristically introduce additional vertices to establish a Delaunay-like triangulation.

All of these methods do not pay attention to the geometric quality of the inserted surface, which often results in very unintuitive solutions for complex hole boundaries [68]. [Barequet and Sharir 1995] address this by trying to find a solution with minimum surface area. [Liepa 2003] further refines the approach by introducing so called discontinuity penalties along the hole boundaries and applying mesh fairing techniques to the constructed solutions. Subsequently [Bac et al. 2008] improved the computational efficiency of the approach by performing the filling and fairing steps interleaved in a

[68] I.e. besides self-intersecting solutions, huge gaps in curvature, vertex density and triangle shape can be observed at the boundary between the solution and the input mesh.

multilevel procedure. [Wei et al. 2010] [69] generalize this approach by additionally taking into account internal angles, dihedral angles and triangle areas. However these approaches employ a dynamic programming technique that albeit all optimizations is very complex and time consuming for holes with a large number of boundary edges such as high-resolution scans.

[Zhao et al. 2007] use an advancing front mesh generation method which iteratively derives desirable triangle normals from the triangles adjacent to the boundary edge chain to insert matching triangles at the boundary edges, thus reducing the size of the remaining hole with each iteration. In a final step, the resulting patch is smoothed by updating the vertex positions according to a Poisson equation.

Several other approaches were proposed that aim to compute intuitive solutions. [Branch et al. 2006] fit b-splines to detected boundary edge chains and use a threshold for their average variance of torsion to distinguish between holes that result out of occlusion during digitization and "real" holes of the object. For hole filling, a set of radial basis functions is computed for points located closely to the hole and then interpolated to compute the vertex positions of the solution patch. According to [50] other techniques that have been applied to derive intuitive geometry are NURBS fitting [Kumar et al. 2007], curvature energy minimization [Lévy 2003; Pernot et al. 2006] and Moving Least Squares projection [Wang and Oliveira 2007; Tekumalla and Cohen 2004]. The latter introduces a check that tests each newly computed triangle for intersection with the current mesh. Thus this approach is guaranteed to be free of introduction of new self-intersections. However it might not find a solution for certain input characteristics. [Wagner et al. 2003] applied a randomized optimization technique to remove triangles from the solution in order to break such deadlocks. Though the convergence of the approach cannot be guaranteed. All of the mentioned algorithms so far process each identified hole boundary in isolation, thus effectively ignoring additional cues that could be derived from the input mesh. For example, although islands provide precise information on the geometry of a hole-filling patch they are completely ignored. In consequence the process introduces a new, hard to detect gap between the solution patch and the island boundary and might also introduce self-intersection between them. [Podolak and Rusinkiewicz 2005] avoid these issues by deriving a tetrahedral space-partitioning [70] from the input mesh where tetrahedron facets are aligned to the mesh polygons so that they are either completely inside or outside the model. The partition scheme is computed adaptively in an iterative manner by refining an octree. The initial volume is split into eight cubical subvolumes. If the subsurface in this cube is "watertight" it can be directly used to derive the tetrahedrons and further refinement is not required, it is marked as IO cube. If the cube does not contain any triangles of the mesh, it is marked as blank cube. Otherwise it contains some polygons of the mesh without a trivial way to derive an inside, outside classification. Thus it is split again. This process is repeated until cubes around non watertight surface patches (i.e. whole cubes) contain just one vertex or boundary edge. The hole cube can then be divided into tetrahedrons by using the remaining mesh vertex and the boundary edges of the cube. For further classification a graph representation is established. Each tetrahedron inside a boundary cube becomes a vertex in the graph. If the boundary of the tetrahedrons is not part of the initial mesh, the graph vertices are connected by an edge. Blank cubes are represented by a single vertex in the graph. IO cubes are split into two nodes in the graph, for the parts outside and inside the mesh surface and get connected to adjacent hole and blank nodes but not to each other. In addition, the edges are weighted in a complex scheme. The inside / outside classification for the blank and boundary nodes is then obtained by computing a minimum cost cut of the graph, where each cut resembles the insertion of a triangle. The completed shape is now represented by the geometric boundary of this volumetric classification. In a final step, Laplacian smoothing is applied to the newly inserted mesh patches. Note that strictly taken this algorithm is not a local repair algorithm since it works globally and simultaneously closes all holes and gaps. However it preserves all vertices of the input mesh, is solely targeted at repairing holes (and gaps) and imposes high constraints on the input mesh, which does not suit a classification of global repair approaches (see section 4.2). Hence [50] classifies this algorithm as being a local approach. Table 3 summarizes important aspects of the mentioned algorithms.

[69] An adaption of this is used in the skull assembly and completion approach mentioned in Section 5.2.

[70] I.e. a volumetric representation of the shape.

algorithm	input constraints	required parameters	no new inter-sects
[Bøhn and Wozny 1992]	-	-	
[Mäkelä and Dolenc 1993]	-	-	
[Varnuska et al. 2005]	-	-	
[Roth and Wibowoo 1997]	roughly planar hole boundaries	-	
[Brunton et al. 2010]	simple boundary loops	-	
[Pfeifle and Seidel 1996]	-	-	
[Barequet and Sharir 1995]	-	-	
[Liepa 2003]	-	-	
[Zhao et al. 2007]	-	-	
[Branch et al. 2006]	-	-	
[Lévy 2003]	-	-	
[Pernot et al. 2006]	-	-	
[Wang and Oliviera 2007]	roughly planar hole boundaries	moving least squares radius	
[Tekumalla and Cohen 2004]	-	MLS radius	x
[Wagner et al. 2003]	-	simulated annealing parameters	x
[Podolak and Rusinkiewicz 2005]	no degeneracies, self-intersections and singularities	-	x

Table 3: Hole filling algorithms, taken from [50]: All approaches are prone to introduce new degeneracies and self-intersections except for the last three which are however prone to not filling holes successfully. Except for the last, holes should be rather simple in geometry. Otherwise convolved boundaries produce self-intersecting filling patches and islands result either in self-intersections as well or spurious blobs.

Degeneracy Removal

[Botsch and Kobbelt 2001] use a slicing technique to detect and remove degenerate triangles from a manifold mesh. To avoid numerical instabilities, the mesh slicing operator only uses robust predicates to split faces in a controllable manner. While the approach is mainly targeted at the typical output of CAD systems, [Attene 2010] focuses on meshes obtained through digitization of physical objects. The proposed approach iteratively eliminates near degenerate faces by a combination of edge swapping and contraction. In both approaches a threshold angle ε is used to distinguish near degenerate faces that either have an inner angle α for which $\alpha \leq \varepsilon$ or $\alpha \geq \pi - \varepsilon$ is true. Both algorithms are guaranteed to converge when $\varepsilon = 0$. Since both algorithms apply edge contraction to eliminate acute needle-like triangles, they could break the manifoldness of the mesh. If such contractions must be prevented, both algorithms cannot be guaranteed to eliminate all (near) degeneracies. Table 4 summarizes both algorithms, and those discussed in the next section.

algorithm	fixes	input constraints	required parameters	success guaran-teed	accuracy
[Botsch and Kobbelt 2001]	D	manifold	threshold angle ε		approxi-mation
[Attene 2010]	D,S,H	-	threshold angle ε		approxi-mation
[Bischoff and Kobbelt 2005]	S, G	manifold	tolerance distance ε_0, gap width γ_0	x	approxi-mation
[Campen and Kobbelt 2010]	S	no boundary, no degeneracies	-	x	exact
[Granados et al. 2003]	S	-	-	x	exact

Table 4: Degeneracy and Self-Intersection removal algorithms with type of defects fixed (D= degenerate facets, S = self-intersections, H = holes, G = gaps) their input constraints and required parameters, guarantees of success and accuracy of their results. In any case all of the algorithms do not introduce new defects (taken from [50]).

Self-Intersection Removal

[Bischoff and Kobbelt 2005] apply a voxel grid to the input mesh. Vaguely similar to [Podolak and Rusinkiewicz 2005] (for hole-filling) the geometry is changed only within voxels that contain defects. The corrected geometry is guaranteed to stay within a user-provided tolerance distance e0 from the input mesh. The algorithm also closes gaps that separate nearby patches with a maximum gap width γ_0 . The approach presented in [Attene 2010] [71] is designed to repair a number of different defects in meshes obtained through digitization sequentially and imposes no rigid constraints on the input. After converting the mesh to an oriented manifold, closing its holes and removing degenerate facets, it locates and repairs self-intersections. As with the previous algorithms, a spatial subdivisioning is used for efficiency reasons. In contrast to [Bischoff and Kobbelt 2005] the algorithm assumes that the size of the self-intersecting triangles are small to easily fill the resulting holes after their removal. While this is not guaranteed to work with all models, it generally results in lower distortion.

Self-intersection might not be detected correctly due to limited numeric precision. [Campen and Kobbelt 2010] address this by converting the model to a plane-based BSP model [72]. This algorithm guarantees exact and robust computations for an input with limited numerical precision while limiting the impact on run time and memory consumption. [Granados et al. 2003] take an opposite approach and generally use arbitrary precision arithmetics. In contrast to [Campen and Kobbelt 2010] this method can handle input meshes with open boundaries, dangling edges and isolated elements while its run time and memory requirements are significantly higher. Table 4 summarizes the characteristics of these algorithms.

Sharp Feature Restoration

[Kobbelt and Botsch 2003] present and interactive approach to store corrupted sharp edges. The user provides a number of "fishbone" structures that resemble the spine and orthogonal ribs of the sharp edges in the model which are then automatically tessellated. [Attene et al. 2005] presents an automated

[71] Note that the algorithms is also mentioned as degeneracy removal algorithm in the previous section.

[72] Gilbert Bernstein and Don Fussell. Fast, exact, linear booleans. Comput. Graph. Forum, 28(5):1269–1278, 2009.

method for corrupted sharp edges and corners that is based on growing smooth regions around edges by their average dihedral angle. Remaining triangle strips surrounded by smooth regions of the mesh are considered as having aliasing artifacts. The areas are reconstructed by intersection the planar extrapolations of the neighboring smooth regions. The chamfer triangles and edges are then subdivided by new vertices that are located on the intersections of the extrapolations. [Chen and Cheng 2008] present a method to restore sharp features within smoothly filled holes. In contrast to [Attene et al. 2005] the method works for chamfering that affects areas that are wider than a triangle strip. An iterative procedure is used to adjust the face normals. [Wang 2006] proposes a similar method that is targeted at restoring sharp features in dynamically remeshed models. In contrast to the previous two approaches, the algorithm does not support fully automated way to detect whether a path between two smooth areas is subject to chamfering. Thus two threshold parameters have to be provided. All algorithms thus far can potentially introduce new self-intersections. [Attene et al. 2005] could also produce degenerate elements. Table 5 summarizes the characteristics of the mentioned sharp feature restoration algorithms.

algorithm	input constraints	possible parameters	possible new defects
[Kobbelt and Botsch 2003]	manifold	interactive user input	self-intersections
[Attene et al. 2005]	manifold, no degeneracies	-	self-intersections, degeneracies
[Chen and Cheng 2008]	manifold, no degeneracies	-	self-intersections
[Wang 2006]	no noise, no degeneracies	two thresholds	self-intersections

Table 5: Algorithms for restoration of corrupted sharp features along with their input constraints, possible parameters and defect types that could be introduced by them (taken from [50]).

Mesh Denoising

Mesh denoising is a widely studied problem. In this section only algorithms that are particular significant according to [50] are discussed. A very simple approach to mesh denoising is Laplacian Smoothing where at each iteration vertices are moved towards the center of mass of their neighbors. Due to characteristics of most objects, Laplacian Smoothing tends to reduce the overall volume of objects. [Taubin 1995] addresses this problem by alternating inward and outward diffusion steps that are controlled by two parameters λ,μ. However, this class of algorithms does not distinguish between areas that should ideally be smooth and areas that contain important morphological features such as sharp edges. Thus such features are smoothed along with the rest of the object surface.

[Fleishman et al. 2003] adapted the bilateral filtering technique from image processing for mesh denoising. This algorithm can be considered as feature-preserving and must be provided with parameters for the number of iterations *n* and two parameters σ_c,σ_s to tune the bilateral filter. Another feature-preserving algorithm is presented in [Jones et al. 2003]. This method is non-iterative and even works on input meshes with connectivity defects (i.e. polygon-soups). It can be configured through a parameter σ_{noise}, that should ideally be adjusted to match the variance of noise of the input mesh. [Hildebrandt and Polthier 2004] goes one step further and actually sharpens feature lines while removing noise in other regions of the mesh. [Fan et al. 2010] assume that the input mesh represents an object with piecewise smooth surfaces. Features that are located at the intersection of multiple smooth surfaces are sharpened, while high-frequency features in other areas are assumed to be noise and smoothed. This algorithm seems particularly suited for denoising digitized, manmade, and mechanical objects. All mentioned mesh denoising algorithms move vertices to new positions which

could introduce degenerate elements or self-intersections. Table 6 summarizes the characteristics of the mentioned mesh denoising algorithms.

algorithm	type of repair	input constraints	parameters
[Taubin 1995]	N	closed manifold	λ, μ, n
[Fleishman et al. 2003]	N, F	manifold	$\sigma_c \sigma_s, n$
[Jones et al. 2003]	N, F	-	σ_{noise}
[Hildebrandt and Polthier 2004]	N, S	manifold	λ, r
[Fan et al. 2010]	N, S	manifold	n

Table 6: Mesh denoising algorithms and their types of repairing (N = noise removal, F = feature preservation, S = feature sharpening), input constraints and possible parameters (n denotes the number of iterations for iterative algorithms). All algorithms are guaranteed to converge with success. All of them might generate meshes with degeneracies and self-intersections (taken from [50]).

Topological Noise Removal

Topological noise removal is relatively easily accomplished if detailed background knowledge of the specific object can be used. However, generically targeting objects with a non-zero genus [73] is considered to be a very challenging and error-prone task. Thus an interactive approach as suggested by [Sharf et al 2007] might be the most reliable approach. While some topological noise removal algorithms work directly on the mesh, others require a voxelization that leads to modification on the entire mesh. Though technically, this conversion makes them global repair approaches, they are listed here as they focus on the removal of topological noise. [Fischl et al. 2001] present a method that is targeted at removing all handles from 3D models of the brain cortex. The input model is inflated by alternating steps of Laplacian Smoothing [74] and a radial projection (i.e. a mapping of the initial surface onto a sphere). Handles are then detected by searching for big folds in the mapped surface. The patches around handles are then removed and filled with simple, disk-like patches. [Guskov and Wood 2001] detect tunnels and small handles by a local wave front traversal. Then non-separating cuts are identified to remove the inner surface area of tunnels and handles. Finally, the holes along the cuts are filled. The size of the handles and tunnels to be removed can be controlled by a user provided threshold. [Attene and Falcidieno 2006] present an algorithm that gains efficiency by assuming that edge length is of little variance across the mesh (which is generally the case for raw digitized models). A depth-first region growing detects handles and tunnels that are located closely to splitting points in the regions front. All of the mentioned algorithms thus far add material for filling holes and could hence introduce self-intersections. However, topological noise normally results in small handles and tunnels that, when cut out, result in small holes. Effectively this dramatically reduces the probability of introduced self-intersections.

Given a volumetric model representation, [Wood et al. 2004] perform an axis-aligned sweep through the volume to locate and compute the size of volumes. Selective removal of handles is controlled by a threshold parameter. For handle detection a Reeb-graph is continuously updated and analyzed for short non-separating cycles. The removal is performed on the volumetric representation and restricted to preserve geometric detail. [Szymczak and Vanderhyde 2003] propose another method for volumetric models that, instead of computing a Reeb graph, is based on a more simple morphological method called topology-sensitive carving that trades precision for the ability to efficiently process models with a large number of handles. A threshold limits the number of allowed topology-altering operations.

[73] I.e. the objects are assumed to actually have real handles (or grips) besides those that were introduced by topological noise.

[74] Note that, as opposed to most other cases, Laplacian Smoothing tends to increase model volume around small tunnels and concave spikes pointed towards the inside of the object.

An even more efficient approach for 3D images (i.e. voxel-based representation) is presented in [Zhou et al. 2007] and employs an adaptive grid structure to handle very high resolution models (4096^3 voxels) on commodity PC hardware in a few minutes. A discrete curve skeleton is computed whose elements are associated with solid parts of the model. The association is performed in a way that provably prevents the introduction of new handles as a result of breaking detected handles. The algorithm accepts two parameters to control the size of handles and tunnels that should be removed.

[Ju et al. 2007] presents a similar approach but goes on step further. It can not only simplify the topology (i.e. the genus) but modify it to match a provided target shape by introducing handles, tunnels and cavities. Earlier works in the area of voxel based topological noise removal include [Shattuck and Lehay 2001] that removes all handles as well as [Han et al. 2002] which is more flexible in terms that it is not limited to the production of zero genus outputs. Table 7 summarizes the characteristics of the presented noise removal algorithms.

algorithm	input constraints	parameters	possible new defects
[Fischl et al. 2001]	oriented manifold	-	self-intersections
[Guskov and Wood 2001]	oriented manifold	threshold	self-intersections
[Attene and Falcidieno 2006]	-	threshold	self-intersections
[Shattuck and Lehay 2001]	no large holes	-	(chamfered features)
[Han et al. 2002]	no large holes	-	(chamfered features)
[Szymczak and Vanderhyde 2003]	no large holes	threshold	(chamfered features)
[Wood et al. 2004]	no large holes	threshold	(chamfered features)
[Zhou et al. 2007]	no large holes	two thresholds	(chamfered features)
[Ju et al. 2007]	no large holes	target shape	(chamfered features)

Table 7: Algorithms to repair topological defects with their input constraints, possible parameters and new defects potentially introduced by them. The three algorithms that do not require any parameters repair only models with zero desired handles, i.e. they might remove any handles (However Han et al 2002 al distinguishes between foreground and background handles). The lower six methods work on voxelization, thus the input must not have large holes that would inhibit the inside outside classification that is implied by this conversion. Subsequently the voxelized representation has to be converted back to a mesh which implies possible introduced feature chamfering and also aliasing effects. If the input is guaranteed to have very small or no holes, the resolution of the voxelization could be increased to lower the amount of newly introduced defects (Table adapted from [50]).

4.2. Global Approaches

The local methods described in the previous subsection are mainly focused on removing defects of a specific type. In general, the absence of these defects is required for further processing in downstream applications. However it is complicated to guarantee absence of defects of different type after isolated, local processing, as many of the methods might introduce new defects of different type (e.g., especially self-intersections). In the case of collocated defects of different type, it might be impossible to consistently compute plausible results with a loosely coupled pipeline of local repair methods.

Since for many applications the meshes are assumed to represent watertight models, many of the global repairing methods apply some kind of intermediate volumetric representation. This also enables

the use of more meaningful disambiguation rules. It is also guaranteeing manifoldness of outputs by relying on contouring methods for robust reconversion from the intermediate volumetric representation to an output mesh. Essentially, repairing models in voxel-space boils down to deciding whether a voxel is located inside or outside of the model. On the other hand, the conversion of model representations might lead to aliasing effects and under some conditions even introduces handles (e.g., when the resolution in voxel-representation is too low to accommodate small interstices of the mesh model). As global repair methods are targeted at several defect types at once, they are not grouped by defect category as in the previous subsection but by their input constraints instead. Note that often these constraints are not of strict nature and the algorithms will compute a manifold mesh even if they are violated. However the output is less likely to be of a plausible nature. Table 8 summarizes the characteristics of mentioned global repair algorithms.

Input without Gaps/Holes

The most simple methods for global repair just convert the input mesh to a voxel-based representation and back again to a mesh. Each voxel is basically assigned a signed scalar, where the sign reflects whether the voxel is inside or outside the object and the absolute value might indicate the distance of the voxel from the object surface. For hole-free input the voxelization can be computed by simple flood-filling a starting seed of voxels that is located around the mesh surface [Oomes et al. 1997]. Without such a starting seed, the bounding box of the mesh could be used. This effectively results in the computation of the outer hull of the mesh in voxel space where internal structures are lost during conversion [Andújar et al. 2002]. As a side effect, the discretization process discards high-frequency noise and details of the input surface.

When the input mesh contains significant holes and gaps, additional measures need to be taken to plausibly compute voxel signs as mentioned in the following sections.

Input With Known Orientation

Probably the most common source of holes and gaps is occlusion during digitization. However due to the (optical) nature of occlusion, there actually is geometric information on the maximum geometric extent and its orientation for such occluded areas.

[Curless and Levoy 1996] exploit this line-of-sight information to classify voxels as "definitely outside" and "possibly inside". However this rather conservative approach produces implausible excess geometry for larger holes and gaps. [Furukawa et al. 2007] employ an additional heuristic to reduce these effects.

[Davis et al. 2002] propose a diffusion process where the input is converted to a signed distance field aligned to the surface and then iteratively diffused away from the initial surface (i.e. the field is moved towards the inside of the hole and away from the last ray that was captured during digitization), effectively closing gaps and holes up to a parametrized threshold width. [Guo et al. 2006] and [Masuda 2004] describe variants of this approach with different diffusion behavior that is more suited towards non-smooth surfaces. By these methods, holes and gaps are shrunk iteratively. Each additional iteration then implicitly smoothes out.

In [Sagawa and Ikeuchi 2008] hole regions are considered globally. The initial inside/outside voxel classification takes into account the (camera) normals of the input mesh. Voxels along the interfaces in hole regions are iteratively reclassified to minimize the interface area. [Shen et al 2004] present an implicit approach to surface reconstruction that is based on a least squares interpolation of the input mesh. Holes and gaps are closed if they are below a certain threshold parameter. Finally, [Verdera et al 2003] apply a partial differential equation adapted from image inpainting methods to obtain smooth fillings. However setting up the necessary boundary conditions for non-trivial hole and gap constellations can get very complex.

Arbitrary Orientation

[Ju 2004] proposes a method to patch holes in a voxelized version of a mesh. While output is guaranteed to be manifold, gaps, which are normally not bounded by a closed loop, are likely to not get closed due to an explicit tracing of boundary loops that is used for hole detection.

[Nooruddin and Turk 2003] explicitly discard internal structures of a voxel model by considering everything between the first and last intersections of numerous casted rays with inside-voxels as belonging to the inside of the model. In case of holes and gaps, this leads to misclassifications which are countered by so called ray-stabbing. Rays that do not intersect with well-classified inside voxels cut off potential excess geometry that would otherwise be introduced by the previous step.

[Bischoff et al. 2005] focus on the outer hull of the model as well but use morphological closing operations applied on the voxel model. The outer hull is then found by flood-filling the outside of the model. An octree voxelization scheme is applied in conjunction with hole and gap boundary detectors to spatially restrict morphology to increase efficiency.

[Hétroy et al. 2011] compute the outer hull by using a discrete membrane-shrinking technique. In their approach, morphological operators are applied globally so that intact concave regions are altered by "closing" them as well.

While [Hornung and Kobbelt 2006] coarsely determine the outer region by using morphological operators and flood-filling as well. A refined solution is subsequently computed by a global graph-cut approach minimizing the area of the filling patches.

All of the previous four algorithms types mentioned in this section not only remove redundant internal structures but also voids that could be intentional. [Nooruddin and Turk 2003] as well as [Spillmann et al. 2006] could preserve internal voids correctly if internal structures are intentional and not redundant.

[Murali and Funkhouser 1997] present a method that is very different from the algorithms mentioned so far. It does not affect geometry in intact regions of the mesh, with only alterations to its tessellation. Instead of voxels, a volumetric space partitioning is established by arbitrary polyhedral cells that are aligned with the input polygons and partitioned using a BSP-tree. In a global process, the sign assignment of the cells is optimized so that the output mesh conforms to the input mesh as good as possible. The geometry of the patches that are used for hole filling is however highly dependent on the structure of the BSP-tree and can be very unintuitive for larger holes.

algorithm	input constraints	signing method
[Oomes et al. 1997]	no significant holes / gaps	flood-filling
[Andújar et al. 2002]	no significant holes / gaps	flood-filling
[Curless and Levoy 1996]	oriented range meshes	line-of-sight
[Furukawa et al. 2007]	oriented range meshes	line-of-sight
[Davis et al. 2002]	oriented range meshes	normals and diffusion
[Guo et al. 2006]	oriented	normals and diffusion
[Masuda 2004]	oriented	normals and diffusion
[Sagawa and Ikeuchi 2008]	oriented	normals and area minimization
[Shen et al 2004]	oriented	normals and MLS interpolation
[Verdera et al 2003]	oriented	normals and inpainting
[Ju 2004]	(no significant gaps)	parity-counting
[Nooruddin and Turk 2003]	-	parity-counting, ray-stabbing
[Bischoff et al. 2005]	-	morphology and flooding
[Hétroy et al. 2011]	-	membrane shrinking
[Hornung and Kobbelt 2006]	-	morphology and graph cut
[Spillmann et al. 2006]	-	parity-counting
[Murali and Funkhouser 1997]	(no significant holes)	global sign optimization

Table 8: Global repair algorithms with their input constraints and their approaches for making inside / outside decisions. In principle, all algorithms are able to guarantee defect free output. However there can be no guarantee that the result is semantically plausible. Depending on the final mesh abstraction step employed, new near degeneracies and singularities might be generated. According to [1] this can be prevented relatively easy in many cases. Input constraints listed in brackets are not mandatory to obtain a manifold output mesh, however they are recommended for a semantically plausible result. (taken from [50]).

5. COMPLEX SHAPE REPAIR

In contrast to the basic shape repair approaches from Section 4, the repair techniques in this section are mostly application specific [75] and are often composed of several basic repair algorithms as well. Due to the complexity of the (partially ill-posed) problems in the application domain, they often rely on additional knowledge that is provided by the user or external, annotated sets of template models as well as other sources of information such as e.g., 2D images of an object. To repair missing large parts of the physical object, some approaches try to recombine existing shapes or segments to compose a plausible solution. Hence similarity searches and segmentation techniques are included in the methods as well.

5.1. Mesh Completion and Inpainting

The hole filling and gap closing algorithms mentioned in Section 4.1 are generically geared towards smooth surfaces and create more or less smooth solution patches by some form of interpolation or fairing. For non-smooth and structured surfaces, this produces implausible fillings. Mesh completion algorithms address this issue by trying to mimic structure and features that are found in intact parts of the input model or additional information which is made available to the methods.

The first group of algorithms is not actually repairing the input mesh but replaces it by a template model that is fitted to the input by some form of non-rigid transformations. This approach is inherently limited to narrow classes of input meshes and has been used for scans of humans heads [Blanz et al. 2004], bodies [Allen et al. 2003] or teeth [Kähler et al. 2002]. Most of these methods require additional user input such as correspondences or feature markers in the input data.

Other methods keep the input mesh and only fill in the holes with special template patches. However, the merging of existing input meshes with the templates is hard to achieve when robustness needs to be guaranteed, they expect point samples instead of meshes [Sharf et al. 2004], [Bendels et al. 2005], [Breckon and Fisher 2005], [Park et al. 2006], [Xiao et al. 2007]. This eliminates the need to deal with the topology of the input mesh but implies extraction of point samples and complete remeshing if they are used for mesh repair. In principle, they could however be adapted to try to retain the tessellation of the intact parts of the input. These methods can be categorized according to whether they rely on intra- or inter-shape similarities.

[Sharf et al. 2004] propose an inter-shape similarity approach. Adapted from context-based image completion methods, intact parts of the input are used to repair similar, but due to holes, defective parts of the mesh while progressing from a large to small scale. [Park et al. 2006] describe a variation of this approach. The similarity search is performed in a discrete manner regarding scale, rotation and location. Other methods such as [Bendels et al. 2005] and [Breckon and Fisher 2005] operate on a finer per-point basis.

[Nguyen et al. 2005], inspired by 2D texture synthesis, applies an approach for generating local gradient images to hole regions of the input mesh. Similar methods based on texture synthesis applied to encoding geometric detail is described in [Xiao et al. 2007].

[Xu et al. 2006] infers the missing geometry information by a shape-from-shading technique respective to a photoconsistency measure applied to photos taken from the physical object that was digitized. [Jia and Tang 2004] apply a tensor voting approach to derive hole filling geometry, unfortunately no detailed information is provided on how the merging itself is realized.

[Kraevoy and Sheffer 2005] present a method that relies on a template mesh that is fitted either globally or locally to the input mesh in order to derive hole filling patches. User intervention is needed in cases of complex topology or non-trivial gap and hole constellations. [Pauly et al. 2005] use a repository of template models to fill in holes and gaps from several models that are selected by user-specified keywords and shape similarity.

[75] Note that some of the basic repair methods were also developed in application specific context, however these methods lack other aspects of complex repair algorithms, such as the use of additional external information.

Table 9 summarizes the mentioned mesh completion algorithms.

algorithm	input constraints	required parameters	possible new defects
[Sharf et al. 2004]	- (point based)	resolution	(topological noise, aliasing)
[Bendels et al. 2005]	- (point based)	number of scale levels	(topological noise, aliasing)
[Breckon and Fisher 2005]	- (point based)	window size	(topological noise, aliasing)
[Park et al. 2006]	- (point based)	resolution	(topological noise, aliasing)
[Xiao et al. 2007]	- (point based)	several	(topological noise, aliasing)
[Nguyen et al. 2005]	roughly planar hole boundaries	approx. threshold, number of scale levels	self intersections, degeneracies
[Xu et al. 2006]	roughly planar hole boundaries	model-aligned images	self intersections, degeneracies
[Jia and Tang 2004]	roughly planar hole boundaries	tensor voting parameters	self intersections, degeneracies
[Kraevoy and Sheffer 2005]	manifold with boundary	templates and correspondences	self intersections
[Pauly et al. 2005]	-	model database, keywords	self intersections, degeneracies

Table 9: Mesh completion algorithms and their input constraints and mandatory parameter requirements. The algorithms focus on generating a plausible geometry rather than preserving as much as possible of the input mesh structure. Hence strict statements about robustness, guarantees, etc., cannot always be made. The point-based algorithms require a subsequent reconstruction of the mesh which could possibly induce additional feature chamfering, aliasing and topological noise (taken from [50]).

Free-form Surface Inpainting

3D shape inpainting methods are inherently complex and often benefit from user interaction. Therefore, Bendels et al. proposes a semi-automatic approach for 3D inpainting [76]. The basic idea is that the user performs the coarse-level inpainting manually, while the system automatically finishes the inpainted surface patches to the shape. The method workflow starts by the user marking up the defective area. The area is removed and replaced by a template surface patch which serves as a starting point to repair the defective area. The template can be of a primitive shape, e.g., a planar patch, or a surface segment manually copied from an external repair model. The temporary patch is fine-adjusted to the defect by a nonrigid transformation to fulfill its boundary constraints. The user then interactively adapts the inserted template patch to indicate the desired shape, using a handle-based transformation approach. After the interactive adaption is done, the system automatically fine-merges the edited patch to the input shape. Figure 8 illustrates. Overall, this approach is relying on interactive input at important stages.

[76] Gerhard H. Bendels, Michael Guthe, and Reinhard Klein. Free-form modelling for surface inpainting. In The 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa (Afrigraph 2006), pages 49–58. ACM Siggraph, January 2006.

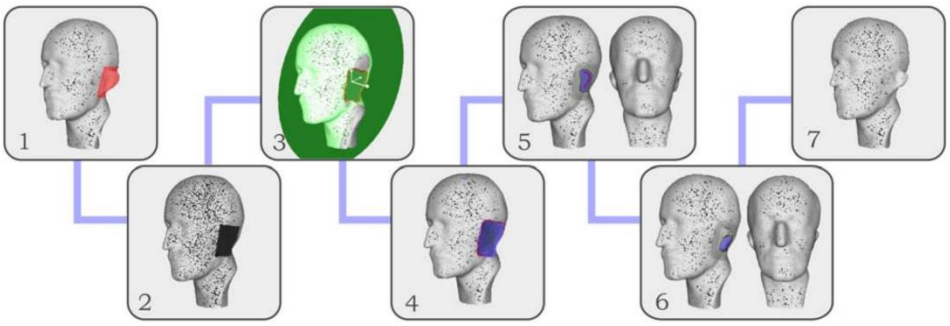


Figure 8: Free-form modeling for surface Inpainting (Figure taken from [76]).

Surface Completion by Minimizing Energy Based on Similarity of Shape

In [77, 78] Kawai et al. propose to determine an optimal patch for filling holes in a given mesh model by evaluating the local similarity of the shape around the defect and other candidate patches from the model which could fill the missing part. In their approach, the user manually selects the area that should be repaired in the model. For determining similarity, the area of the solution patch (covering the missing region) is extended to a so called data region that includes adjacent facets of the model. Next, an initial trivial solution patch is generated that adds just one vertex to the mesh and connects it to the holes boundary edges by zippering. Then, in a next step points are added and deleted from the solution iteratively to match the point density and distribution in the data region. In a final step, the position of the vertices of the solution patch are iteratively shifted until the energy functions converges. Figure 9 illustrates the basic workflow. The method has the advantage that its repair is relying only on the given model and does not require external template information or other constraints. On the other hand, a prerequisite is that the model shows local self-similarities for the defected areas, such that model information can be re-used.

[77] Norihiko Kawai, Tomokazu Sato, and Naokazu Yokoya. Surface completion by minimizing energy based on similarity of shape. 2008 15th IEEE International Conference on Image Processing, pages 1532–1535, 2008.

[78] Norihiko Kawai, Tomokazu Sato, and Naokazu Yokoya. Efficient surface completion using principal curvature and its evaluation. In Proceedings of the 16th IEEE international conference on Image processing, ICIP'09, pages 521–524, Piscataway, NJ, USA, 2009. IEEE Press.

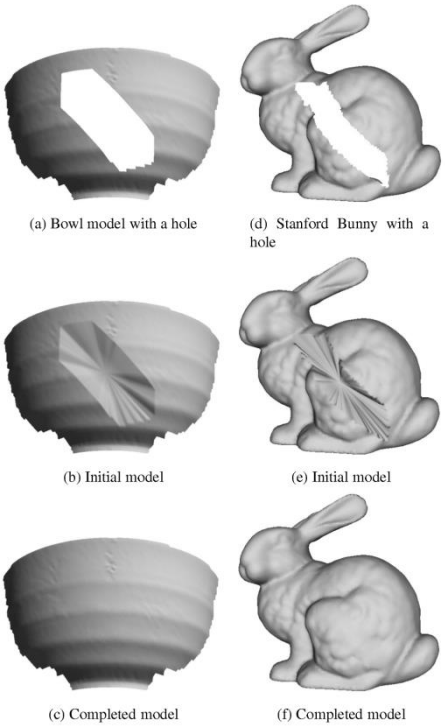


Figure 9: Surface completion by minimizing an energy function: The user selects and cuts out an area that should be repaired (a),(d). An initial patch, that adds one vertex to the mesh is generated to fill the hole (b),(e). In an iterative process, points are added and deleted from the solution by considering their density in comparison to the surrounding patches of the model. The positions of the points are then slightly shifted to minimize an energy function that is inverse to a local similarity measure between the solution patch and the whole model (c)(f) (taken from [**78**]).

5.2. Assembly-Based 3D Modeling

Modeling by Example

A number of systems have been proposed that support the modeling of new shapes by recombination of existing shapes from a given shape repository. While this is an interesting approach, such techniques typically require the solution of a number of non-trivial problems, including searching for candidate input shapes, interactive or automatic segmentation of relevant model parts, and merging and post processing parts to form a final new or repaired shape. We next describe a number of relevant works from the state of the art.

In their classic paper, Funkhouser et al. [79] suggest a methodology and system for modeling of new forms from existing forms. The modeling process starts by searching for starting elements using either text-queries (if textual annotations are available in the shape repository) or query-by-example. Semiautomatic cutting and pasting assists in the assembly of a composite shape. User interaction is critical in this approach as the user annotates the target area where a relevant shape parts need to be inserted (e.g., 2 elongated blocks could be input to search for a new arm for a statue). A semi-automatic segmentation method takes a minimalistic user input (a user stroke), and the system then determines the cut-line according to a weighted sum of quality measures for the cut, including length, orientation and continuity of the stroke. Finding the cut amounts to computing shortest paths between sample of start and end points in neighborhood of the stroke, and taking the minimum cost. For the search method, a volume-oriented version of the surface-distance integral is used, based on a distance

[79] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. Modeling by example. ACM Transactions on Graphics (Proc. SIGGRAPH), August 2004.

transform of the models. Binary weights mark regions to include (or exclude) from the search. The advantage is that the method is robust and fast, as only one descriptor is needed for each model. As a disadvantage, the objects need all be aligned and scaled in a reference frame. The user selects local areas to search for or exclude. This makes the approach to depend strongly on user interaction. Once candidate parts to compose have been retrieved and segments, stitching of matched parts can take place. A custom alignment approach aligns the replacement part to the replaced section of the given model. The approach is semi-automatic in that the user selects the join boundaries, and an algorithm finding then the point correspondences to be stitched. Figure 10 illustrates the basic steps in the modeling by example approach.

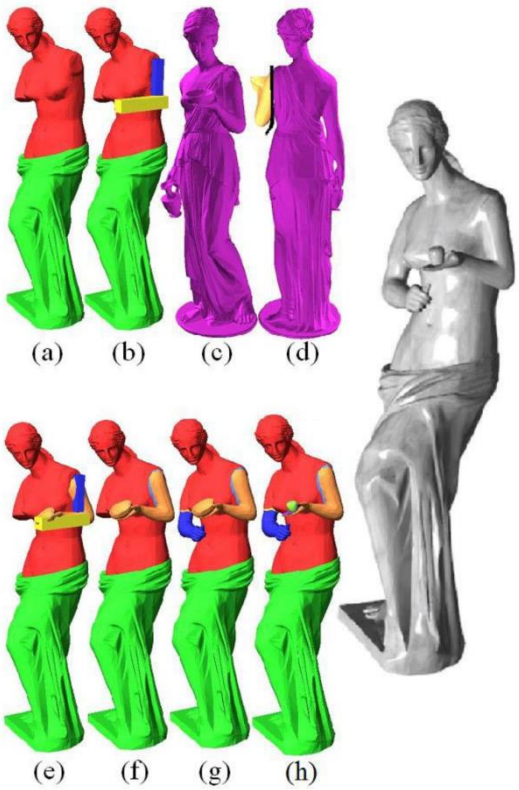


Figure 10: Modeling by example: Instead of modelling the missing parts of the Venus de Milo sculpture (a) from scratch, two boxes are drawn by the user (b) and a query on a model repository yields a model of the sculpture of Hebe with an arm in similar pose (c). The arm is cut off by the aid of a user provided mouse stroke (d). It is then pasted to the Venus de Milo model with automatically determined alignment and scaling (e). A hole is cut into Venus' shoulder and stitched to the open boundary of Hebe's arm (f). A second arm is retrieved from the repository and added in a similar fashion in (g). In (h) the bowl has been cut of the arm and is replaced by an apple found through a text search in the repository. (Figure taken from [**79**]).

Sketch based Search and Composition of 3D Models

Fully automatic recombination of existing 3D shapes to form new models is not possible in the general case, but depends on the application and user context. To this end, sketch-based approaches are researched as a lightweight, friendly interface to involve the user in the re-combination process [80]. In

[80] Matthew T. Cook and Arvin Agah. A survey of sketch-based 3-D modeling techniques. Interacting with Computers, 21(3):201–211, July 2009.

[81], an extension of the Modeling by Example approach (see above) is discussed. There, instead of simple box queries, the user provides a 2D sketch to search for matching segments in a given model repository. The sketch triggers a contour image based similarity search on model fragments located in the repository. For 2D contour similarity search, the sketch is normalized for scale, translation and rotation by PCA normalization. Along with the model fragments, 2D contour images in 24 axis aligned orientations are extracted and matched against the sketches. The rendered 2D projection is then centered and scaled isotropically to normalize for size and translation. The system uses different descriptors for both sketches and contour images, including Turning Functions and Fourier Descriptors. After the similarity search took place, the user is provided with a selection of the most relevant fragments. The fragment is then aligned to the model using the sketch as cue for transformation, orientation and scaling. The resulting fragment position is slightly shifted so that larger overlaps are removed while at least one point of the fragment touches the model. Finally, the fragment is rotated to find in total at least three points that are in contact to the model. Figure 11 illustrates the basic concept of the approach.

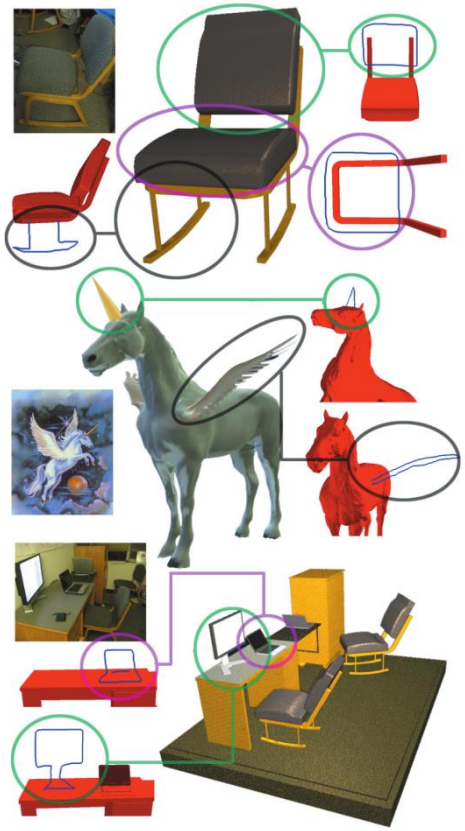


Figure 11: Sketch based search and composition of 3D Models: An existing model can be extended interactively by the aid of user provided sketches. The sketches are used for a similarity search on an input shape repository. Matching segments are aligned and scaled according to the user-sketch and stitched onto the model (Figure taken from [**81**]).

Probabilistic Reasoning for 3D Modeling

Chaudhuri et al. [82] proposed a system similar in spirit to [79, 81]. Instead of relying on user provided boxes or sketches to retrieve plausible shape parts to compose, the systems uses a previously trained

[81] Jeehyung Lee and Thomas Funkhouser. Sketch-based search and composition of 3d models. In Proceedings of the Fifth Eurographics conference on Sketch-Based Interfaces and Modeling, SBM'08, pages 97–104, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[82] Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. Probabilistic reasoning for assembly-based 3d modeling. ACM Trans. Graph., 30(4):35:1–35:10, July 2011.

Bayesian Network to recommend probably suitable parts. The network captures semantic and geometric properties learned from the spatial and shape similarity relations existing among a target shape repository. To that end, models from the shape repository are segmented and semantically annotated as a requirement. A Conditional Random Field based approach by Kalogerakis et al. [83] is used to automatize large parts of the segmentation and semantic labeling of the training-models. The system continuously evaluates the current partial model, comparing it with the network, and updating a list of suggested shape components to add to the current model. The user step by step selects one part from the list of ranked suggestions; components can be adjusted regarding scaling, rotation and position, and finally be attached to the surface of the current model. The system also infers symmetry relationships between components and can suggest these to the user. Figure 12 illustrates. Kalogerakis et al. [84] further extended the approach of Chaudhuri et al. Specifically, after the training step, the system does not require additional user input but automatically generates a set of plausible recombinations from the training-models by inferencing. The user can subsequently review and reject generated recombinations in an optional step. Figure 13 illustrates.

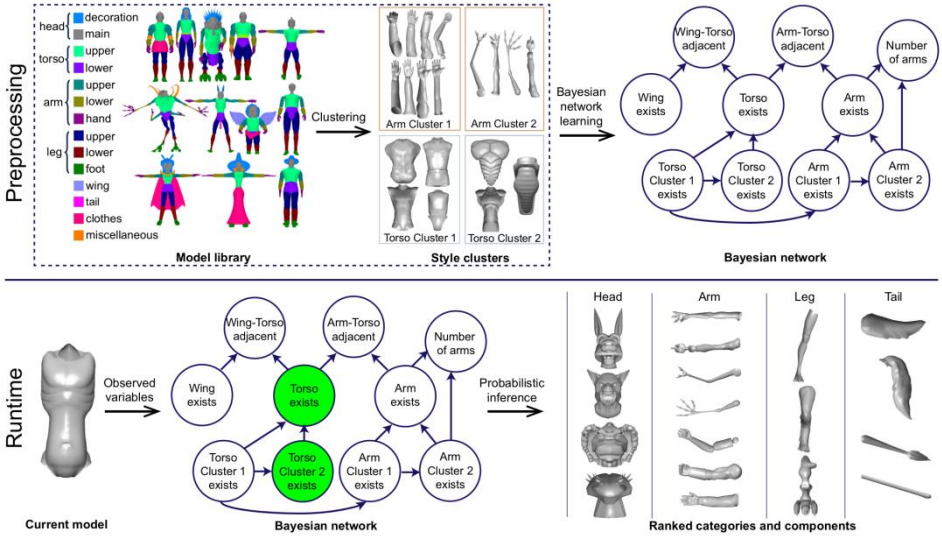


Figure 12: Probabilistic Reasoning for Assembly-based 3D Modeling: In a preprocessing stage, a set of training models is automatically segmented and semantically labeled (top-left). A Bayesian network is trained with the data and taking into account the alignment of the segments, their geometrical properties and their semantic labeling (top-right). At run time, the currently edited model is continuously evaluated by the network to suggest a ranked list of matching parts through inferencing (bottom) (Figure taken from [**82**]).

These approaches are interesting in that they can be applied to any shape repository, independent of a particular application domain. The interface is accessible also to novice users, as it does not require particular skills for query formulation, but simple browsing and selection among the candidate components is sufficient to retrieve components for assembly. The effectiveness of these systems, however, depends on a number of parameters (including similarity functions), quality of geometric processing techniques (for segmenting and merging of components) and availability of semantically annotated shape repositories.

[83] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3d mesh segmentation and labeling. ACM Trans. Graph., 29(4):102:1–102:12, July 2010.

[84] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model for component-based shape synthesis. ACM Transactions on Graphics, 31(4):1–11, July 2012.

Figure 13: A Probabilistic Model for Shape Synthesis: Given 69 training creatures (green), the system by Kalogerakis et al. synthesizes 563 creatures (blue) using a previously trained Bayesian network that takes into account semantic labeling and geometric properties of the pre-segmented training-models (Figure taken from [**84**]).

Skull Assembly and Completion using Template-based Surface Matching

Besides open-ended modeling as supported by the previously discussed works, re-assembly modeling based on templates can support specific application domains. Wei et al. [85] proposes a framework for the assembly of broken skull parts relying on a shape template. As input separately digitized skull fragments are imported into the system. The assembly then performs in an automatic way, matching the fragments to a given complete skull template. Since the template skull might differ significantly in local small scale features, an invariant matching procedure is needed. The authors propose to first use coarse spin images with a bin size of around the fragments bounding box to perform an initial matching. Then, a refined spin image signature is used to compare the fragment spin images with spin images from the template skull. To compute an alignment between fragment and skull, a subset of points is selected automatically by a particular interest point detector. During computation of the spin images bin values, points that would normally only affect adjacent bins are also taken into account to some extent, this effectively works similar to a low pass filter to blur high-frequency features as well as potential aliasing effects and makes the process more robust against sampling distance and resolution differences between fragment and skull. In addition, points for which adjacent facets have a largely deviating normal orientation are ignored for the spin image computation, which further reduces the impact of high frequency features. After computation of correspondences, mean-shift clustering is used to identify a set of five best matching points on the template skull for every relevant point of the fragment. The distance of the points in the fragment is then compared with the distance of their correspondences on the skull to filter out inappropriate matching. Then, a rigid transformation (rotation, shifting and uniform scaling) is performed on the fragments to optimize the matching of the the aligned fragments so that they best match the template. The transformations are then optimized globally according to a least square method that utilizes a nonlinear error-function which penalizes intersecting fragments. To complete the skull, a non-rigid mapping between the template skull and the ensemble of fragments is computed. Furthermore, symmetry of the skull fragments is exploited to mirror missing parts on one side of the fragment skull to the opposite site. Parts that are still missing are copied from the template skull using non-rigid transformations. To finish the model, resulting gaps in the surface curvature are smoothed. Figure 14 illustrates the repair process.

[85] Li Wei, Wei Yu, Maoqing Li, and Xin Li. Skull Assembly and Completion Using Template-Based Surface Matching. 2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission, (1):413– 420, May 2011.

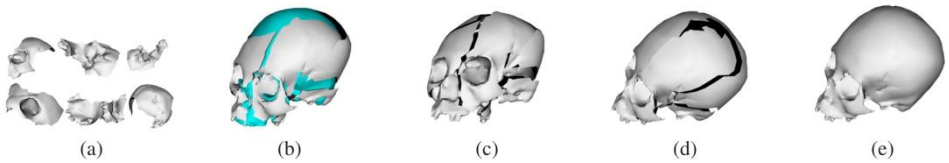


Figure 14: Skull Assembly and Completion using Template-based Surface Matching: digitized but unaligned skull fragments (a) are matched and aligned to a (blue) template skull (b) and rigidly transformed to best match the skull size of the template (c). A global optimization process minimizes the gaps between the fragments (d) Gaps and Holes are inpainted (e) (Figure taken from [**85**]).

Structure recovery by part assembly

Shen et al. [86] presented a local-to-global approach which can recover object structures within noisy 3D scans based on a repository of training shapes. The approach assumes as input a low-quality point cloud scan and an RGB image obtained from the same point of view. Furthermore, it requires a set of training shapes which are segmented and semantically annotated. The recovery approach follows three main steps. First, a number of candidate components from the repository are matched to regions of the point cloud (local matching). The method first computes a coarse alignment of the input data to the repository model and then compares the template structures against the point cloud and image data. For the latter comparison, an edge image is extracted from the RGB image to help the matching. In a second step, matched object parts are grouped together based on proximity, goodness of match to the input data, and degree of non-overlap. Thereby, structural compositions are formed. In a post processing step, the set of components is conjoint. Note that in the approach, the reconstruction can stem from different input objects, thereby possibly, synthesizing new models. Figure 15 illustrates the basic ideas of the method. As a related work, the approach presented in [87] implements a global search, where an incomplete point cloud is matched against globally, and roughly similar model templates with applications in CAD model reconstruction.

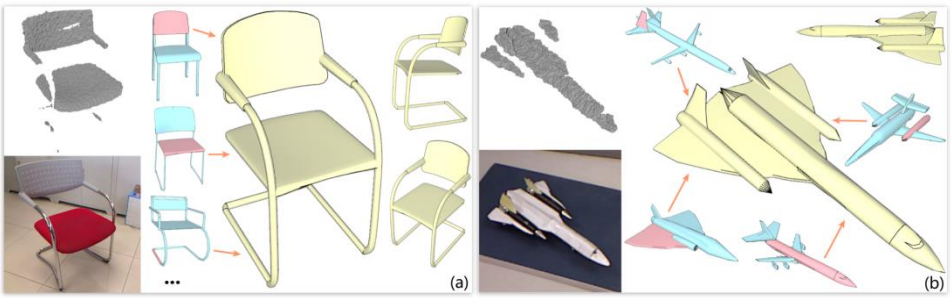


Figure 15: Structure recovery by part assembly: Segments of Models (blue and red) in a repository are aligned with point clouds obtained by a Kinect (upper left corners) to compose a new 3D model (yellow) (Figure taken from [**86**])

5.3. Symmetry-driven Segmentation and Part Repair

A family of related work in object repair is based on the idea, that many shapes entail symmetries in their parts, a fact which can be exploited by repair and reconstruction methods. The symmetry concept matches the Invariance Principle of "Gestalt Psychology", which is one of the four key principles that are claimed to be a fundamental basis of the nature of human perception. While some objects are globally symmetric (e.g., the global reflectional symmetry of human skulls is used in the previously

[86] Chao-Hui Shen, Hongbo Fu, KangChen, and Shi-MinHu. Structure recovery by part assembly. ACM Transactions on Graphics, 31(6):1, November 2012.

[87] A Bey, R Chaine, R Marc, and G Thibault. Effective shapes generation for Bayesian CAD model reconstruction. Proc. EG Workshop on 3D Object Re- trieval, pages 63–66, 2012.

mentioned Template Based Skull Assembly and Completion approach of [85], the number of partially intrinsic reflectional symmetric objects is much larger. An object is classified as partially reflectional intrinsic symmetric, if it can be segmented in a way such that at least one of its segments is reflectional symmetric on its own. We next briefly survey a number of works which are based on the principle of symmetry.

Xu et al. [88] propose a method to detect and extract partial intrinsic reflectional symmetries (PIRS) in 3D shapes. Given a closed manifold mesh, a voting scheme is developed that computes an intrinsic reflectional symmetry axis (IRSA) transform. This transform is a scalar field over the mesh that accentuates the most prominent IRSAs of an object. In the next step, Voronoi boundaries between the sets of pairwise symmetrical points on the mesh surface are computed according to the values of the IRSA transform. These boundaries are then refined in an iterative process that extracts the final intrinsic reflectional symmetric axes. These IRSAs can then potentially be exploited for many tasks e.g., serving for feature region detection, shape segmentation (see Figure 16), and shape repair (see Figure 17 for an example work). Further application possibilities of the IRSA include shape compression or high-level model editing functionality.

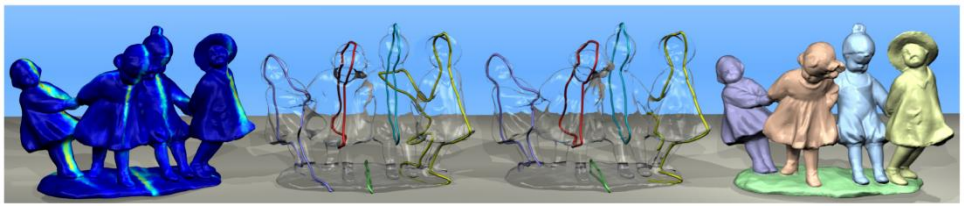


Figure 16: Intrinsic Symmetry Driven Segmentation: For a closed manifold mesh a scalar field, accentuating prominent partial intrinsic reflectional symmetries (PIRS) is computed (first model on the left). Pairwise reflectional symmetric points around the most prominent spots are grouped and used for the computation of a closed Voronoi boundary between them (second model). An iterative refinement scheme is used to extract the final set of intrinsic reflectional symmetry axes (IRSAs) which can be open curves (third model). These IRSAs are incorporated into a mesh segmentation scheme and can yield highly semantic results (right model) (Figure taken from [**88**]).

Several further works proposed to detect and extract partial intrinsic symmetries. E.g., in [89] Xu et al. develop an approach to extract a hierarchy of ISRAs for different scales of reflectional symmetric segments of a model. In [90], Berner et al. propose a more complex, graph-based approach that segments objects into similar (i.e., symmetric) parts. The approach does not only detect reflectional symmetries but tries to mimic human perception. In [91], Raviv et al. propose a framework that detects full and partial reflectional symmetries in shapes that were created by non-rigidly transforming fully or partial symmetric shapes. E.g., the algorithm can detect symmetries in a model of a human body regardless of the pose of its limbs.

[88] Kai Xu, Hao Zhang, Andrea Tagliasacchi, Ligang Liu, Guo Li, Min Meng, and Yueshan Xiong. Partial intrinsic reflectional symmetry of 3D shapes. ACM Transactions on Graphics, 28(5):1, December 2009.

[89] Kai Xu, Hao Zhang, Wei Jiang, Ramsay Dyer, Zhiquan Cheng, Ligang Liu, and Baoquan Chen. Multi-scale partial intrinsic symmetry detection. ACM Trans. Graph., 31(6):181:1–181:11, November 2012.

[90] Alexander Berner, Martin Bokeloh, Michael Wand, Andreas Schilling, and Hans-Peter Seidel. Generalized intrinsic symmetry detection. Research Report MPI-I-2009-4-005, Max-Planck-Institut für Informatik, Stuhlsatzen- hausweg 85, 66123 Saarbrücken, Germany, August 2009.

[91] Dan Raviv, Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Full and Partial Symmetries of Non-rigid Shapes. International Journal of Computer Vision, 89(1):18–39, February 2010.

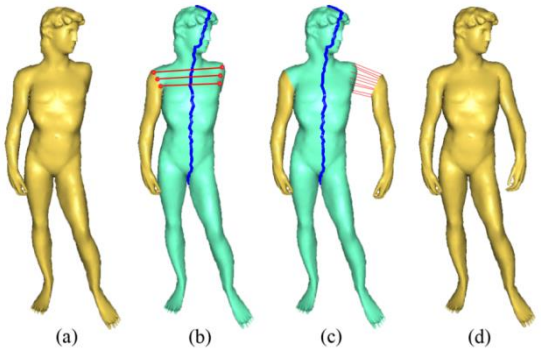


Figure 17: Intrinsic symmetry-driven Part Repair: Starting from a David sculpture with an initially missing arm (a), PIRS segments are computed before asymmetric boundaries between the center segment and its adjacent segments are detected (b). The arm segment is reflected along a plane that best fits the IRSA of the center segment. Subsequently, an alignment is computed between the reflected arm and the boundary of the center segment (c). Finally, the arm and center segment are merged by stitching (d). (Figure taken from [89]).

6. CONCLUSIONS ON SHAPE REPAIR

Many of the basic and complex shape repair approaches that were introduced in the previous sections are explicitly targeted at the repair of digitized objects. Thus they seem to be readily applicable to the PRESIOUS use case. In particular, the assembly based 3D Modeling approaches is a promising way to address the repair of partially reassembled cultural heritage objects. Certain aspects of other, even more automated repair approaches might also be adapted for the repair of digitized cultural heritage objects and fragments. In addition for many of the introduced basic repair categories that are required for numerous steps in the complex approaches, free implementations are available [92]. A possible workflow for object reassembly and completion in the context of PRESIOUS is outlined in the next section.

Many of the referenced sources provide little to no comparative evaluation about their performance for high resolution models obtained by 3D scanners. Run time requirements, robustness (especially to high frequency noise and scale invariance) are not known. Thus in Section 6.2, we focus on evaluation aspects that are important when considering algorithms for use in PRESIOUS.

6.1. A Possible Approach for Reassembly and Repair of Fragmented and Incomplete Cultural Heritage Objects

Given the introduced approaches we sketch a possible outline of how the object repair workflow can be arranged. The sketch is however of conceptual nature and actual feasibility of the individual steps within the project has yet to be assessed. Note that not all suggested steps seem promising for certain object classes and defects with larger impact on the overall shape and structure.

Besides the reassembly of input fragments which is described in part B of this Deliverable, the repair should aim at comparing partially reassembled objects with models from a repository to support further classification either manually by the user or in an automated way. Hence in the first step a repository should be populated with an initial set of model data.

In a second step, the models in the repository are then segmented and annotated. While the segmentation could be achieved in a fully automated way (e.g., by Partial Intrinsic Symmetry Detection as in [88]), semantic labeling cannot be automated completely (partial automation is presented in [83]). While the semantic annotations enable meta-data driven search by the user, they could also be exploited in a reasoning driven approach for reassembly or directly to synthesize new recombinations of the initial model set to increase the size of the repository (see[82, 84]).

In a third step, an interest point detection and local feature extraction is performed on the models and their segments. The resulting feature vectors are indexed in the repository (see Sections 1 and 2).

In the next step, digitized fragments of a cultural heritage object are imported into the system. Optionally preprocessing (e.g., denoising or adaptive reduction of the mesh resolution) as well as basic repair of other simple defects (as e.g., gaps and holes resulting out of the digitization) could be performed.

In step five, the fragments are reassembled by complementary surface matching (see Part B). It is to be expected that not all fragments of the object could be recovered and that some parts of the object are still missing.

In step six, the partially reassembled model might be analyzed for breaking edges. This could be driven by the intermediate results that were computed during the fragment reassembly in the previous step or other detection methods such as e.g., a partial intrinsic symmetry based detection [89]. Alternatively the user might also provide direct input to explicitly select or exclude certain surface areas.

In step seven, the object is segmented either automatically or again according to user input. This could be later on exploited for segment-wise similarity search and interchange of segments with similar

[92] E.g., see http://meshrepair.org

segments from the model repository; however it is critical that the method used for segmentation would be robust to all kinds of defects, especially missing large parts and fissures.

In the following step, interest point detection and feature extraction is performed on the partially reassembled object and its segments (see Section 2).

In step nine, the extracted features are used for similarity search, either to find matching segments in the repository or similar models with respect to the identified areas of interest of the object. Features that are not located in areas of interest (see step six) are not considered for determining similarity.

In step ten, a list of similar objects or matching segments could be used for automated approaches as e.g., [85, 88, 86, 84] or user driven assembly as in [79, 81, 82].

In the final step, minor defects could be corrected by the aid of partially user-driven inpainting (mesh completion) algorithms as described in Section 5.1.

6.2. Evaluation

During our research on the state of the art for 3D object repair approaches, we found that there is not yet an established, commonly used methodology for the evaluation and comparison of the repair or reassembly approaches. In most sources only a partial or unstructured evaluation is performed on certain aspects while others are left out. Often the proposed methods have been evaluated only on models of a specific type or certain resolutions. Hence there is little indication of how these might perform with digitized, high-resolution cultural heritage objects. Furthermore, many of the proposed complex repair methods (e.g., those mentioned in Section 5.2) not just depend on the characteristics of the fragmented input model but also on the quality and quantity of the models in the repository.

We conclude, that for the methods developed within WP4, we could develop a more generic, multi-faceted evaluation approach that could serve as a basis to quickly assess and compare several methods in context of a certain use case. The following subsections enumerate the various aspects that seem to be of importance.

Generality of the Approach

As the range of cultural heritage object types might be broad, the value of the contribution of a method also largely depends on its applicability to different object types.

However almost all methods are proposed in a certain application context and often include various specific tweaks and heuristics. For example, partial intrinsic symmetry based methods will not work if the objects to be repaired are not symmetric at all or if defects destroy large parts of the partial intrinsic symmetries for individual parts or entire objects. Other methods for meaningful and robust segmentation might also fail, however if segmentation is provided through user input it might significantly differ across the models in the repository due to different inclinations and preferences of the individual users. Other examples of limited scope of applicability can be observed in the Skull assembly method by Wei et al. [85] where input fragments must be scaled to a logical size so that their combined size after assembly would roughly match the logical size of the fixed template model. If this is not the case, the fragments cannot be matched successfully to the template [93]. In that case, the algorithm would have to be adapted for a dynamic set of templates. Physical size must either be known in advance or the same sensor resolution and post-processing (i.e., scaling) setup must be used for all digitized and the input fragments would have to be scaled accordingly.

Amount of Required User Input

Relying on user input often increases effectiveness of a particular task and is a flexible way to indirectly support the exploitation of very specific domain knowledge for object repair and assembly. Yet, for certain complex and ill-posed problems, it is in many cases the only way to address them. However, it might negatively impact the applicability of an approach. Besides differences in

[93] This is due to the way the size and similarity of the spin images is determined for template and fragments.

individual, user-specific inclinations and preferences that lead to problems when multiple users are involved, gathering user input slows down the repair workflow and requires users with a certain level of expertise. For large volumes of object models, the efficiency of a mostly user-driven repair method can be expected to be problematic.

Plausibility

Establishing a general and mathematically exact definition of plausibility of object repair results can be regarded as an inherently ill-posed problem. In some repair method proposals, an explicit formal notion of plausibility is established. Hence its mathematical nature it is even directly optimized in the repair algorithm (see Kawai et al. [78]) to compute the solution. It is obvious that using such a definition for evaluation of plausibility results in circular reasoning.

Kalogerakis et al. [84] conducted simple user studies, where a group of voluntary users directly rated the plausibility of generated models versus initial models in a double-blind test. In context of PRESIOUS a similar user study could be performed with experts from the cultural heritage domain.

However this approach has its limits when applied to methods that heavily rely on user input for critical repair decisions since individual preferences and inclinations have already been used to establish the result and then get evaluated by individual considerations of either the same or other users again. Thus, the evaluation might, in this case, eventually reflect the inter-individual variance of expert opinion. For repair methods that heavily rely on user-input, a survey of subjective usability of the system would be a more appropriate choice.

Robustness

The repair of digitized cultural heritage objects is affected by various kinds of defects (see Section 3). For ideally robust methods, the repair result should not be affected by any defects at all. This could be addressed by transforming input fragments to simulate various kinds and intensities of defects and comparing the respective repair results for invariance. For this we could possibly also make use of erosion simulation (see also report D3.1).

Efficiency

All introduced complex repair methods were at most evaluated with only specific model resolutions and repository sizes. However digitized models within PRESIOUS are acquired in high resolution. There is often little to no indication how the methods will scale concerning run time and memory consumption if resolution and the size of the repository is increased. The efficiency of the methods might indirectly affect the effectiveness of the methods. Large repositories and higher resolution models in principle provide more information that could be used for the computation of the result, if the methods scale well enough to stay usable with the increased data volume. For evaluation, run time and memory consumption measurements could be performed with varying model resolutions and repository sizes.

Appendix of Part A: Additional References to Mesh Repair

The following is a list of references from [50] as discussed in Sections 4 and 5.1. For conciseness of the report, we list these for reference in this appendix.

[Allen et al. 2003] ALLEN, B., CURLESS, B., AND POPOVIC, Z. 2003. The space of human body shapes: Reconstruction and parameterization from range scans. ACM Trans. Graph. 22, 3, 587-594.

[Andujar et al. 2002] ANDUJAR, C., BRUNET, P., AND AYALA, D. 2002. Topology-Reducing surface simplification using a discrete solid representation. ACM Trans. Graph. 21, 2, 88-105.

[Attene 2010] ATTENE, M. 2010. A lightweight approach to repairing digitized polygon meshes. Vis. Comput. 26, 11, 1393-1406.

[Attene and Falcidieno 2006] ATTENE, M. AND FALCIDIENO, B. 2006. ReMESH: An interactive environment to edit and repair triangle meshes. In Proceedings of the International Conference on Shape Modeling and Applications. 271-276.

[Attene et al. 2005] ATTENE, M., FALCIDIENO, B., ROSSIGNAC, J., AND SPAGNUOLO, M. 2005. Sharpen and bend: Recovering curved sharp edges in triangle meshes produced by feature-insensitive sampling. IEEE Trans. Visual Comput. Graph. 11, 2, 181-192.

[Attene et al. 2009] ATTENE, M., GIORGI, D., FERRI, M., AND FALCIDIENO, B. 2009. On converting sets of tetrahedra to combinatorial and pl manifolds. Comput. Aided Geom. Des. 26, 8, 850-864.

[Bac et al. 2008] BAC, A., TRAN, N.-V., AND DANIEL, M. 2008. A multistep approach to restoration of locally undersampled meshes. In Proceedings of the 5th International Conference on Advances in Geometric Modeling and Processing. 272-289.

[Barequet and Kumar 1997] BAREQUET, G. AND KUMAR, S. 1997. Repairing cad models. In Proceedings of the 8th Conference on Visualization (VIS ’97). IEEE Computer Society Press, 363-370.

[Barequet and Sharir 1995] BAREQUET, G. AND SHARIR, M. 1995. Filling gaps in the boundary of a polyhedron. Comput. Aided Geom. Des. 12, 2, 207-229.

[Bendels et al. 2005] BENDELS, G. H., SCHNABEL, R., AND KLEIN, R. 2005. Detail-Preserving surface inpainting. In Proceedings of the 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST). Eurographics Association, 41-48.

[Bischoff and Kobbelt 2005] BISCHOFF, S. AND KOBBELT, L. 2005. Structure preserving cad model repair. Comput. Graph. Forum 24, 3, 527-536.

[Bischoff et al. 2005] BISCHOFF, S., PAVIC, D., AND KOBBELT, L. 2005. Automatic restoration of polygon models. ACM Trans. Graph. 24, 4, 1332-1352.

[Blanz et al. 2004] BLANZ, V., MEHL, A., VETTER, T., AND SEIDEL, H.-P. 2004. A statistical method for robust 3D surface reconstruction from sparse data. In Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT ’04). 293-300.

[Boehn and Wozny 1992] BOHN, J. H. AND WOZNY, M. J. 1992. A topology-based approach for shell-closure. In Selected and Expanded Papers from the IFIP TC5/WG5.2 Working Conference on Geometric Modeling for Product Realization. North-Holland Publishing Co., Amsterdam, The Netherlands, 297-319.

[Borodin et al. 2002] BORODIN, P., NOVOTNI, M., AND KLEIN, R. 2002. Progressive gap closing for mesh repairing. In Advances in Modeling, Animation and Rendering. Springer, 201-213.

[Botsch and Kobbelt 2001] BOTSCH, M. AND KOBBELT, L. 2001. A robust procedure to eliminate degenerate faces from triangle meshes. In Proceedings of the Conference on Vision, Modeling and Visualization. 283-290.

[Branch et al. 2006] BRANCH, J., PRIETO, F., AND BOULANGER, P. 2006. Automatic hole-filling of triangular meshes using local radial basis function. In Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT’06). IEEE Computer Society, 727-734.

[Breckon and Fisher 2005] BRECKON, T. P. AND FISHER, R. B. 2005. Non-Parametric 3D surface completion. In Proceedings of the 5th International Conference on 3-D Digital Imaging and Modeling. IEEE Computer Society, 573-580.

[Brunton et al. 2010] BRUNTON, A., WUHRER, S., SHU, C., BOSE, P., AND DEMAINE, E. 2010. Filling holes in triangular meshes using digital images by curve unfolding. Int. J. Shape Model. 16, 1-2, 151-171.

[Campen and Kobbelt 2010] CAMPEN, M. AND KOBBELT, L. 2010. Exact and robust (self-)intersections for polygonal meshes. Comput. Graph. Forum 29, 2, 397-406.

[Chen and Cheng 2008] CHEN, C.-Y. AND CHENG, K.-Y. 2008. A sharpness dependent filter for recovering sharp features in repaired 3D mesh models. IEEE Trans. Visual Comp. Graph. 14, 1, 200-212.

[Curless and Levoy 1996] CURLESS, B. AND LEVOY, M. 1996. A volumetric method for building complex models from range images. In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96). ACM Press, New York, 303-312.

[Davis et al. 2002] DAVIS, J.,MARSCHNER, S. R., GARR, M., AND LEVOY, M. 2002. Filling holes in complex surfaces using volumetric diffusion. In Proceedings of the 1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT '02). 428-438.

[De Floriani et al. 2003] DE FLORIANI, L., MORANDO, F., AND PUPPO, E. 2003. Representation of non-manifold objects through decomposition into nearly manifold parts. In Proceedings of the 8th ACM Symposium on Solid Modeling and Applications. 304-309.

[Fan et al. 2010] FAN, H., YU, Y., AND PENG, Q. 2010. Robust feature-preserving mesh denoising based on consistent subneighborhoods. IEEE Trans. Visual Comput. Graph. 16, 2, 312-324.

[Fischl et al. 2001] FISCHL, B., LIU, A., AND DALE, A. M. 2001. Automated manifold surgery: Constructing geometrically accurate and topologically correct models of the human cerebral cortex. IEEE Trans. Med. Imag. 20, 1, 70-80.

[Fleishman et al. 2003] FLEISHMAN, S., DRORI, I., AND COHEN-OR, D. 2003. Bilateral mesh denoising. ACM Trans. Graph. 22, 3, 950-953.

[Furukawa et al. 2007] FURUKAWA, R., ITANO, T.,MORISAKA, A., AND KAWASAKI, H. 2007. Improved space carving method for merging and interpolating multiple range images using information of light sources of active stereo. In Proceedings of the 8th Asian Conference on Computer Vision (ACCV'07). 206-216.

[Granados et al. 2003] GRANADOS, M.,HACHENBERGER, P.,HERT, S., KETTNER, L.,MEHLHORN, K., AND SEEL, M. 2003. Boolean operations on 3D selective nef complexes: Data structure, algorithms, and implementation. In Proceedings of the 11th European Symposium on Algorithms (ESA '03). 654-666.

[Gueziec et al. 2001] GUEZIEC, A., TAUBIN, G., LAZARUS, F., AND HORN, B. 2001. Cutting and stitching: Converting sets of polygons to manifold surfaces. IEEE Trans. Visual Comput. Graph. 7, 2, 136-151.

[Guo et al. 2006] GUO, T.-Q., LI, J.-J., WENG, J.-G., AND ZHUANG, Y.-T. 2006. Filling holes in complex surfaces using oriented voxel diffusion. In Proceedings of the International Conference on Machine Learning and Cybernetics. 4370-4375.

[Guskov and Wood 2001] GUSKOV, I. AND WOOD, Z. 2001. Topological noise removal. In Proceedings of Graphics Interface. 19-26.

[Han et al. 2002] HAN, X., XU, C., BRAGA-NETO, U., AND PRINCE, J. L. 2002. Topology correction in brain cortex segmentation using a multiscale, graph-based algorithm. IEEE Trans. Med. Imag. 21, 2, 109-121.

[Hetroy et al. 2011] HETROY, F., REY, S., ANDUJAR, C., BRUNET, P., AND VINACUA, A. 2011. Mesh repair with user-friendly topology control. Comput. Aided Des. 43, 101-113.

[Hildebrandt and Polthier 2004] HILDEBRANDT, K. AND POLTHIER, K. 2004. Anisotropic filtering of non-linear surface features. Comput. Graph. Forum 23, 391-400.

[Hornung and Kobbelt 2006] HORNUNG, A. ANDKOBBELT, L. 2006. Robust reconstruction ofwatertight 3D models from nonuniformly sampled point clouds without normal information. In Proceedings of the Eurographics Symposium on Geometry Processing. 41-50.

[Jia and Tang 2004] JIA, J. AND TANG, C. -K. 2004. Inference of segmented color and texture description by tensor voting. IEEE Trans. Pattern Anal. Mach. Intell. 26, 6, 771-786.

[Jones et al. 2003] JONES, T. R., DURAND, F., AND DESBRUN, M. 2003. Non-Iterative, feature-preserving mesh smoothing. ACM Trans. Graph. 22, 943-949.

[Ju 2004] JU, T. 2004. Robust repair of polygonal models. ACM Trans. Graph. 23, 3, 888-895.

[Ju et al. 2007] JU, T., ZHOU, Q.-Y., AND HU, S.-M. 2007. Editing the topology of 3D models by sketching. ACM Trans. Graph. 26, 3, 42-1-42-9.

[Kaehler et al. 2002] KAHLER, K., HABER, J., YAMAUCHI, H., AND SEIDEL, H.-P. 2002. Head Shop: generating animated head models with anatomical structure. In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'02). 55-63.

[Kobbelt and Botsch 2003] KOBBELT, L. AND BOTSCH, M. 2003. Feature sensitive mesh processing. In Proceedings of the 19thSpring Conference on Computer Graphics (SCCG'03). 17-22.

[Kraevoy and Sheffer 2005] KRAEVOY, V. AND SHEFFER, A. 2005. Template-based mesh completion. In Proceedings of the Eurographics Symposium on Geometry Processing. 13-22.

[Kumar et al. 2007] KUMAR, A., SHIH, A. M., ITO, Y., ROSS, D. H., AND SONI, B. K. 2007. A hole-filling algorithm using non-uniform rational b-splines. In Proceedings of the 16thInternational Meshing Roundtable. 169-182.

[Levy 2003] LEVY, B. 2003. Dual domain extrapolation. ACM Trans. Graph. 22, 3, 364-369.

[Liepa 2003] LIEPA, P. 2003. Filling holes in meshes. In Proceedings of the Eurographics Symposium on Geometry Processing. 200-205.

[Maekelae and Dolenc 1993] MAKELA, I. AND DOLENC, A. 1993. Some efficient procedures for correcting triangulated models. In Proceedings of the Symposium on Solid Freeform Fabrication. 126-134.

[Masuda 2004] MASUDA, T. 2004. Filling the signed distance field by fitting local quadrics. In Proceedings of the 2nd 3D Data Processing, Visualization, and Transmission International Symposium. IEEE Computer Society, 1003-1010.

[Murali and Funkhouser 1997] MURALI, T. AND FUNKHOUSER, T. 1997. Consistent solid and boundary representations from arbitrary polygonal data. In Proceedings of the Symposium on Interactive 3D Graphics. 155-162.

[Nguyen et al. 2005] NGUYEN, M. X., YUAN, X., AND CHEN, B. 2005. Geometry completion and detail generation by texture synthesis. Visual Comput. 21, 8-10, 669-678.

[Nooruddin and Turk 2003] NOORUDDIN, F. AND TURK, G. 2003. Simpli cation and repair of polygonal models using volumetric techniques. ACM Trans. Vis. Comput. Graph. 9, 2, 191-205.

[Oomes et al. 1997] OOMES, S., SNOEREN, P., AND DIJKSTRA, T. 1997. 3D shape representation: Transforming polygons into voxels. In Proceedings of the 1st International Conference on Scale-Space Theory in Computer Vision. 349-352.

[Park et al. 2006] PARK, S., GUO, X., SHIN, H., AND QIN, H. 2006. Surface completion for shape and appearance.Visual Comput. 22, 3, 168-180.

[Patel et al. 2005] PATEL, P. S., MARCUM, D. L., AND REMOTIGUE, M. G. 2005. Stitching and filling: Creating conformal faceted geometry. In Procs of 14th International Meshing Roundtable. 239-256.

[Pauly et al. 2005] PAULY, M., MITRA, N. J., GIESEN, J., GROSS, M., AND GUIBAS, L. J. 2005. Example-Based 3D scan completion. In Proceedings of the Eurographics Symposium on Geometry Processing. 23-32.

[Pernot et al. 2006] PERNOT, J. P., MORARU, G., AND VERON, P. 2006. Filling holes in meshes using a mechanical model to simulate the curvature variation minimization. Comput. Graph. 30, 6, 892-902.

[Pfeifle and Seidel 1996] PFEIFLE, R. AND SEIDEL, H.-P. 1996. Triangular b-splines for blending and filling of polygonal holes. In Proceedings of the Conference on Graphics Interface (GI '96). Canadian Information Processing Society, 186-193.

[Podolak and Rusinkiewicz 2005] PODOLAK, J. AND RUSINKIEWICZ, S. 2005. Atomic volumes for mesh completion. In Eurographics Symposium on Geometry Processing. 33-42.

[Rock and Wozny 1992] ROCK, S. AND WOZNY, M. J. 1992. Generating topological information from a bucket of facets. In Proceedings of the Solid Freeform Fabrication Symposium. 251-259.

[Rossignac and Cardoze 1999] ROSSIGNAC, J. AND CARDOZE, D. 1999. Matchmaker: Manifold breps for non-manifold r-sets. In Proceedings of the 5th ACM Symposium on Solid Modeling and Applications. 31-41.

[Roth and Wibowoo 1997] ROTH, G. AND WIBOWOO, E. 1997. An efficient volumetric method for building closed triangular meshes from 3-d image and point data. In Proceedings of the Conference on Graphics Interface. Canadian Information Processing Society, 173-180.

[Sagawa and Ikeuchi 2008] SAGAWA, R. AND IKEUCHI, K. 2008. Hole filling of a 3D model by flipping signs of a signed distance field in adaptive resolution. IEEE Trans. Pattern Anal. Mach. Intell. 30, 4, 686-699.

[Sharf et al 2007] SHARF, A., LEWINER, T., SHKLARSKI, G., TOLEDO, S., AND COHEN-OR, D. 2007. Interactive topology-aware surface reconstruction. ACM Trans. Graph. 26, 3, 43-1 43-9.

[Sharf et al. 2004] SHARF, A., ALEXA, M., AND COHEN-OR, D. 2004. Context-based surface completion. ACM Trans. Graph. 23, 3, 878-887.

[Shattuck and Lehay 2001] SHATTUCK, D. W. AND LEHAY, R. M. 2001. Automated graph based analysis and correction of cortical volume topology. IEEE Trans. Med. Imag. 20, 11, 1167-1177.

[Shen et al 2004] SHEN, C., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. ACM Trans. Graph. 23, 3, 896-904.

[Sheng and Meier 1995] SHENG, X. AND MEIER, I. R. 1995. Generating topological structures for surface modelsIEEE Comput. Graph. Appl. 15, 6, 35-41.

[Spillmann et al. 2006] SPILLMANN, J.,WAGNER, M., AND TESCHNER, M. 2006. Robust tetrahedral meshing of triangle soups. In Proceedings of the International Workshop on Vision, Modeling, and Visualization (VMV). 9-16.

[Szymczak and Vanderhyde 2003] SZYMCZAK, A. AND VANDERHYDE, J. 2003. Extraction of topologically simple isosurfaces from volume datasets. In Proceedings of the IEEE Visualization Conference. 67-74.

[Taubin 1995] TAUBIN, G. 1995. A signal processing approach to fair surface design. In Proceedings of the SIGGRAPH 22nd Annual Conference on Computer Graphics and Interactive Techniques. ACM Press, New York, 351-358.

[Tekumalla and Cohen 2004] TEKUMALLA, L. S. AND COHEN, E. 2004. A holefilling algorithm for triangular meshes. Tech. rep., School of Computing, University of Utah.

[Turk and Levoy 1994] TURK, G. AND LEVOY, M. 1994. Zippered polygon meshes from range images. In Proceedings of the SIGGRAPH 21st Annual Conference on Computer Graphics and Interactive Techniques. ACM Press, New York, 311- 318.

[Varnuska et al. 2005] VARNUSKA, M., PARUS, J., AND KOLINGEROVA, I. 2005. Simple holes triangulation in surface reconstruction. In Proceedings of Algorithmy. 280-289.

[Verdera et al 2003] VERDERA, J., CASELLES, V., BERTALMIO, M., AND SAPIRO, G. 2003. Inpainting surface holes. In Proceedings of the International Conference on Image Processing. 903-906.

[Wagner et al. 2003] WAGNER, M., LABSIK, U., AND GREINER, G. 2003. Repairing non-manifold triangle meshes using simulated annealing. In Proceedings of the 4th Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics. 88-93.

[Wang and Oliveira 2007] WANG, J. AND OLIVEIRA, M. M. 2007. Filling holes on locally smooth surfaces reconstructed from point clouds. Image Visual Comput. 25, 1, 103-113.

[Wei et al. 2010] WEI, M., WU, J., AND PANG, M. 2010. An integrated approach to filling holes in meshes. In Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence. IEEE Computer Society, 306-310.

[Wood et al. 2004] WOOD, Z., HOPPE, H., DESBRUN, M., AND SCHROEDER, P. 2004. Removing excess topology from isosurfaces. ACM Trans. Graph. 23, 2, 190-208.

[Xiao et al. 2007] XIAO, C., ZHENG, W., MIAO, Y., ZHAO, Y., AND PENG, Q. 2007. A uni ed method for appearance and geometry completion of point set surfaces. Visual Comput. 23, 6, 433-443.

[Xu et al. 2006] XU, S., GEORGHIADES, A.,RUSHMEIER, H., DORSEY, J., ANDMCMILLAN, L. 2006. Image guided geometry inference. In Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT '06). IEEE Computer Society, 310-317.

[Zhou et al. 2007] ZHOU, Q., JU, T., ANDHU, S. 2007. Topology repair of solid models using skeletons. IEEE Trans. Visual. Comput. Graph. 13, 4, 675-685.

Part B

Fragmented Object Reassembly

7. REASSEMBLY OF OBJECTS FROM FRAGMENTS

In this part of the report we focus on the reassembly of fragmented 3D objects. We first formalize the problem statement for the most general case and then we discuss some special cases of the problem that have been investigated in the bibliography. Subsequently, we investigate the computational complexity of the problem and we present the general solving strategy for this family of problems. Finally, we provide a detailed overview of the previous work that has been proposed in the bibliography and is closely related to our research.

Since the task of computing the reassembly of a set of fragmented 3D objects involves solving other related problems, such as the segmentation and registration of objects and surfaces, our overview of the previous work includes an investigation of these problems, too. Because many of these problems are computationally intensive, an obvious research direction is to take advantage of GPUs and similar massively parallel architectures in order to minimize the computation times. For this reason, we have also included in this report a detailed overview of these architectures and we highlight the general principles that algorithms should follow in order to better take advantage of former.

7.1. Problem Statement

Reassembling objects from their parts is a problem that has been mainly addressed in the scope of computational archaeology, but the application domains of forensics and computer-assisted surgery have also expressed interest in this field and perform related research. In computational archaeology the problem is described as the automatic process of identifying the fragmented parts/regions of an object, the search for corresponding pieces within the set of given fragments and finally the matching of the parts that result in a virtual representation of (partially) reassembled objects. In the general case, the problem has $2 \ldots N$ input part representations (surfaces, volumes, point-cloud representations etc.) expressed in their local coordinate system and the solution to the problem is the output of $1 \ldots M$ assemblies with $2 \ldots N$ sets of geometric pose transformations.

Depending on the type of the fragments, the general reassembly problem can be divided into the following categories:

- **Jigsaw Puzzles**: In this genre of reassembly problems, the fragments and the final reassembled objects are considered to be of a known geometric shape (ex. rectangles). The problem is solved usually in two dimensions.
- **Two-Dimensional Reassembly**: The description of this problem is more generic compared to that of the jigsaw puzzles, as fragments and the final reassembled object are irregularly shaped (two-dimensional (2D) free form). Although in the real world, 2D objects do not exist, several flat objects, such as frescos, stone tables and pieces of torn documents, can be simplified and reduced to two-dimensions without this affecting the quality of the final reassembly.
- **Restricted Three-Dimensional (2.5D) Reassembly**: While methods in this category operate in three dimensions (3D), they rely on restrictions that render the methods unsuitable for more generic cases of free-form 3D shapes. The solutions in this category usually address *"thin-walled fragments"* such as potsherds and *"flat pieces with thickness"*, i.e. fresco fragments.
- **Three-Dimensional Reassembly**: This is the most generic case of the problem, which handles 3D free-form fragments, without making any assumptions on the shape of the fragments and the final reassembled object.

Later in this report we will present an overview of the proposed algorithms for each one of the above categories.

The most generic case of the reassembly problem, i.e. the 3D reassembly, is closely related to the problem of alignment and registration of partially overlapping surfaces. This problem often occurs when scanning 3D objects, where smaller parts of the object are digitized and the individual parts (partial scans) should be merged to form an incremental reconstruction of the original object. This is often performed using the *ICP algorithm* [94][95][96] or one of its many variants. Since this family of algorithms is widely used for the refinement of the alignment of fragment pairs in the context of fragmented object reassembly, we also review the state of the art of these algorithms in Section 8.

7.2. Computational Complexity

As discussed earlier, the problem of three-dimensional fragmented object reassembly is a general case of a 2D jigsaw puzzle solver. A general algorithm that solves the first problem should be able to also solve the second one. However, Demaine and Demaine [97] prove that the problem of *jigsaw puzzles* is *NP-Hard*, which means that the more general problem of fragmented object reassembly is also *NP-Hard.* The implication of this observation is that a known polynomial-time algorithm for this problem does not currently exist and it is also very difficult to find one, since it is equivalent to solving the *P=NP* problem. Therefore, all the methods proposed in the bibliography use *heuristic algorithms* or other approximations, in order to provide a solution within a reasonable execution time.

7.3. General Solving Strategy

The general methodology for solving a fragment puzzle, following the work of Huang et al. [98], consists of the following steps:

1. **Facet extraction and Labeling**: In this stage, a search across the entire surface of each fragment is performed. The goal is to identify boundaries of interest that will split the fragment's surface into facets. These facets are extracted and, based on the morphology of their surface, are classified as fragmented or intact. In the case of jigsaw puzzles, 2D reassembly and some 2.5D reassembly methods, this step is reduced to the extraction of the boundary of the fragments.
2. **Feature Extraction**: In order to perform the matching of the segmented facets, each method considers one or more distinct features that can characterize the surfaces and/or boundary lines. These features can vary from geometric descriptors, to color and shape features. Using these attributes, the faces are described in mathematical terms.
3. **Pairwise Matching**: In this step, a search for correct pairwise matches in terms of computed features and subsequent relative pose estimation for the (aligned) pieces is performed. In general, the search is performed on all pairs and for each one, a matching or mismatch metric in terms of feature compatibility is calculated. Several approaches try to minimize the set of

[94] Yang, Chen, and Gérard Medioni. "Object modelling by registration of multiple range images." Image and vision computing 10.3 (1992): 145-155.

[95] Besl, Paul J., and Neil D. McKay. "Method for registration of 3-D shapes." Robotics-DL tentative. International Society for Optics and Photonics, 1992.

[96] Zhang, Zhengyou. "Iterative point matching for registration of free-form curves and surfaces." International journal of computer vision 13.2 (1994): 119-152.

[97] Demaine, Erik D., and Martin L. Demaine. "Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity." Graphs and Combinatorics 23.1 (2007): 195-208.

[98] Huang, Qi-Xing, et al. "Reassembling fractured objects by geometric matching."*ACM Transactions on Graphics (TOG)*.Vol. 25.No. 3.ACM, 2006.

possible pairs (matching solutions), using either the local features or the classification of the faces.

4. **Multi-Piece Matching**: In this step the set of pairwise matches are merged and updated, using all combinations until all possible fragments are matched. The result of this stage is a set of aggregate objects representing – possibly partially – reassembled objects.
5. **Final Alignment**: Matching algorithms often give a rough alignment between the different fragments. In this case, the final alignment of the fragments is computed using a specialized alignment algorithm, such as ICP.

Section 2 of this report provides an overview of the ICP algorithm and its variants, Section 9 provides an overview of relevant facet extraction and data segmentation methods, Section 10 provides an overview of pair-wise matching methods and finally Section 11 provides an overview of the algorithms used for multi-piece matching. An interesting observation is that mixing and matching different algorithms from these categories and following the above methodology, new frameworks for the solution of the fragmented object reassembly can be constructed. However, our goal is also to advance the state-of-the art on each one of these categories by identifying the problems of the existing solutions and proposing insights and solutions to further improve their robustness and performance.

8. THE ICP AGORITHM AND ITS VARIANTS

Registration of two free-form surfaces has always been a computationally demanding procedure. Various methods have been employed to minimize the computational cost and increase the accuracy of the surface alignment at the same time, mainly taking advantage of the specific context of the problem (knowledge on the shape of the surfaces, initial pose transformations, etc.). However, the family of methods used more extensively is the *Iterative Closest Point* (*ICP*) algorithm and its variants.

The ICP algorithm, a relatively simple and elegant method, iteratively recalculates the transformation used to align two point sets, by attempting to minimize a distance metric. While the form of this metric varies amongst different methods, in most cases, a mean squared error function is preferred. In the next section, we attempt to establish the foundations for this family of techniques, by referencing the most important work done in the field in the last two decades. Subsequently, we proceed by demonstrating the evolution and the current trends in surface registration, through modern research and experiments.

8.1. The Besl-McKay method

The ICP algorithm, formulated by Paul Besl and Neil McKay [95], was the first method to address the problem of approximating an optimal transformation that aligns two general point sets. Most of preexisting work in the literature was dealing with specific classes of shapes, such as polyhedral models, piece-wise super-quadratic models and point sets with known correspondence. The presented algorithm converges monotonically to a local minimum of a mean-square distance metric, resulting in a satisfactory geometric transformation (rotation followed by translation) that aligns a point set to a given model shape. Their work implies that the model shape can be of arbitrary form and representation, whereas the data shape to be aligned with must be sampled as a point set.

Let us define a model shape X and a point set P to be aligned with the model X. In every iteration, a point set Y, subset of the model shape X is calculated, such that the points in Y are the result of a "closest point calculating operation" between every point in P and the model X. Let N_p denote the number of points in the point set and N_x denote the number of supporting primitives in the model shape. The procedure that will be described below is a worst-case $O(N_p N_x)$ complexity algorithm.

An initial quaternion-based registration vector $q_0 = [1,0,0,0,0,0,0]$ consisting of both the rotational and translational part and an initial transformation of the point set equal to the original set of point locations $P_0 = P$ are given. The following steps (describing the k-th iteration) are repeated until convergence is achieved with a predetermined tolerance τ:

1. Point set Y_k is calculated by finding for each point in P_k, its closest point in X and putting it in Y_k. This step holds a worst-case $O(N_p N_x)$ complexity. An average case would hold a $O(N_p log N_x)$ complexity.
2. The registration vector q_k and the mean-square point-matching error d_k are computed. In general, the registration vector q is calculated using the unit quaternion vector q_R (describing the rotation) and the three-dimensional vector q_T (describing the translation) as follows:

   $$q = [q_R | q_T]^T$$

   The point-matching error is given by the following equation:

   $$d = \frac{1}{N_p} \sum_{i=1}^{N_p} \| y_i - \mathbf{R}(q_R) p_i - q_T \|^2$$

   Where y_i are all the points in Y, p_i are all the points in P ($\equiv P_0$) and $\mathbf{R}(q_R)$ denotes the rotation matrix generated by the unit quaternion.

   The vectors q_R and q_T are calculated using the following SVD-based method. First, the centers of mass μ_P of P and μ_X of X are calculated:

$$\mu_P = \frac{1}{N_p}\sum_{i=1}^{N_p} p_i \qquad and \qquad \mu_X = \frac{1}{N_X}\sum_{i=1}^{N_X} x_i$$

The cross-covariance matrix $\Sigma_{P,X}$ of P and X is given by

$$\Sigma_{P,X} = \frac{1}{N_p}\sum_{i=1}^{N_p} (p_i\, {x_i}^T) - \mu_P\, {\mu_X}^T$$

The cyclic components of the anti-symmetric matrix $A_{ij} = (\Sigma_{P,X} - {\Sigma_{P,X}}^T)_{ij}$ are used to form the column vector $\Delta = [A_{23}\ \ A_{31}\ \ A_{12}]^T$. This vector is then used to form the symmetric 4x4 matrix $\mathbf{Q}(\Sigma_{P,X})$:

$$\mathbf{Q}(\Sigma_{P,X}) = \begin{bmatrix} tr(\Sigma_{P,X}) & \Delta^T \\ \Delta & \Sigma_{P,X} - {\Sigma_{P,X}}^T - tr(\Sigma_{P,X})\mathbf{I}_3 \end{bmatrix}$$

where $\mathbf{I}_3$ is the 3x3 identity matrix. The unit eigenvector corresponding to the maximum eigenvalue of $\mathbf{Q}(\Sigma_{P,X})$ is selected as the optimal rotation q_R. The optimal translation vector is calculated as:

$$q_T = \mu_X - \mathbf{R}(q_R)\mu_P$$

This step holds an $O(N_p)$ complexity in its general case.

3. The registration vector calculated in the previous step is applied on P_0, in order to get the next iteration's "intermediate" point set. So, $P_{k+1} = q_k(P_0)$. This holds $O(N_p)$ complexity.
4. The whole operation is terminated if the difference of the point matching errors $|d_k - d_{k-1}|$ falls below the preset tolerance τ.

This algorithm is proven to always converge to a translation and rotation that will align the point set P to the model X. Although this alignment might be not the optimal one, since the method is not guaranteed to converge to a global minimum, this is not regarded as a significant drawback. The main problem of the ICP algorithm is the fact that it fails to deal with statistical outliers. The methods that will be presented later in this report show variants of the algorithm that address this issue.

8.2. The Chen and Medioni Method

Chen and Medioni [94] concurrently proposed a similar approach to that of Besl and McKay. Their work was focused on aligning data from range images (i.e. images obtained using a range scanner), whereas the method by Besl and McKay [95] directly addresses the registration of 3D shapes, represented mainly by point clouds. Although the two approaches are similar, in the sense that both aim to minimize an objective error function by iteratively revising a transformation, there are a few key differences that need to be stressed.

First of all, the Chen-Medioni method assumes that an initial registration partially aligning the two available views (range data) exists. On the other hand, in the Besl-McKay algorithm, no such assumption is made. The Chen-Medioni method aims to supply a finer, more accurate transformation that aligns two views, given an initial approximation of it. Another interesting difference is that this method does not depend on closest point operations on all points of the shape, but only on a set of control points. The concept of control points will be explained in the presentation of the algorithm that follows.

Let P and Q be two surfaces and also let $\mathbf{T}^0$ be an initial transformation that provides a rough alignment of the two shapes. The goal of this approach is to approximate one of the two surfaces (let that be Q) using its tangent plane S_j at points q_j of Q, with $q_j = q|\min_{q\in Q}\|\mathbf{T}p_i - q\|$, where p_i are

control points chosen on P, after a transformation **T** is used on them. After this step, the objective is to minimize the distance between planes S_j and surface P at the corresponding points, as shown in *Figure 18:*

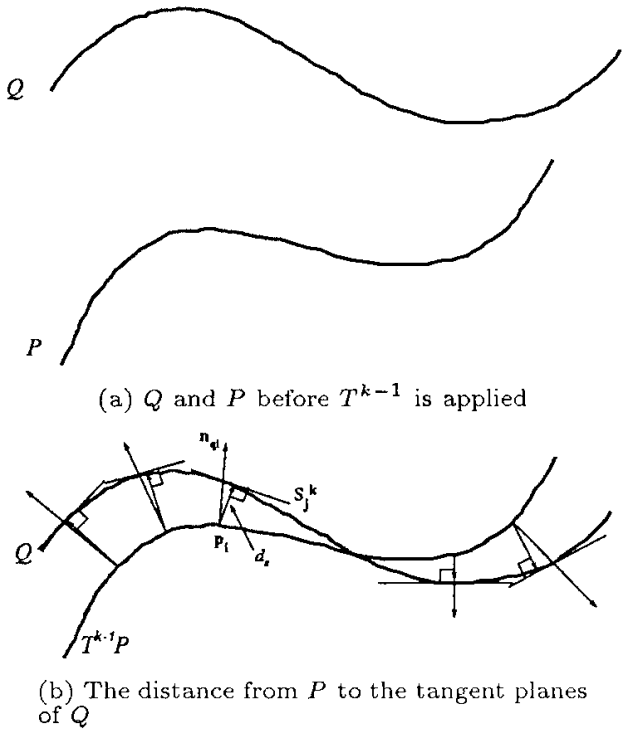


*Figure 18: Distance measures between **P** and **Q** illustrated in the 2D case. Figure taken from [**94**].*

In the *k*-th iteration, the error function that needs to be minimized is the following:

$$e^k = \sum_{i=1}^{N} d_S^2(\mathbf{T}\mathbf{T}^{k-1}p_i, S_i^k)$$

where $\mathbf{T}\mathbf{T}^{k-1} = \mathbf{T}^k$ and S_i^k is the tangent plane to Q at q_j^k, with q_j^k being the intersection of Q with line $\mathbf{T}^{k-1}l_i$. The lines l_i are the lines perpendicular to P at the control points p_i. Finally, d_s is the signed distance from a point to a plane. The registration algorithm tries to find and update the transformation **T** that minimizes e^k with a least squares method, as shown below:

1. A set of control points $p_i \in P$ $(i = 1 \dots N)$ is selected and the surface normals n_{pi} at p_i are computed. This method implies that control points do not need to represent meaningful surface features and can be picked over a regular grid, as the main purpose is to save computation time. On the other hand, they can be picked in relatively smooth areas, so that their correspondences on Q are more reliable and easier to find. More recent studies suggest that control points should be more densely distributed in areas with unique characteristics (e.g. abrupt gradient changes).
2. At each iteration *k*, the following steps are repeated until the process converges:
   For each control point p_i:
   - Apply $\mathbf{T}^{k-1}$ to both the control point p_i and the normal n_{pi} (rotational part), to obtain p'_i and n'_{pi}
   - Find the intersection q_i^k of surface Q with the normal line defined by p'_i and n'_{pi}
   - Compute the tangent plane S_i^k of Q at q_i^k

Find **T** that minimizes e^k with a least squares method and let $\mathbf{T}^k = \mathbf{T}\mathbf{T}^{k-1}$.

The algorithm stops when $\frac{\|e^k - e^{k-1}\|}{N'} \leq e_e$, where e_e is a user-defined threshold and N' is the actual number of control points used, as some p_i may not correspond to points on Q ($N' \leq N$). The whole process yields a complexity similar to that of Besl-McKay method, and does not deal with statistical outliers either.

8.3. *k*-d Trees and Outlier Handling: The Zhang Method

Accelerated variants of the ICP method retain more or less the general methodology of the original algorithm, although they incorporate variations in the most complex part of it, the closest-point calculation step. In this section, we will present how k-d trees are used in the work of Zhang [99], which also addresses the removal of outliers in a novel way.

A *k*-d tree is essentially a generalization of bisection in one dimension to *k* dimensions. Its use was proposed by Besl and McKay [95] but a full implementation was first presented by Zhang. Let P be the point cloud that needs to be registered. The *k*-d tree, with *k*=3 used in Zhang's method is constructed by choosing a plane parallel to yz −plane passing through a data point in P, thus cutting the whole space into two rectangular parallelepipeds, such that there are approximately equal numbers of points on either side of the cut. That way, a left and right "child" are obtained. Each of them is then split further by a plane parallel to xz −plane in the same manner as before, producing a left and a right "grandchild". This splitting continues with a plane parallel to xy −plane and so on, until a rectangular parallelepiped that contains no points is reached. The construction takes $O(NlogN)$ time and $O(N)$ storage (N is the total number of points in the cloud).

At this point, the method by Zhang introduces the concept of "maximum distance" D_{max}, which denotes the maximum tolerable distance that a point of P and its corresponding closest point in Q must have in order to be incorporated in the calculations for the optimal transformation. The method of choosing this tolerance is based on the statistical behavior of distances between corresponding points. In particular, let $\mathfrak{D}$ be a user-supplied parameter that indicates whether the registration of two frames is "good" or not. Let p_i be the points on P to be registered and q_i their corresponding closest points on Q (with d_i denoting the distance between them). The algorithm computes the mean μ and sample deviation σ of the distances d_i. The maximum tolerable distance in the I −th iteration D_{max}^I is calculated as follows:

- if $\mu < D$, then $D_{max}^I = \mu + 3\sigma$, in which case the registration is considered quite good.
- else if $\mu < 3\mathfrak{D}$, then $D_{max}^I = \mu + 2\sigma$, in which case the registration is still considered good.
- else if $\mu < 6\mathfrak{D}$, then $D_{max}^I = \mu + \sigma$, in which case the registration is passable.
- else $D_{max}^I = \xi$. ξ is the median of all distances. In this case, the registration is considered unacceptable.

In every iteration, a (p_i, q_i) pair is considered noisy and is removed when distance d_i is larger than D_{max}^I. This method crosses out statistical outliers by removing pairs that have been badly registered or falsely paired together.

The three-dimensional *k*-d tree structure is used in order to find a closest point (or a list of candidate closest points) to a given point x of P without having to search through the whole point set Q. D_{max}^I is used to decide whether the current node of the tree (which is a point) can be considered a "closest point" to x, or that the process needs to continue with one of the two sub-trees.

[99] Z. Zhang. Iterative point matching for registration of free-form curves. *International Journal of Computer Vision*, 1994

8.4. Efficient Variants and Optimization

To date, many fast variations to the main ICP method have been proposed. However, what is mainly used in applications (both commercial and non-commercial) is a standard ICP algorithm, usually with a *k*-d tree to speed up closest point search. Minor tweaks to the main method are also common, to better suit the context and the goals of specific applications.

Research in this field has been focused on two major aspects of the algorithm; obviously the first one is the closest point calculation, and the sampling and matching step in general. The other aspect is the convergence of the algorithm and the effectiveness of the error metric used. At this point, we shall recount parts of the work of Rusinkiewicz and Levoy [100], in which many of the trends of the past decade in the field of ICP-based registration are presented and evaluated.

An issue raised in this study is the effective selection of pairs of points between the two surfaces. Better correspondences could lead to faster and more accurate convergence and the authors argue that "closest-point" correspondence of all the points in the surfaces (as in Besl and McKay [95]) is not the safest way to that end. On the problem of point selection, they argue that a sampling of points (much like the Chen and Medioni algorithm [94]) may reduce the computation cost, though they recommend a normal-space sampling instead of a grid-like sampling. The justification for this choice is that the use of a grid may underestimate the presence of significant features that can aim the accuracy of the convergence process as shown in *Figure 19*.

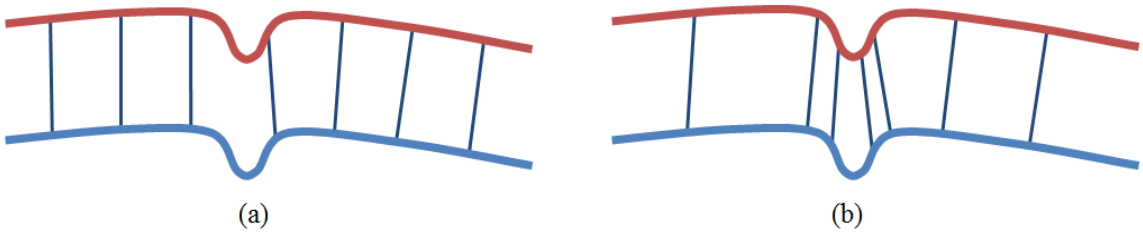


Figure 19: (a) Uniform sampling and (b) normal-space sampling. Using uniform sampling, sparse (and more robust) features may be overwhelmed by non-descript points. Figure taken from [**100**].

When creating pairs, the idea used originally by Besl and McKay was to match points of the first point set with their closest counterparts on the other set. This method, though stable, has slower convergence, because the correspondence may not be physically accurate. *Figure 20(a)* illustrates this claim. Chen and Medioni [94] presented their method of "shooting normals" from the surface P to surface Q, which can offer better results in relatively smooth surfaces with detail, than in complex or noisy meshes. Another method proposed is that of simply projecting point p in P to the coordinate space and point of view of Q (see *Figure 20(b)*). This idea shows worse performance per iteration (again, because of no physical correspondence between points, like "closest-point" techniques), but reduces this step's exponential complexity to a mere linear one. A way to refine this approach is to search the neighborhood around the projection location on Q using a distance metric in order to find a better correspondence. That is the reason why Rusinkiewicz and Levoy [100] recommend that an ICP registration algorithm, in order to be effective in real-time applications, should use this approach in creating correspondences. One should have in mind, though, that this study was finished more than ten years ago, before hardware acceleration and massive parallelization were a mainstream way of speeding up such computations.

[100] S. Rusinkiewicz and M. Levoy. Efficient Variants of the ICP Algorithm. *3rd Int. Conf. on 3D Digital Imaging and Modeling*, 2001

Another approach to speed up the basic ICP algorithm was presented by Jost and Hügli [101]. It was also focused on the exponentially complex part of closest-point computation. The proposed method is based on the assumption that there exists a neighborhood relationship between the two sets of points P and X. The relationship hypothesis is that two neighbors in a data set possess closest points that are also neighbors in the other data set. This is demonstrated in *Figure 21*.

The proposed idea towards a faster search is to look for good approximations in the neighborhood of a known correspondence instead of exact closest points. The neighborhood relationship is used to get a first approximation of the closest point and then, a local search refines the result. If that fails, an exhaustive search through the whole point set is performed. Apparently, for every p_i in the point set P, this leads to a process that either looks through the full set X or only inside the neighborhood of a known closest point x_k on X of a neighbor p_k of p_i.

Naturally, the above method would (in its worst case) perform a full closest-point search for every point in P, yielding a $O(N_p N_x)$ complexity. However, a best case scenario would make this method a $O(N_p)$ time algorithm (faster than a k-d tree implementation, which holds a best case complexity of $O(N_p log N_x)$). A combination of the two methods, i.e. a neighborhood search when known closest point is present and k-d tree-based search otherwise, seems a quite efficient approach.

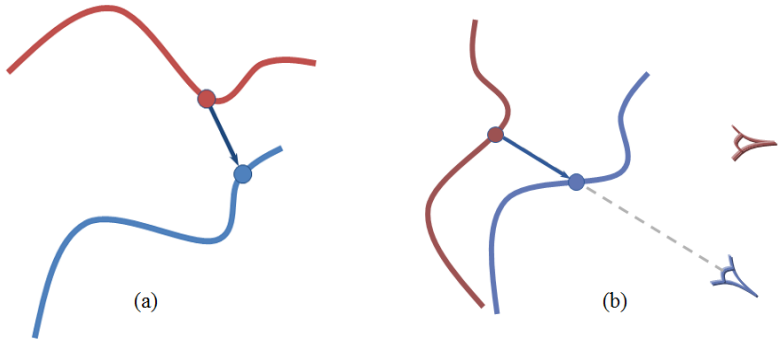
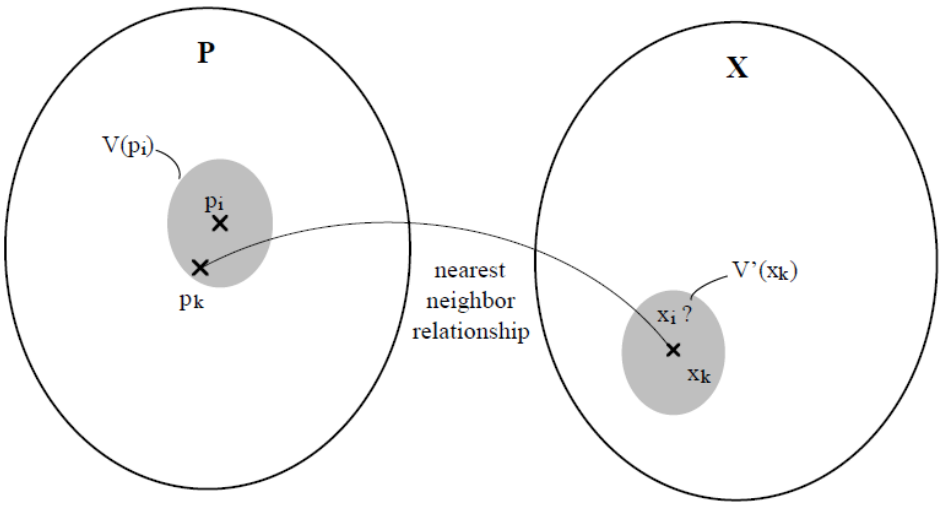


*Figure 20: (a) Closest-point correspondence and (b) projective correspondence. Figure taken from **[100]**.*



[101] T. Jost and H. Hügli. Fast ICP algorithms for shape registration. *Pattern Recognition (In: Lecture Notes in Computer Science)*, 2002

Figure 21: The neighborhood relationship assumption. If $\boldsymbol{p_i}$ has a neighbor $\boldsymbol{p_k}$ in data set $\boldsymbol{P}$ with a known closest point $\boldsymbol{x_k}$ in set $\boldsymbol{X}$, finding the closest point of every $\boldsymbol{p_i}$ in the neighborhood $\boldsymbol{V}(\boldsymbol{p_k})$ can be reduced to searching the closest point in the neighborhood $\boldsymbol{V'}(\boldsymbol{x_k})$ of $\boldsymbol{x_k}$. Figure taken from [101].

A different trend in ICP variants is expressed in Sharp et al. [102]. Their work focuses on optimizing the uncertain correspondence between points in the original closest-point concept by using a weighted linear combination of both positional and feature distances. As stated in their paper, the closest-point correspondence fails when the input point set is not approximately aligned with the model. In this case, shape descriptors could provide additional information to improve the correspondence search. Those descriptors must be invariant to rigid camera motion. Therefore, this study uses Euclidian invariants: curvature, moment invariants and spherical harmonics invariants.

In their proposed method, called ICPIF (ICP with invariant features) curvature is calculated by estimating the surface normal at each point and then differentiating. Moment invariants are calculated using a method described by Sadjadi and Hall [103], which computes second order moment invariants using centralized moments. This fixes the coordinate system center at the center of mass and achieves translation invariance. As a final step, the method proposed by Burel and Henocq [104] for deriving rotationally invariant features from spherical harmonics is used. *Figure 22* illustrates an example.

In particular, given two points p and q, this method computes a positional distance between them $d_e(p,q)$, as well as a feature distance $d_f(p,q)$ representing the feature difference between the calculated invariant features of points p and q. A combined positional and feature distance between p and q is calculated as follows:

$$d_\alpha(p,q) = d_e(p,q) + \alpha^2 d_f(p,q)$$

where α controls the contribution of the feature distance and is provided by the user. α^2 is shown to have an optimal value approximately equal to the mean squared distance from a point cloud location to its closest model point. Much like other methods, when α decreases this method is proven to converge monotonically to a local minimum. However, there is no need for a user-supplied initial estimation of the registration, which is a desired characteristic for closest-point correspondence techniques. As a final optimization, the authors propose a *k*-d tree implementation, similar to the one by Zhang [99], in order to speed up the point-correspondence search procedure.

An analysis by Tsamoura and Pitas [105] compares the ICPIF with two other ICP variants, namely the Robust ICP (RICP) [106] and the "trimmed ICP" and "picky ICP" algorithms [107]. RICP is a closest-point approach with an outlier handling feature, that works by rejecting pairs of corresponding points (let those points be p_i and m_j) with residual $s = \mathbf{R}p_i - m_j$ greater than a predefined threshold, where the transformation **R** is derived from the following equation:

$$\begin{pmatrix} m_{1_x} & m_{2_x} \\ m_{1_y} & m_{2_y} \end{pmatrix} = \mathbf{R} \begin{pmatrix} p_{1_x} & p_{2_x} \\ p_{1_y} & p_{2_y} \end{pmatrix}$$

[102] G. Sharp, S. Lee and D. Wehe. ICP Registration using Invariant Features. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2002

[103] F. Sadjadi and E. Hall. Three-dimensional moment invariants. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1980

[104] G. Burel and H. Henocq. Three-dimensional invariants and their application to object recognition. *Signal Processing*, 1995

[105] E. Tsamoura, I. Pitas. Automatic Color Based Reassembly of Fragmented Images and Paintings. *IEEE Trans. On Image Processing*, 2010

[106] E. Trucco, A. Fusiello, and V. Roberto. Robust motion and correspondencesof noisy 3D point sets with missing data. *Pattern Recognit. Lett.*, 1999

[107] D. Chetverikov,D. Svirko, D. Stepanov, and P. Krsek, The trimmed iterative closest point algorithm, *Proc. of 16th Int. Conf. Pattern Recognition*, 2002

R is calculated over a number of iterations and the one that minimizes the residual is chosen. On the other hand, the trimmed ICP and picky ICP methods reject corresponding pairs based on the distance between their points. The procedure is a typical closest-point approach that uses the trimmed subset of the initial set of correspondences.

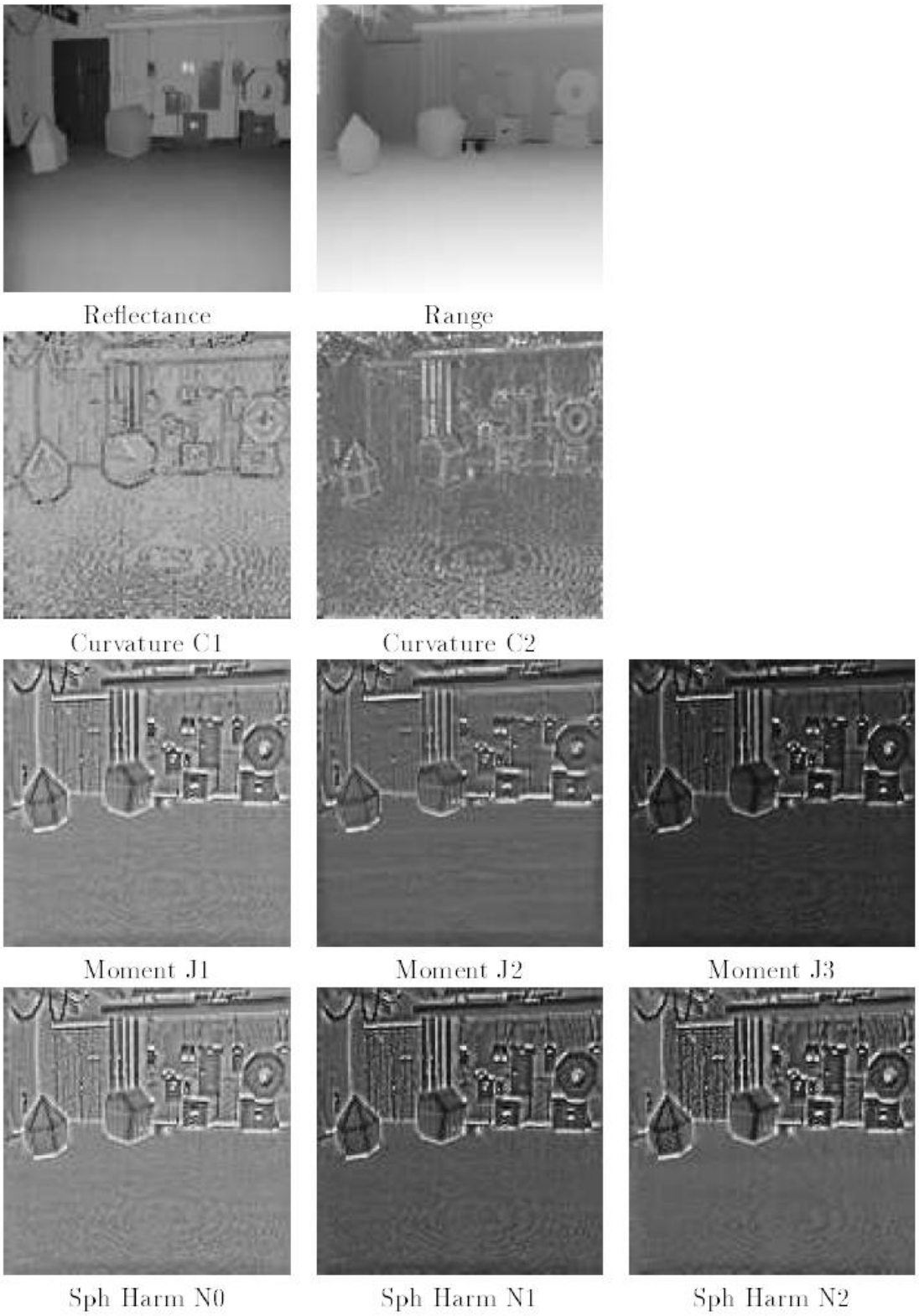


Figure 22: A range image and eight different variants.Figure taken from [102].

As stated in the analysis in [105], despite their good performance, RICP, trimmed ICP and picky ICP have the disadvantage that the outlier percentage in the input, must not exceed 20%. Otherwise, the computed transformation is wrong. Both these methods and ICPIF [102] were evaluated by aligning correctly matched contour segments of a reassembled image (details of the method may be found in [105]). *Table 10: Performance in correctly matched contour segment alignment. Data taken from [105]*.

compares the percentage of visually correctly identified alignments over correctly matched contour segments of the three methods, including the original ICP algorithm of Chen and Medioni [94]. It is shown that ICPIF outperforms the other approaches.

ICP variant	Performance %
ICP	45.71
Trimmed ICP	62.68
RICP	64.36
ICPIF	77.12

Table 10: Performance in correctly matched contour segment alignment. Data taken from [105].

Before moving to more recent endeavors, it would be useful to note that the method of Sharp et al. [102] summarizes a trend of approaches more suitable to the situations we will be dealing with.

Recent research is more concerned with improving the convergence rate of the ICP algorithm or in refining its results. We have already seen a technique that uses invariant features to describe surface behavior and thus provide more robust matching criteria. We have also encountered variants that endorse the point-to-plane method of Chen and Medioni [94], as well as various strategies on how to choose control points and create pairs. The approach that will be presented now is an attempt to incorporate both the "standard ICP" methods (Section 8.1) and the point-to-plane variants into a single probabilistic framework.

In the work of Segal et al. [108], a generalization of the ICP algorithm, which takes into account the locally planar structure of both scans in a probabilistic model, is proposed. The probabilistic variation is inserted in the registration update step, leaving the original algorithm intact, so that other variants and optimization techniques (such as *k*-d trees) can be still applicable. Following this paper's notation, given two point sets A and B, the update step of registration $\mathbf{T}$ of the standard ICP is

$$\mathbf{T} \leftarrow argmin_{\mathbf{T}} \left\{ \sum_i w_i \, \|\mathbf{T} b_i - m_i\|^2 \right\}$$

$$w_i = \begin{cases} 1 & \|\mathbf{T} b_i - m_i\| \leq d_{max} \\ 0 & otherwise \end{cases}$$

where b_i is every point in B, m_i is the closest point in A. On the other hand, the update step in a point-to-plane method is

$$\boldsymbol{T} \leftarrow argmin_{\mathbf{T}} \{ \sum_i w_i \, \|n_i(\mathbf{T} b_i - m_i)\|^2 \}$$

[108] A. V. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. *Robotics: Science and Systems*, 2009

where n_i is the surface normal at m_i. The authors show that these two assignments may be replaced by

$$\mathbf{T} \leftarrow argmin_{\mathbf{T}}\{\sum_i d_i^{(\mathbf{T})^T}(C_i^B + \mathbf{T}C_i^A\mathbf{T}^T)^{-1}d_i^{(\mathbf{T})}\}$$

where $d_i^{(\mathbf{T})}$ is the distance d_i between a_i and b_i, after $\mathbf{T}$ is applied on a_i and C_i^A, C_i^B are covariance matrices associated with the points a_i and b_i. This formula is a generalized version of the previous ones. It is proven in the study that, for certain values of C_i^A and C_i^B, we can obtain the update steps of the original algorithms.

The authors claim that point-to-plane ICP was an improved variant of the traditional ICP and, in that manner, they show that Generalized-ICP can be used to take into account surface information from both scans, thus formulating a "plane-to-plane" method. The study provides experimental results showing a significantly reduced error ratio in relationship with the maximum match distance d_{max}, compared to the one we get from the traditional ICP techniques, as shown in *Figure 23*.

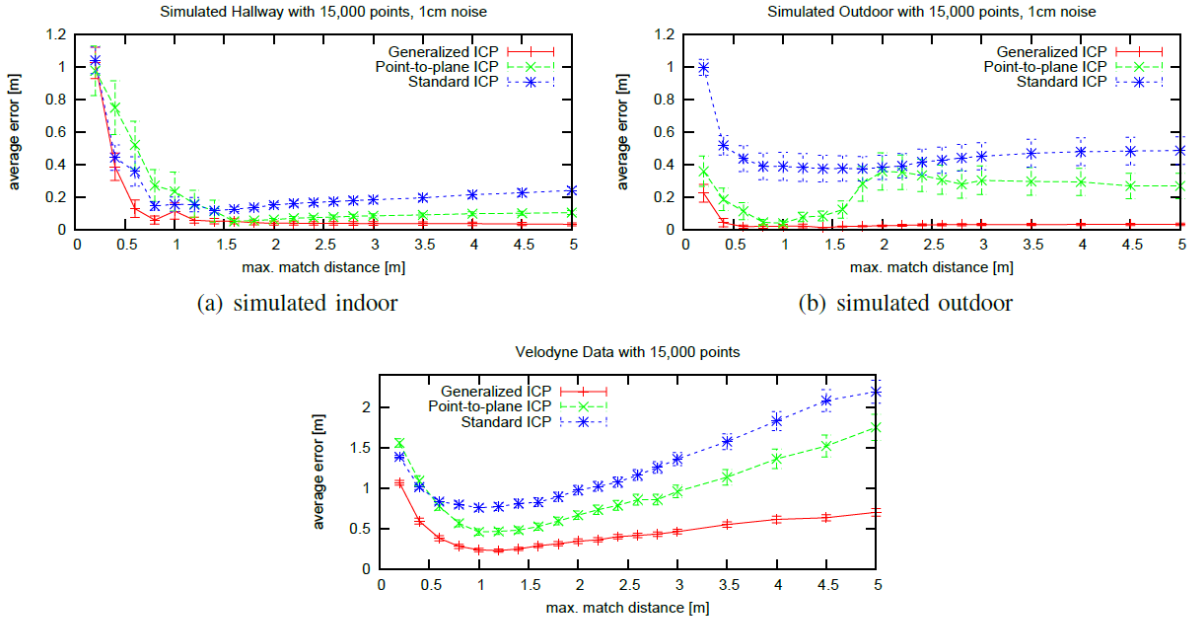


Figure 23: Average error as a function of $\boldsymbol{d_{max}}$*. Figure taken from [108].*

An interesting fact is that all the methods presented in this section and the previous ones assume that the points are observed with isotropic Gaussian noise. The work of Meinzer et al. [109] shows that this may not be the case and the Gaussian noise assumption may lead to errors. This study generalizes the algorithm and proposes a variant called A-ICP that accommodates for anisotropic and inhomogeneous localization error and proves the new method's guaranteed convergence.

According to the authors, the main problem of previous methods is that they do not utilize the covariance matrices of all input points in order to perform anisotropic inhomogeneous weighting. Given two points *x* and *y* (of point sets *X* and *Y* respectively), the covariance matrices $\boldsymbol{\Sigma}_x$ and $\boldsymbol{\Sigma}_y$ that represent their localization error and the two-space cross covariance $\boldsymbol{\Sigma}_{xy} = \boldsymbol{\Sigma}_x + \boldsymbol{\Sigma}_y$, the anisotropically weighted distance between x and y is

[109] H. Meinzer, M. Fangerau, M. Schmidt, J. M. Fitzpatrick, T. R. dos Santos, A. M. Franz and L. Maier-Hein. Convergent Iterative Closest-Point Algorithm to Accomodate Anisotropic and Inhomogenous Localization Error. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2012

$$d(x,y) = \left\|\mathbf{W}_{xy}(x-y)\right\|_2$$

with the weighting matrix $\mathbf{W}_{xy} = w\boldsymbol{\Sigma}_{xy}^{-\frac{1}{2}}$ and a normalization constant $w > 0$. In the case of zero-mean, isotropic noise, we have that $\boldsymbol{\Sigma}_x = \boldsymbol{\Sigma}_y = c\mathbf{I}$, for some scalar value c.

The original registration error formula of the ICP algorithm could be written as follows:

$$e = \sum_{i=1}^{N} \|\mathbf{R}x_i + t - z_i\|_2^2$$

where **R** is the rotation matrix, *t* is the translation vector and z_i is the best correspondence to x_i. The proposed weighted registration error is given by the following equation:

$$e_{weighted} = \sum_{i=1}^{N} \|\mathbf{W}_i(\mathbf{R}x_i + t - z_i)\|_2^2$$

with $\mathbf{W}_i$ being a function of the rotation matrix **R**, $\mathbf{W}_i = w(\mathbf{R}\boldsymbol{\Sigma}_{x_i}\mathbf{R}' + \boldsymbol{\Sigma}_{z_i})$.

A major drawback of this method is its inability to accommodate the use of *k*-d trees in order to speed up the quadratic complexity of establishing correspondences. To make the correspondence search more efficient, A-ICP is initialized with the (faster) original ICP to determine a good starting pose for the iterative refinement. Furthermore, the authors propose constraining the search of the weighted nearest neighbor to a certain radius r from the point of interest.

Figure 24 and *Figure 25* depict the comparison between the standard ICP variants and the A-ICP (initialized by the ICP) using both a Voronoi-based and Principal Components Analysis (PCA)-based method for the computation of the covariance matrices.

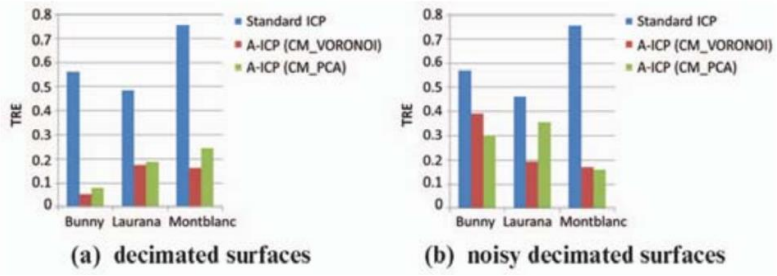


Figure 24: Results for whole surface registration. Error given in mm for the Bunny and Laurana and in m for Montblanc. Figure taken from [109].

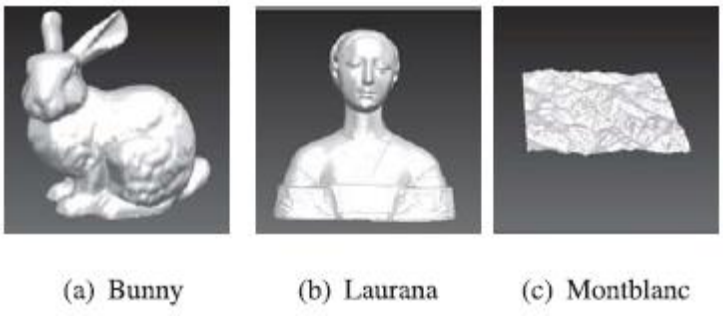


Figure 25: Meshes used in experiments.Figure taken from [109].

8.5. Conclusion

At this point, it may be clear that optimization and variations of the core ICP method have almost always targeted certain aspects of the algorithm. In terms of computational complexity, the common trend is to utilize *k*-d trees to speed up the correspondence search. In general, these classes of methods

have $O(N_p log N_x)$ complexity with respect to the size of the point clouds to be aligned. One should have in mind that the original ICP formulations (Besl and McKay [95], Chen and Medioni [94]) were both algorithms of exponential complexity. We have also presented the Jost and Hügli [101] algorithm that uses a nearest neighbor search that exploits the locality of already registered points to achieve linear complexity. This, however, is a best-case complexity (worst case is similar to any *k*-d tree-based implementation).

Speed of convergence is another optimization target and one might say that, of all the proposed methods, the one presented by Sharp et al. [102] holds the most interesting features, in the context of fragment matching and reassembly.

When scanning surfaces such as cultural heritage objects, it is natural to exploit salient features and spatial locality (neighbor search), in order to increase the efficiency of the registration algorithm. Methods that do not exploit these characteristics and depend solely on closest-point correspondence are expected to show slower convergence and higher error rate. Of course, any modern ICP variant will include a method to accelerate the correspondence search step, using either some acceleration structure, like *k*-d trees, or other means to reduce the search space.

Finally, it is noteworthy to mention that since in the context of final fractured surface alignment for object reassembly, the correspondence of surfaces has already been based on feature congruence on both parts, ICP relaxation methods can directly operate on the same descriptors and already evaluated correspondences of the partial match.

Table 11 provides a general overview and comparison of the algorithms that we have presented in this section.

Method		Type	Complexity	Comment
Besl & McKay	[95]	Closest-point	$O(N_p N_x)$	Every point in P is paired with its closest point in X.
Chen & Medioni	[94]	Point-to-plane	$O(N_p N_x)$	Point to plane distance. Control points.
Zhang	[99]	Closest-point	$O(N_p log N_x)$	*k*-d trees and statistical outlier handling.
Jost & Hügli	[101]	Closest-point	$O(N_p log N_x)$	Neighborhood search for good approx. localized correspondence.
Sharp et al.	[102]	Closest-point	$O(N_p log N_x)$	Invariant features to assist accuracy in correspondence search.
Segal et al.	[108]	Plane-to-plane	$O(N_p log N_x)$	Combination of closest-point and point-to-plane methods.
Meinzer et al.	[109]	Closest-point	$O(N_p N_x)$	Anisotropic localization error handling

Table 11. General overview of ICP-based surface alignment methods.

9. FACET EXTRACTION AND LABELING

As discussed in Section 7.3, the first step in many fractured object reassembly methods is to perform a *segmentation* of each fragment into smaller segments or facets. This segmentation in smaller and simpler facets is crucial for pair-wise object matching algorithms, because it is easier and more reliable to design algorithms that test the compatibility of two objects with simplistic topology. Furthermore, segmentation algorithms can facilitate the classification of object regions as *fractured* or *intact.* Any subsequent matching operations are performed only between fractured facets. Therefore, this classification of the facets in *fractured* or *intact* ones can potentially increase the efficiency of the reassembly methods, let aside provide additional features for constrained matching, such as fracture boundary curves.

For these reasons, data segmentation is a crucial part in many general 3D object reassembly methods and many of these methods begin with a segmentation of the initial three-dimensional objects in smaller parts. Other applications of 3D object segmentation include mesh simplification, data compression, surface parameterization, texture mapping (texture atlas generation), animation, collision detection, reverse engineering and more. Therefore, algorithms for geometric segmentation have been studied excessively in the bibliography.

In this section, we will provide an overview of the 3D object segmentation algorithms, since they are an essential step in 3D object reassembly. The remainder of this section is organized as follows. First we provide the mathematical formulation of the problem. We then describe the main two categories of 3D object segmentation algorithms and finally we provide a detailed overview of the general methodologies that have been proposed in the bibliography in order to solve this problem. At the end of this section, along with our conclusions, we provide a comparative table presenting the main characteristics of the previously discussed algorithms.

9.1. Mathematical Formulation of Segmentation Methods

The term *segmentation* of a 3D object refers to the partitioning of this object into k non-overlapping parts. An example segmentation of a 3D mesh is shown in *Figure 26*. In particular, following the formal definition given by Agathos et al. [110], if S is the set of vertices, edges or faces of the initial object, the segmentation algorithm should find k non-overlapping sets S_i, such as:

$$\cup_{i=1}^{k} \mathrm{S}_i = \mathrm{S}, \qquad \mathrm{S}_i \subset \mathrm{S}, \qquad \mathrm{S}_i \cap \mathrm{S}_i = \emptyset, \qquad i,j = 1,\dots k, \qquad i \neq j$$

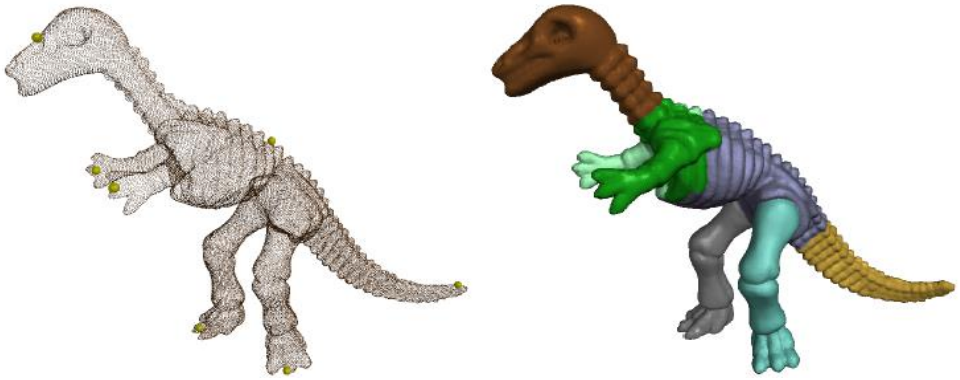


Figure 26: Example of a segmentation algorithm. Left: Input point cloud. Right: direct segmentation of the point cloud dataset. This figure is taken from [111].

[110] Agathos, Alexander, et al. "3D mesh segmentation methodologies for CAD applications." *Computer-Aided Design and Applications* 4.6 (2007): 827-841.

[111] Lai, Yu-Kun, et al. "Rapid and effective segmentation of 3D models using random walks." *Computer Aided Geometric Design* 26.6 (2009): 665-679.

The exact sets S_i that are created by the segmentation algorithm depend on a set of criteria that are used to guide the segmentation. These criteria in turn depend on the exact problem domain, which the segmentation procedure is applied to. As expected, a completely different set of segmentation criteria is used in 3D object reassembly methods than on mesh compression or other problem domains. Later in this report we will present an overview of the most used segmentation criteria.

Additionally, as discussed by Shamir [112], for each specific problem, it is possible to define a general criterion function $J(S_1, S_2, \dots, S_k)$, that for a given set of segments (partitions) provides a measure on how desirable this partitioning is. The segmentation problem can then be seen as an optimization of the function $J: 2^S \rightarrow R$ under a set of constraints C. In this regard, 3D object segmentation can be formalized as an optimization problem.

While the criterion function that we seek to optimize is different for each specific application of segmentation algorithms, we can identify a common set of general methodologies that are used in order to perform this segmentation.

9.1.1. Objectives of segmentation methods for 3D object reassembly

For the purpose of 3D object reassembly, we would like to calculate geometry segments with the following characteristics:

- The produced segments should be rather simple, in order to facilitate the matching algorithm to perform robustly. As noted earlier, it is difficult and more computationally expensive to design robust geometry matching algorithms for geometric objects with complex topology.
- The segmentation should not produce an excessively large number of small segments (over-segmentation), something that would decrease the reliability of the matching. For the same reason, segmentation to very few segments should be avoided. The optimal number of segments for a specific object depends on the matching algorithm that will be used later in the reassembly procedure.
- A single segment should be composed from areas that are either *intact* or *fractured*, but not both. Subsequent matching operations should be performed only on fractured segments.
- The segmentation should be fully automatic, without any used intervention. Therefore, in this report we will focus on algorithms for automatic segmentation of meshes.
- Each typical input mesh for our purpose consists of hundreds of thousands of triangles. Even if on-the-fly matching and reassembly is not required, a high performance segmentation algorithm is desired in order to make the technique practical. For this reason, we will investigate whether a massively parallel algorithm can be developed, for efficient execution on GPUs and similar stream processors.

9.1.2. Related Problems

One closely related field of research to *mesh segmentation* is *image segmentation*. In fact many methods for mesh segmentation are analogous to well-established methods for image segmentation. This is to be expected, since meshes and geometrical data in general can be often represented as images, which is particularly the case with *Range Images* or *Geometry Images* [113]. Another closely related area of research is clustering and partitioning algorithms in statistics and machine learning. While these fields are closely related to segmentation, our overview here will focus on methods that directly address the problem of segmentation in three dimensional meshes.

---

[112] Shamir, Ariel. "A survey on mesh segmentation techniques." *Computer graphics forum*. Vol. 27. No. 6. Blackwell Publishing Ltd, 2008.

[113] Gu, Xianfeng, Steven J. Gortler, and Hugues Hoppe. "Geometry images." ACM Transactions on Graphics (TOG). Vol. 21. No. 3. ACM, 2002.

9.2. General Categorization

Segmentation methods can be classified into two general categories, *surface-based* and *part-based.* These two classes of segmentation algorithms are inherently different and provide segments with different characteristics, as shown in *Figure 27.* These two classes of segmentation algorithms are outlined below:

Surface-based algorithms decompose the input object into regions, which represent distinct surfaces or facets. The geometrical topology of each facet can be roughly approximated as (part of) a simpler primitive, such as a plane, sphere, cylinder, a generalized quadratic surface etc. The segmentation algorithms used by most pairwise 3D object matching algorithms fall into this category, since the matching is based on the similarity of the simpler regions that are produced by the segmentation. In this report we will mainly focus on algorithms in this general category.

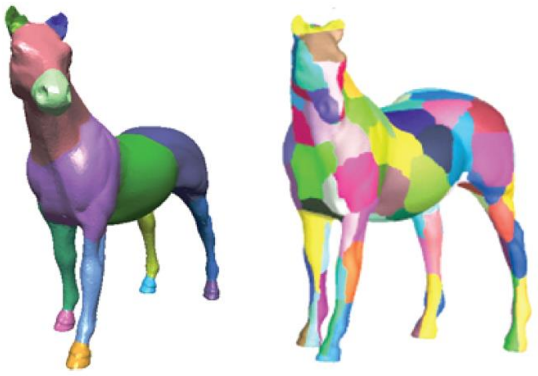


Figure 27: Two different types of mesh segmentation. Left: Part-based segmentation. Right: Surface-based segmentation of the same mesh. Image taken from [112].

Part-based algorithms divide a single complex object into smaller *meaningful* parts with more simplistic geometry. Most methods in this category follow the *minima rule* introduced by Hoffman and Richards [114], which states that human perception usually divides a surface into parts along concave portions of the surface. The minima rule is based on the observation that when two separate objects interpenetrate each other, they always meet in concave discontinuities, as indicated in *Figure 28.* The output of this class of algorithms typically consists of small volumetric parts, rather than the simplistic surfaces in surface-based algorithms.

This kind of output is not very suited for direct geometric matching, because in this case, more complex algorithms would be required to perform the matching. However, we still review some of the most influential work in this area, since this kind of decomposition of a complex object in simpler meaningful parts can still be useful in various areas of our research, other than matching.

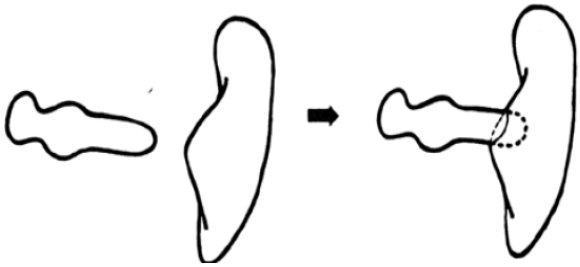


Figure 28: The "minima rule" is based on the observation that when two objects interpenetrate each other, they always meet in concave discontinuities, as indicated by the dashed contours. This figure is taken from [114].

[114] Hoffman, D.; Richards, W.: Parts of recognition, Cognition 18, 1984, 65–96.

9.3. Segmentation Criteria

In the next paragraphs we will review the most widely used criteria that are directly related to our research for the segmentation of a mesh into distinct regions. In particular, we will discuss criteria for the partitioning of a mesh into planar facets, simple primitives (spheres, cylinders, etc…) or curvy segments. Other criteria not related to our research and hence not analyzed below, include: symmetry, parameterization distortion, convexity, medial axis and motion characteristics.

9.3.1. Planarity of Various Forms

Planarity is one of the most useful criteria in mesh partitioning methods. It is used for mesh simplification and parameterization, texture atlas generation and most importantly for our research, for 3D object matching. As noted earlier in this report, the creation of relatively planar segments with simple topology is desirable in order to simplify the surface matching algorithms.

Many measures of planarity are used in the bibliography. The most important, as indicated by Shamir [112], are:

L_∞ distance norm. Given a representative plane (a, b, c, d) for a cluster of vertices, for any vertex $(v_x, v_y, v_z, 1)$ it measures the maximum distance from the plane:

$$\left|(a, b, c, d) \cdot (v_x, v_y, v_z, 1)\right| \leq \epsilon$$

L_2 distance norm. Given a representative plane (a, b, c, d) for a cluster of vertices, for a set of vertices v_i it measures the average distance from the plane:

$$\frac{1}{k}\sum_{i=1}^{k}\left((a, b, c, d) \cdot (v_x, v_y, v_z, 1)_i\right)^2 \leq \epsilon$$

L_∞ orientation norm. Given a representative plane (a, b, c, d) for a cluster of vertices, for any point or face normal (n_x, n_y, n_z) it measures the maximum normal deviation:

$$\left(1 - (a, b, c) \cdot (n_x, n_y, n_z)\right) \leq \epsilon$$

L_2 orientation norm. Given a representative plane (a, b, c, d) for a cluster of vertices, and a set of point or face normals n_i, it measures the average difference of normals:

$$\frac{1}{A}\sum_{i=1}^{k}\frac{1}{A_i}\left(1 - (a, b, c) \cdot (n_x, n_y, n_z)_i\right)^2 \leq \epsilon$$

where A_i is the weight associated with each normal n_i, and $A = \sum_i A_i$. In most of the cases, A_i is the area of each face and A is the total area of the object.

9.3.2. Fitting to simple primitives

For many algorithms, it is often required to create non-planar clusters of vertices. In this case, one solution is to try to find the best fitting primitive for a set of vertices using a variant of the *least squares* fitting algorithm. Several works use simple primitives such as spheres, cones, cylinders, surfaces of revolution, etc.

This partitioning to simple primitives can also be performed with *slippage analysis*, as presented by Gelfand and Guibas [115]. Slippable shapes are rigid motions which, when applied to a shape, slide the transformed version against the stationary version without forming any gaps. Slippable shapes include rotationally and translationally symmetrical shapes such as planes, spheres, and cylinders, which are often found as components of scanned mechanical parts. The algorithm determines the

[115] Gelfand, Natasha, and Leonidas J. Guibas. "Shape segmentation using local slippage analysis." Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing. ACM, 2004.

slippable motions of a given shape by computing eigenvalues of a certain symmetric matrix derived from the points and normals of the shape. Slippable components are discovered in the input data by computing local slippage signatures at a set of points of the input and iteratively aggregating regions with matching slippable motions.

9.3.3. Difference in geometric normals

Perhaps the simplest way to create non-planar clusters is to measure the difference of the normal directions or the angle between the normals (*dihedral angles*) of two elements (points or facets). Depending on the tolerance, both planar and curved surfaces can be created. However, this simple method has difficulty dealing with noisy data or small scale details, which should not be taken into consideration when segmenting the input mesh.

9.3.4. Curvature

Segmentation can also be based on the local *curvature* of a mesh. This can be computed either with local differences or by fitting a quadratic polynomial locally around the point of interest and using the curvature of this polynomial as the local curvature of the mesh. The resulting measure is shown on *Figure 29* (left).

Many other metrics that measure the curvature of a surface have been proposed in the bibliography. The survey by Gatzke and Grimm [116] provides a detailed overview of these metrics and the methods to compute them.

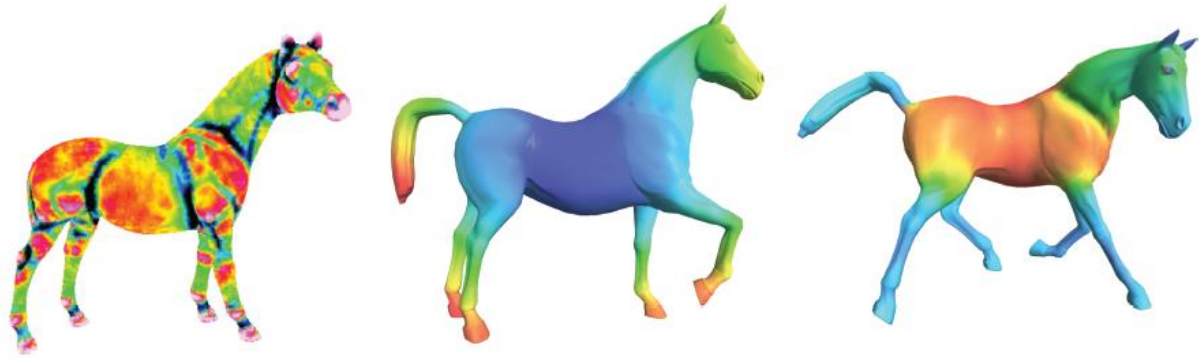


Figure 29: Different measures for mesh segmentation. Left: Minimum curvature, Middle: average geodesic distance, Right: Shape Diameter Function. Image taken from [112].

9.3.5. Average Geodesic distance

The *Average Geodesic Distance (AGD)* or *centricity* is mainly used for part-based segmentation, since it depends more on the global geometry and topology of the object, as shown in *Figure 29* (middle). It measures the average geodesic distance from each point to all other points on the mesh. For a given vertex v of the mesh, its centricity $m(v)$ is calculated as:

$$m(v) = \int_{p \in M} g(v,p) dM$$

where $g(v,p)$ denotes the geodesic distance between the surface points v and p and M denotes the surface of the mesh.

[116] Gatzke, Timothy D., and Cindy M. Grimm. "Estimating curvature on triangular meshes." International journal of shape modeling 12.01 (2006): 1-28.

9.3.6. Shape Diameter Function

A similar measure, typically used for part-based segmentation, is the Shape Diameter Function (SDF), which measures the local diameter of the object at each point of the mesh. A visualization of this function on a simple mesh is shown in *Figure 29* (right). This is computed by firing rays for each point on the surface towards the interior of the mesh, and averaging the distance they travel until they hit another surface. This is a good measure of the local thickness of a mesh, but the computation of the SDF is rather expensive because it requires many ray-casting operations.

9.4. Classification of Segmentation Methodologies

Segmentation of three dimensional objects has been a topic of interest for the research community for several years and several methods have been proposed in the bibliography. A number of surveys [110] [117] [112] provide an overview and comparisons of these methods. As discussed in these surveys, we can roughly classify segmentation algorithms based on the general methodologies and partitioning strategies that are used by each one of them. Our overview in this section is largely based on the material presented in these reports, but we have also updated the classification that was presented with some recent advances on the field of mesh segmentation and we have mostly focused on the methodologies that are related to our work.

The general classes of algorithms that we identify are region growing methods, with a special case for multi-region growing, hierarchical clustering, iterative clustering, contour extraction, spectral analysis, implicit methods and critical points-based methods. The next paragraphs discuss each category of algorithms separately.

9.4.1. Region Growing

One of the simplest methods for mesh segmentation is the greedy approach generally known as *region growing*. Region growing proceeds by selecting an element (point or face) from the input mesh as a seed to a region, which is incrementally grown into a larger region, by merging neighboring elements, until some specific criteria are met. When no other neighbors that meet these criteria are found, the process starts with a new seed and the algorithm terminates when all input elements are classified into a region.

The main strategy used by the region growing methods is shown in the following algorithm:

```
Initialize an empty priority queue Q.
Loop until all elements belong to a region
     Choose an un-clustered seed element E .
     Insert E into Q.
     Create a cluster C from E.
     Loop until Q is empty
           Get the first element E′ from Q
           If E′ meets the criteria
                 Cluster E′ with C.
                 Insert E′ neighbors into Q.
Merge small clusters into neighboring ones.
```

Region growing methods require a structured representation of the input data, where the neighbors of an element can be easily found. In particular, these methods were originally proposed for images, were the neighboring elements are trivially to compute. However, similar approaches can be also applied to 3D meshes and other structured representations.

The main difference among region growing methods is the criterion that is used to determine if a neighboring polygon can be merged into a cluster. The priority used in the queue is also usually tightly coupled with this criterion. The most trivial implementation of a region growing method selects seed

[117] Shamir, Ariel. "Segmentation and shape extraction of 3D boundary meshes." EUROGRAPHICS. Vol. 6. 2006.

polygons randomly and uses a queue without a priority. Since the algorithm tends to create very small regions, a post-processing step is used in order to merge small clusters into neighboring ones, as shown in *Figure 30*.

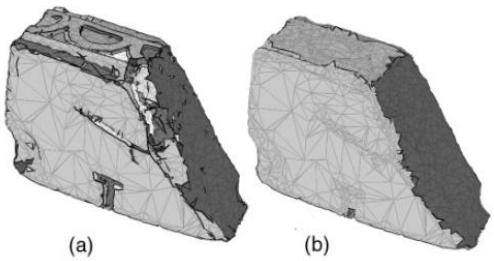


Figure 30: A post processing step is often used in region growing methods in order to merge small clusters. a) Initial clusters after the segmentation process. b) Final clusters after the post-processing merge process. Image taken from [118].

Perhaps the most relevant work in this category is the method by Papaioannou et al. [118], where the authors present a region growing method for mesh segmentation, in the context of 3D object reassembly. Even though the seed polygons are selected randomly and no priority queue is used, the method gives reasonable results for the test dataset, consisting of fragmented archeological objects. The segmentation criterion is the deviation of a polygon's normal from the average normal of a region. Much like other methods in this category, the algorithm results in over-segmentation of the input mesh, that is corrected in a post-processing step.

Besl and Jain [119] perform region growing on range images. They compute an initial labeling of the elements using the Gaussian and mean curvature. Seed regions are constructed based on this labeling and region growing is used to create the final segmentation. The authors fit variable order bi-variate polynomials to each region and neighboring elements are added according to their distance from the approximating polynomials. This algorithm has been also extended to triangular meshes by Vieira and Shimada [120].

The superfaces algorithm by Kalvin and Taylor [121], simplifies meshes using a region-growing approach. Seed faces are selected randomly and are grown using an $\boldsymbol{L}_\infty$ face-distance criteria, along with a variant of the face-normal criteria and a constraint that prevents regions from folding over. A post processing step is used in this method too to refine the computed segments. The method by Lavoué et al. [122] creates clusters of constant curvature by using a surface curvature criterion.

The main advantage of region growing methods is their simplicity combined with the rather satisfactory results they provide. On the other hand, the main problem with these methods is that the resulting segmentation is highly dependent on the selected seeds and to the specific order that these seeds have been selected and processed. A bad selection of initial seeds can lead to bad segmentation. Another drawback of these methods is the handling of smooth edges, where the boundary between two

[118] Papaioannou, Georgios, E-A. Karabassi, and Theoharis Theoharis. "Segmentation and surface characterization of arbitrary 3D meshes for object reconstruction and recognition." *Pattern Recognition, 2000. Proceedings. 15th International Conference on*. Vol. 1. IEEE, 2000.

[119] Besl, Paul J., and Ramesh C. Jain. "Segmentation through variable-order surface fitting." Pattern Analysis and Machine Intelligence, IEEE Transactions on 10.2 (1988): 167-192

[120] Vieira, Miguel, and Kenji Shimada. "Surface mesh segmentation and smooth surface extraction through region growing." Computer aided geometric design 22.8 (2005): 771-792.

[121] Kalvin, Alan D., and Russell H. Taylor. "Superfaces: Polygonal mesh simplification with bounded error." Computer Graphics and Applications, IEEE 16.3 (1996): 64-77.

[122] Lavoué, Guillaume, Florent Dupont, and Atilla Baskurt. "A new CAD mesh segmentation method, based on curvature tensor analysis." Computer-Aided Design 37.10 (2005): 975-987.

regions is not clearly defined. In these cases, the greedy nature of the algorithm can potentially fail to detect the desired boundaries. Furthermore, region growing methods with a single active region have limited parallelism and are not well suited for massively parallel implementation on stream processors and GPUs. This problem is partially mitigated with the multi-region growing methods that we examine in the next paragraph.

9.4.2. Multi-Region Growing

A common variation of the simple region growing algorithm grows multiple regions in parallel. The main algorithm that these methods follow is shown below:

```
Initialize an empty priority queue Q.
Loop until all elements belong to a region
     Choose a set of seed elements [Ei].
       Create a cluster Ci for every Ei.
     Insert the pairs <Ei,Ci> into Q.
           Loop until Q is empty
           Get the first pair  <Ei', Ci'> from Q
           If Ei' meets the criteria to be clustered in Ci
                 Cluster Ei' with Ci.
                 Insert all un-clustered neighbors into Q.
Merge small clusters into neighboring ones.
```

As with the single-region growing approach, the quality of the segmentation is highly dependent on the initial seed selection. The main difference between the various methods that have been proposed in the bibliography lies on the criteria that are used in order to test if a polygon should be clustered within a region and the priority of elements in the queue.

As an example, multi-region growing has been used for mesh parameterization by Sorkine et al. [123], a technique also known as automatic texture atlas generation. This is a segmentation problem from another area of research but shares a lot of common characteristics with the segmentation in the context of surface matching. This parameterization method starts the region growing process from multiple randomly selected seeds and the main criterion in order to merge a triangle with a cluster is the magnitude of distortion that it will create on the triangle when flattening to 2D.

The well-known *watershed algorithm*, which was first used for image segmentation [124], is essentially a region growing method with multiple seeds and falls into this category. In this algorithm, the segmentation is performed in a similar way the water fills a landscape surface, as shown in Figure 31. As the water floods the geometrical basins, called *catchment basins*, there will be points where the flood regions meet. These points define the watershed lines and divide the surface to distinct regions.

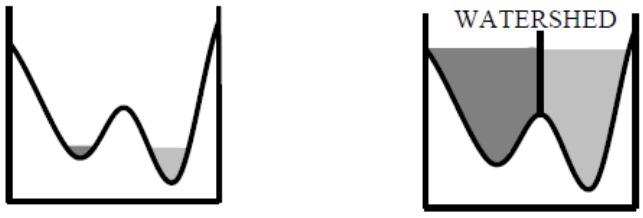


Figure 31: The watershed algorithm. Left: Initial flooding of the landscape surface. Right: Watershed line emerged at a certain flooding level. Image taken from [110].

[123] Sorkine, O., Cohen-Or, D., Goldenthal, R., & Lischinski, D. (2002, October). Bounded-distortion piecewise mesh parameterization. In Proceedings of the conference on Visualization'02 (pp. 355-362). IEEE Computer Society.

[124] Pratikakis, I.: Watershed-driven image segmentation, PhD Thesis, Vrije Universiteit Brussel (VUB), 1998.

For 3D mesh segmentation, the algorithm does not operate directly on the mesh geometry, but a height function $F: R^3 \rightarrow R$ is used, as proposed by Mangan and Whitaker [125]. The local minima of this function correspond to the basins of the geometry (seeds), where the flooding (or region growing) will begin. Similarly, the watershed lines that divide the mesh into different segments will be located at the local maxima of this function. A common problem of watershed methods is that they result in over-segmentation. One potential solution is to merge the smaller regions in a post-processing step.

An interesting multi-seed semi-automatic segmentation method that is based on random walks was proposed by Lai et al. [111]. Their random walk method is a variation of an earlier method [126] for image segmentation. The user first places a number of initial seeds on the input mesh, as shown in Figure 32 (left). In this case, the number of final segmented regions is known in advance and is equal to the number of seed elements. The algorithm associates probabilities to the edges of every face. Each one of these corresponds to the probability that a random walk proceeds from on face to an adjacent one through the edge that lies between them. A face that is not a seed is marked as belonging to the to a specific seed region, if a random walk starting at that face has higher probability reaching this seed element than any other seed. The probability from stepping from one face to another depends on the dihedral angle between the two faces. The authors also propose a variation of their method for automatic segmentation. In this case, a large number of evenly distributed seeds are selected randomly, resulting in an over-segmentation of the mesh. The method then refines the results, by merging smaller regions. The main advantages of this method are that it is computationally efficient, thus it can be used for the segmentation of large objects or a large number of models, it handles noise and small-scale surface details well and it can work in both point clouds and structured meshes.

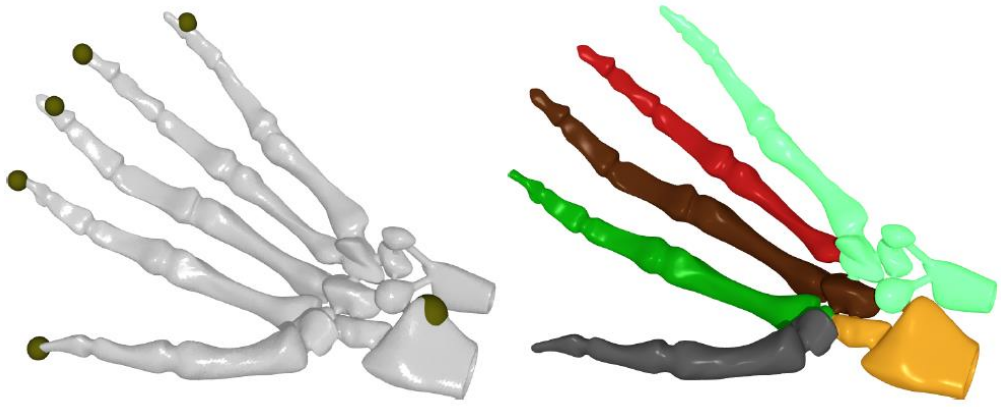


Figure 32: Left: input mesh with seeds selected by the user. Right: Segmentation result using random walks. Image taken from [111].

9.4.3. Hierarchical Clustering

Hierarchical clustering methods, much like region growing, incrementally create big clusters of elements by merging smaller clusters. Initially each polygon is considered a cluster, and clusters are merged to bigger ones by choosing the best merging operation for all clusters.

The main algorithm used by this family of methods is shown below:

[125] Mangan, Alan P., and Ross T. Whitaker. "Partitioning 3D surface meshes using watershed segmentation." Visualization and Computer Graphics, IEEE Transactions on 5.4 (1999): 308-321.

[126] Grady, Leo. "Random walks for image segmentation." Pattern Analysis and Machine Intelligence, IEEE Transactions on 28.11 (2006): 1768-1783.

```
Initialize an empty priority queue Q.
Insert all valid element pairs in Q.
Loop until Q is empty
      Get the next pair (u, v) from Q
      If (u, v) can be merged
            Merge (u, v) into w
            Insert all valid pairs of w to Q
```

This bottom-up creation of clusters is still greedy, but unlike region growing methods, the clusters are grown hierarchically, taking into account all the existing clusters of the mesh. Because this family of methods does not concentrate on growing a few clusters at a time, the resulting segmentation can be potentially more optimal, according to a global optimization function. To highlight the difference with the region growing methods, Shamir [117] uses the term "*global-greedy*" methods in order to characterize the hierarchical clustering methods and "*local-greedy*" for the region growing ones.

Much like the region growing methods, the discrimination among the proposed methods in the literature lies on the criteria used to merge the clusters. The method proposed by Garland et al. [127] uses the L_2 distance and orientation norms as measures of planarity. The formulation presented by the authors uses a quadric error metric for efficient computation. The algorithm also uses a *shape bias* during the clustering that favors the creation of more compactly shaped clusters. Attene et al. [128] use as a clustering criteria the approximation error when fitting a set of triangles into a set of primitives (planes, spheres, cylinders). Their method merges a set of triangles into a single cluster when the minimum approximation error, computed against all possible fitting primitives, is below a specified threshold.

9.4.4. Iterative Clustering – Classification

In the previous approaches, the number of clusters in the segmentation is not known in advance. However, there is a family of problems where the number of desired clusters is given a priori. As an example, for the case of 3D object reconstruction such a problem is the classification of the input facets to two clusters, namely *fragmented* and *intact*. Obviously, in this particular problem, the number of desired clusters is always the same.

In this family of problems the optimal segmentation can be found using as a basis *the k-means* clustering algorithm, which was initially proposed by Stuart Lloyd [129] as a technique for pulse-code modulation. The segmentation begins with *k* clusters defined by *k* representative elements. Each element is assigned to one specific cluster using a quantization process. Subsequently, the *k* representatives and the corresponding clusters are recalculated and the quantization process is repeated. The algorithm terminates when the representatives and the corresponding clusters converge. The main algorithm used by this family of methods is shown below:

```
Initialize k representatives of k clusters
Loop until representatives do not change
      For each element s
            Find the best representative i for s
            Assign s to the i^th cluster
      For each cluster i
            Compute a new representative
```

[127] Garland, Michael, Andrew Willmott, and Paul S. Heckbert. "Hierarchical face clustering on polygonal surfaces." Proceedings of the 2001 symposium on Interactive 3D graphics. ACM, 2001.

[128] Attene, Marco, Bianca Falcidieno, and Michela Spagnuolo. "Hierarchical mesh segmentation based on fitting primitives." The Visual Computer 22.3 (2006): 181-193.

[129] Lloyd, Stuart. "Least squares quantization in PCM." Information Theory, IEEE Transactions on 28.2 (1982): 129-137.

The main concern with this family of methods is the convergence, i.e. in this case, if and how fast the method will terminate by converging to the final set of clusters. The exact method the representative clusters are chosen and the exact method each element is assigned to a specific cluster should guarantee that the process converges. Furthermore, the initial set of clusters can affect the final segmentation.

Cohen-Steiner et al. [130] use a variation of *k*-means clustering technique in order to create planar shape proxies (mesh simplification). The authors propose and examine two different error metrics. The L^2 metric measures the integral of the squared distance between the points of the original and the proxy surface:

$$L^2(R_i, P_i) = \int \int |x - \Pi_i(x)|^2 dx$$

where R_i is a region on the input mesh and $P_i = (X_i, N_i)$ its associated proxy. $\Pi_i(y)$ denotes the orthogonal projection of y on the proxy plane going through X_i and normal to N_i. They also investigate a normal-based measure of distortion, called $L^{2,1}$ which is defined as:

$$L^{2,1}(R_i, P_i) = \int \int |n(x) - n_i|^2 dx$$

where *n(x)* denotes the surface normal at point *x*. As demonstrated in *Figure 33*, the second metric creates clusters with greater anisotropy, something that might be preferable depending on the application. Furthermore, the $L^{2,1}$ metric is significantly faster to compute, since it involves the averaging of normals over the associated region. On the other hand, the L^2 metric requires the computation of a covariance matrix, which is significantly more expensive.

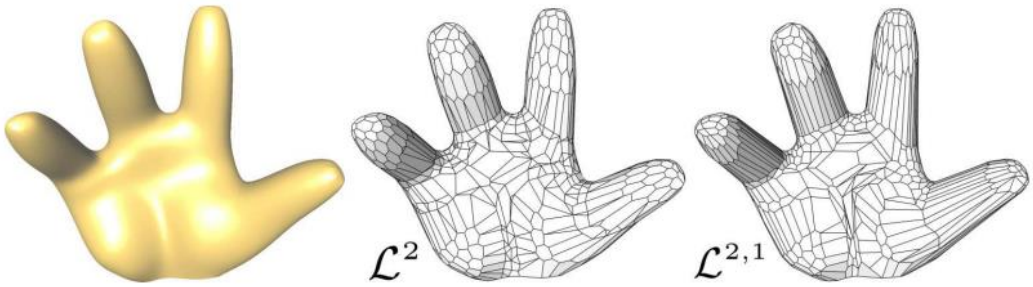


Figure 33: Iterative clustering with two different error metrics. Image taken from [130].

9.4.5. Contour Extraction

Algorithms in this category try to extract the boundaries of the segmented areas directly from the mesh, by following sharp edges, concavities or other features of the surface, depending on the segmentation criteria. The main advantage of the methods in this general category is that they provide direct control on the smoothness, the size or the shape of the segment boundaries, since these are explicitly computed by a "*mesh scissoring*" algorithm. In contrast to this approach, when using methods from the other categories, the boundaries are defined after the completion of the clustering of the mesh elements, thus there is no direct control on the features of the segmentation boundaries.

[130] Cohen-Steiner, David, Pierre Alliez, and Mathieu Desbrun. "Variational shape approximation." ACM Transactions on Graphics (TOG). Vol. 23. No. 3. ACM, 2004.

Lee et al. [131] [132] explicitly extract contours on the mesh, based on the minimum curvature. Subsequently, some of these contours are selected and closed, in order to form the segmentation boundaries. Their selection criteria are based on a protrusion function and the contours are closed by finding the tightest path on the mesh. Finally, geometric snakes [133] are used to refine the segmentation boundaries. This algorithm provides a part-based segmentation, but the selected contours not always divide the input mesh into meaningful parts that follow the minima rule. While this algorithm is part-based, similar surface-based algorithm s can be constructed by changing the criteria of the contour selection/creation, as we will see in the next paragraph.

The work by Huang et al. [134] for fragmented object reconstruction is directly related to our research goals. In this section we will focus on the segmentation algorithm used by their method, while we will review the complete method in more detail later in this report. The authors base their segmentation algorithm on an earlier work by Pauly et al. [135] for multi-scale *feature* extraction on point sampled surfaces. *Features* are usually defined as entities of an object that are considered important by a human for an accurate description of the object. It is clear that this is a highly subjective definition, which is difficult to express with mathematical equations. The authors focus on line-type features, since these are probably the most important features for many types of objects. Given an unstructured point cloud $P = \{p_i \in R^3\}$, their algorithm first classifies the input points according to the probability they belong to a feature. This is performed using a multi-scale approach which is based on a local *surface variation* metric that is introduced by the authors and is defines as:

$$\sigma_n(p) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}, \qquad \lambda_0 \leq \lambda_1 \leq \lambda_2$$

where λ_i are the eigenvalues of the 3x3 covariance matrix C defined as:

$$C = \frac{1}{k}\begin{bmatrix} p_{l_1} - p' \\ \dots \\ p_{l_{1k}} - p' \end{bmatrix}^T \cdot \begin{bmatrix} p_{l_1} - p' \\ \dots \\ p_{l_{1k}} - p' \end{bmatrix}, \qquad i_j \in N_p$$

where N_p is the local neighborhood of the k-nearest points around the sample point p and p' is the centroid. The multi-scale nature of the method avoids problems with noisy data and can also ignore small scale details of the surface that should not be considered as features. This multi-scale approach assigns a weight on each point and only points above a certain threshold are kept. The algorithm then computes the minimum spanning graph of the remaining points. Each separate component of the graph is modeled by a snake, an energy-minimizing spline that is attracted to the feature vertices. Using Euler integration, the feature lines that are represented by the snakes are smoothed, while maintaining a close connection to the underlying surface. The steps of this method are illustrated in *Figure 34*.

[131] Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., & Seidel, H. P. (2004, October). Intelligent mesh scissoring using 3d snakes. In Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on (pp. 279-287). IEEE.

[132] Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., & Seidel, H. P. (2005). Mesh scissoring with minima rule and part salience. Computer Aided Geometric Design, 22(5), 444-465.

[133] Lee, Yunjin, and Seungyong Lee. "Geometric snakes for triangular meshes." Computer Graphics Forum. Vol. 21. No. 3. Blackwell Publishing, Inc, 2002.

[134] Huang, Q. X., Flöry, S., Gelfand, N., Hofer, M., & Pottmann, H. (2006, July). Reassembling fractured objects by geometric matching. In ACM Transactions on Graphics (TOG) (Vol. 25, No. 3, pp. 569-578). ACM.

[135] Pauly, Mark, Richard Keiser, and Markus Gross. "Multi-scale Feature Extraction on Point-Sampled Surfaces." Computer graphics forum. Vol. 22. No. 3. Blackwell Publishing, Inc, 2003.

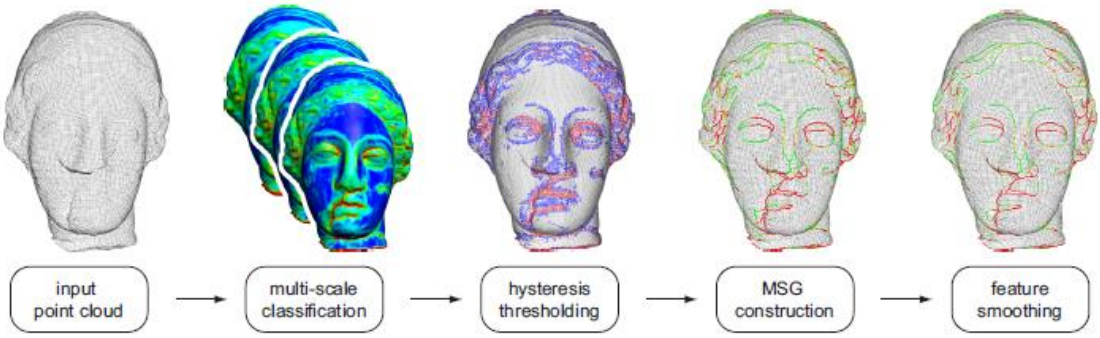


Figure 34: Multi-scale feature extraction pipeline. This approach is the basis for the segmentation algorithm proposed by Huang et al. [134]. (Image taken from [135])

Huang et al. [16] use a variation of the feature extraction algorithm by Pauly et al. in order to determine the segmentation boundaries on the input mesh. In particular, they make the following modifications:

- Instead of the surface variation metric, they determine the edges of the object using the multi-scale integral invariants, introduced by Pottman et al. [136]. This means that points are classified as belonging to edges if they have consistently high sharpness along different scales.
- After reconstructing the Minimum Spanning Graph, the authors specifically extract the long closed cycles from the graph, since the goal of the feature extraction in their algorithm is to form the segmentation boundaries of the mesh.

After determining an initial segmentation using the above modifications to the Pauly et al. algorithm, they refine the segmentation, based on the roughness and the sharpness of the generated segments. In particular, they construct a connectivity graph between the generated segments and they apply the *normalized graph cut* method of Shi and Malik [137], until the surface roughness variance is less than a threshold. As a weight for this operation they use the difference between the mean surface roughness of the adjacent segments. Furthermore, since the edge-extraction sometimes results in over-segmentation, the authors perform a second series of normalized cuts.

More recently, Willis and Zhou [138] proposed another contour-based approach, which operates on a graph that is defined over the mesh geometry. In particular, graph nodes correspond to the points of the mesh and edges correspond to the actual edges of the 3D mesh model. The method then calculated a contour-based segmentation of the mesh using the following steps:

1. A weight for each edge is computed, using a salience function to detect ridges, areas of maximal curvature or valleys. In particular, the authors propose the use of the following functions:

$$w_{ridge}(e_{ij}) = \frac{u_i + u_j}{\|u_i + u_j\|} \cdot \frac{e_{ij}}{\|e_{ij}\|} \cdot k_{max}$$

$$w_{curv}(e_{ij}) = \frac{u_i + u_j}{\|u_i + u_j\|} \cdot \frac{e_{ij}}{\|e_{ij}\|} \cdot \max(k_{max}, k_{min})$$

[136] Pottmann, H., Wallner, J., Huang, Q. X., & Yang, Y. L. (2009). Integral invariants for robust geometry processing. Computer Aided Geometric Design, 26(1), 37-60.

[137] Shi, Jianbo, and Jitendra Malik. "Normalized cuts and image segmentation." Pattern Analysis and Machine Intelligence, IEEE Transactions on 22.8 (2000): 888-905.

[138] Willis, Andrew, and Beibei Zhou. "Ridge Walking for 3D Surface Segmentation.", 3DPVT 2010.

$$w_{valley}(e_{ij}) = \frac{u_i + u_j}{\|u_i + u_j\|} \cdot \frac{e_{ij}}{\|e_{ij}\|} \cdot (-k_{min})$$

where u_i is a vector that approximates the direction of the principal curvature. The values of k_{min} and k_{max} are approximations of the principal curvatures averaged over the extent of the edge e_{ij}. The quantity $\frac{u_i+u_j}{\|u_i+u_j\|} \cdot \frac{e_{ij}}{\|e_{ij}\|}$ represents how well the direction of the edge agrees with the estimated direction of the ridge-line on the surface, which is the direction of the minimum curvature for convex regions and the direction of the maximum curvature for concave regions.

2. Compute the maximum spanning tree of the graph that corresponds to the 3D mesh (see Figure 35). This is performed using standard algorithms, like Prim's or Kruskal's. Denote S the set of edges on the spanning tree, and L ("loopy edges") the remaining edges.
3. Compute weights for the loopy edges and store them in a stack. The ones with the highest weights will be used to form closed contours.
4. Loopy edges are added into the mesh. Each time an edge is added, criteria for the uniqueness and the length of the edges are tested.
5. The mesh segmentation is computed from the contours that were created in the previous step.

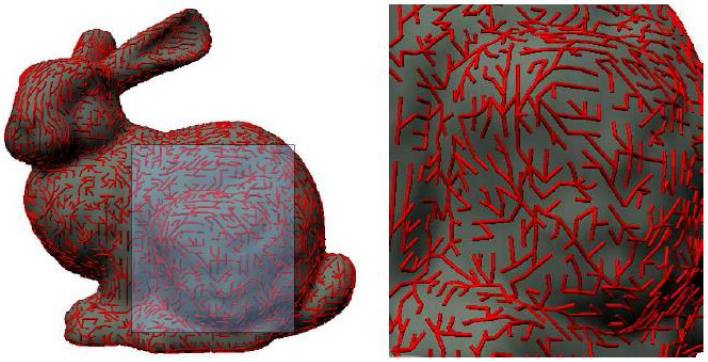


Figure 35: Left: The maximum spanning tree defined over a mesh. Right: A zoomed view of the surface mesh from left. Image taken from [138].

9.4.6. Spectral Analysis

Methods in this category rely on *spectral graph theory* in order to compute a partitioning of the mesh. Spectral graph theory studies the properties of a graph in relationship to certain matrices associated with this graph, like the *adjacency matrix* **A** and the *Laplacian matrix* **L**. In particular, methods in this category use a graph-based representation in order to encode the connectivity information of the input mesh. Such a graph can be represented with an *adjacency matrix*, which is an alternative means to represent which nodes (vertices) of a graph are adjacent to each other. The Laplacian of a graph G is defined as:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

where **A** is the adjacency matrix and **D** is a diagonal matrix which holds the degree (valence) of each vertex i as the diagonal element d_{ii}. Usually, methods in this category consider the eigenvalues and eigenvectors of the Laplacian or a similarly defined matrix.

Most spectral analysis methods are used for part-based segmentation, thus they are not suited for our research, because this category of methods does not meet the criteria we have specified in Section

9.1.1. A notable exception is the work by Zhou et al. [139], where the authors propose a surface-based segmentation for texture atlas generation that combines two seemingly incompatible techniques: stretch-minimizing parameterization, based on the surface integral of the trace of the local metric tensor, and the "isomap" or MDS (multi-dimensional scaling) parameterization, based on an eigen-analysis of the matrix of squared geodesic distances between pairs of mesh vertices. The authors make a very useful observation, that geodesic distance distortion is closely related to stretch, even if these quantities have different definitions. Therefore their method segments the mesh into large meaningful parts, using spectral analysis and at the same time, without extra computations, provides an initial parameterization for each part. This parameterization is then improved with a few iterations of nonlinear stretch optimization. Finally, the authors also apply the graph-cut algorithm, as a post-processing operation, in order to optimize the boundaries of the different segments and further optimize stretching. In particular, their method performs the following steps:

1. Compute the surface spectral analysis, providing an initial parameterization
2. Perform a few iterations of stretch optimization
3. If the maximum stretch is less than a threshold stop
4. Perform spectral analysis to partition the surface into parts
5. Optimize the boundaries using the graph-cut technique
6. Recursively split charts until the maximum stretch is bellow the used specified threshold

While this work focuses on minimizing distortion when performing mesh parameterization, the result is a method that segments the object into meaningful parts, with simple topology. At the same time, each part has minimum distortion when mapped to 2D, where the distortion is measured using the average and worst-case stretching of local distances over the mesh. To a great extent, these characteristics meet the requirements we have outlined in Section 9.1.1. This method has the interesting property that it combines both surface-based and part-based segmentation. However, if desired, this can also be achieved by performing a part-based segmentation on a mesh, followed by a surface-based segmentation of each part.

9.4.7. Implicit Methods

Many segmentation methods do not operate directly on the set of input elements (points, facets), but derive the segmentation implicitly, by partitioning an alternative representation of an object, such as a skeleton or a graph. The techniques that we have included in this category produce part-based segmentation, thus they are not very applicable to our research on fragmented object reassembly. This is because part-based segmentation methods tend to produce complex segments, rather than simple facets, which are more preferable as an input to the 3D matching

For this reason we only present a brief outline of the respective algorithms.

One approach is to first extract the skeleton of the mesh and then perform the partitioning of the object based on the partitioning of the former. For instance, Li et al. [140] construct the skeleton of the object by performing simplification of the surface using the edge contraction method. The partitioning of the object is computed by sweeping a plane along the skeleton edges. The intersection of this plane with the mesh consists of one or more contours. By examining the way these contours change, the algorithm extracts the partitions of the mesh.

A similar method uses the *Reeb graph* of the input mesh in order to guide the segmentation. Reeb graphs are defined with the help of a piecewise linear function $F: M \subseteq \mathrm{R}^3 \rightarrow R$ over the three

[139] Zhou, K., Synder, J., Guo, B., & Shum, H. Y. (2004, July). Iso-charts: stretch-driven mesh parameterization using spectral analysis. In Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing (pp. 45-54). ACM.

[140] Li, X., Woon, T. W., Tan, T. S., & Huang, Z. (2001, March). Decomposing polygon meshes for interactive applications. In Proceedings of the 2001 symposium on Interactive 3D graphics (pp. 35-42). ACM.

dimensional surface *M*. An example of such a function is the height function $F(x, y, z) = z$. A *contour* of the surface (or a level-set of the function *F*) is defined as the set of points with a common value s. The Reeb graph of *F* is obtained by analyzing the evolution of these contours over the surface. In particular, points on the Reeb graph correspond to critical points of the mesh, where the topology of the contours changes. On the other hand, arcs on the Reeb graph represent a family of contours that do not change topology. A representative example is shown in *Figure 36*. Please note that embedding the reeb graph on the mesh provides a medial-axis-type skeleton representation of the surface, therefore this technique is similar to skeleton-based ones. A simple and computationally efficient algorithm for the computation of the Reeb graph is presented by Pascucci et al. [142]. Another way to construct the reeb graph is by the quantization of the centricity (protrusion) function, as discussed in Antini et al. [141]. The segmentation can then be directly derived by the discrete parts of the Reeb graph. To avoid over-segmentation, Antini et al. perform a simplification of the Reeb graph.

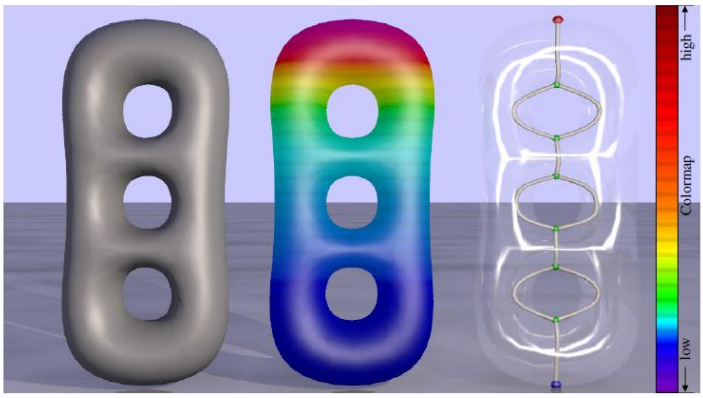


Figure 36: Reeb graph of a triple torus object, computed using a simple height function. Image taken from [142].

9.4.8. Critical Points based

Algorithms in this category use critical points defined on the mesh in order to guide the segmentation. These critical points are the salient features of the mesh and are used for the identification of the different protrusions of the mesh, as shown in *Figure 37*. The critical points typically are selected as the local maxima of the centricity (protrusion) function of the mesh.

Katz et al. [143] use a pose-invariant representation of the mesh computed using the multi-dimensional scaling method. They detect the prominent feature points in this representation and they project them back in the original representation of the mesh. The core components of the mesh are extracted using a spherical mirroring operation and finally the mesh is segmented and each segment represents at least one feature point. Finally a post processing step is used to refine the mesh.

Agathos et al. [145] propose a similar methodology, with the following steps:

1. The salient points are extracted by computing the local maxima of the protrusion function. For the computation of the local maxima the geodesic distance is used.

[141] Antini, G., Berretti, S., Del Bimbo, A., & Pala, P. (2005, July). 3D mesh partitioning for retrieval by parts applications. In Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on (pp. 1210-1213). IEEE.

[142] Pascucci, V., Scorzelli, G., Bremer, P. T., & Mascarenhas, A. (2007). Robust on-line computation of Reeb graphs: simplicity and speed. ACM Transactions on Graphics (TOG), 26(3), 58.

[143] Katz, Sagi, George Leifman, and Ayellet Tal. "Mesh segmentation using feature point and core extraction." The Visual Computer 21.8-10 (2005): 649-658.

2. Salient points are grouped according to their geodesic proximity.
3. An approximation of the core (main body of the object) is computed using the minimum cost paths between the representative salient points. In contrast, Lin et al. [144] uses a simple thresholding of the protrusion function.
4. The partitioning boundaries are created by detecting abrupt changes in the volume of the 3D object.
5. Segmentation boundaries are refined using a minimum-cut methodology.

The authors also discuss how various parts of their method could be parallelized and implemented in GPUs, to achieve high efficiency.

The algorithms in this category compute a part-based segmentation, thus they cannot meet the desirable characteristics and objectives that we have specified in Section 9.1.1, for a segmentation method in the context of fragmented object reconstruction. For this reason, a more detailed presentation of algorithms in this category is omitted. The interested reader is referred to various surveys on general segmentation methods [110][112].

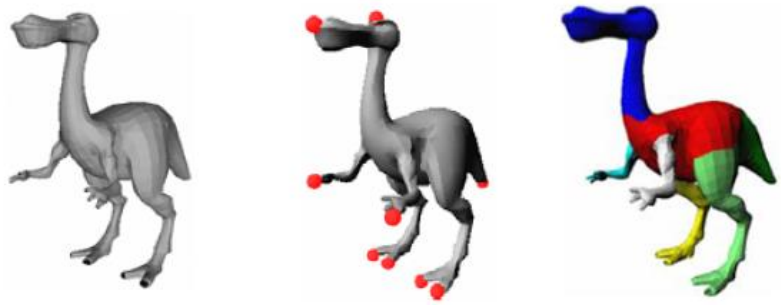


Figure 37: Mesh segmentation guided by critical points. Left: The input mesh. Middle: Critical point selection. Right: The final segments. Image taken from [145].

9.5. Discussion and Comparison

In this section we discuss some of our observations and conclusions about the segmentation methods that have been presented in this report.

9.5.1. Post Processing

Many segmentation methods require a post-processing step, to either merge smaller segments to larger ones, or smooth the boundaries between the segments that have been created, as shown in *Figure 38*. This could be potentially avoided when using a contour extraction method, since in this case, the boundaries are explicitly created by the segmentation algorithm and various constrains could be imposed during this process. In this regard, contour extraction methods have an advantage. Nevertheless, a prost-processing step can improve the results and is often employed in practice.

[144] Lin, H., H-YM Liao, and J-C. Lin. "Visual salience-guided mesh decomposition." Multimedia, IEEE Transactions on 9.1 (2007): 46-57.

[145] Agathos, A., Pratikakis, I., Perantonis, S., & Sapidis, N. S. (2010). Protrusion-oriented 3D mesh segmentation. *The Visual Computer*, *26*(1), 63-81.

Figure 38: Mesh segmentation may require post-processing to smooth out the boundaries between the segments. Image taken from [146].

9.5.2. Performance on noisy data

The data acquired by laser scanners and other digitization equipment are often noisy, containing various imperfections from measurement errors. However, many segmentation methods in the bibliography have been designed to operate on and exclusively applied to noise-free CAD models. For the purpose of the fractured object reassembly, though, we obviously need segmentation methods that have been tested and are reliable on noisy data. Similarly, small scale details should be ignored during the segmentation. As discussed previously in this section, Huang et al. [134] successfully deals with this problem by basing his segmentation method on a multi-scale measure of surface sharpness and roughness, which can ignore small scale surface characteristics. His results indicate that multi-scale object descriptors or measurements can successfully deal with noisy data.

9.5.3. Massively Parallel Implementation

Many of the above algorithms use rather complex data structures, like graphs for the connectivity information of the mesh or priority queues. The use of these data structures might have a negative impact on the performance of a massively parallel implementation on a GPU or a similar stream processor. It should be investigated whether a "naïve" algorithm, with simpler data structures, could potentially perform better, especially if performance in this stage of processing turns out to be an issue. To this end, it might be preferable to perform the segmentation using an image-based representation of the mesh (for example geometry image), in order to avoid the complex graph-based data structure that is required when processing meshes with connectivity information.

9.5.4. Comparison Table

Table 12 presents an overview and direct comparison of the mesh segmentation algorithms presented in this report. We report the general methodology followed by each algorithm, the type of segmentation (surface or part-based) and the attributes of the input mesh that was used in order to produce this segmentation. In this table, the segmentation type classification is based on the criteria proposed by the original authors of each method. However, by changing these criteria, it's relatively easy to create variations of the original methods, which produce other types of segmentation. For example, while the method by Lai et al [17] uses a criterion based on the concavity of the edges, it is straightforward to combine the same random walk methodology with a planarity criterion, in order to produce a surface-based segmentation of the input object.

[146] Sander, P. V., Snyder, J., Gortler, S. J., & Hoppe, H. (2001, August). Texture mapping progressive meshes. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques (pp. 409-416). ACM.

Methodology	Reference		Type	Segmentation Criteria
Region Growing	Papaioannou et al.	[118]	Surface	Deviation of polygon normals
	Besl & Jain	[119]	Surface	Gaussian and mean curvature, distance from fitted polynomials
	Kalvin & Taylor	[121]	Surface	Planarity and normal angles
	Lavoué et al.	[122]	Surface	Curvature based
Multi-Region Growing	Sorkine et al.	[123]	Surface	Parameterization distortion
	Mangan & Whitaker	[125]	Surface	Deviation from flatness
	Lai et al.	[111]	Part	Edge concavity / Dihedral angles
Hierarchical Clustering	Garland et al.	[127]	Surface	Planarity using distance and orientation norms
	Attene et al.	[128]	Surface	Fitting to simple primitives
Iterative Clustering	Cohen-Steiner et al.	[130]	Surface	Planarity and normal angles
Contour Extraction	Lee et al.	[132]	Part	Curvature based
	Huang et al.	[134]	Surface	Multi-scale sharpness and roughness
	Willis & Zhou	[138]	Surface+Part	Curvature based
Spectral Analysis	Zhou et al.	[139]	Surface+Part	Parameterization distortion, geodesic distances
Implicit Methods	Li et al.	[140]	Part	Skeleton based/topological
	Antini et al.	[141]	Part	Reeb Graph Connectivity
Critical Points based	Katz et al.	[143]	Part	Centricity
	Lin et al.	[144]	Part	Geodesic and Angular distances from critical points
	Agathos et al.	[145]	Part	Centricity

Table 12. General overview of the automatic segmentation methods.

10. PAIRWISE MATCHING METHODS

10.1. Jig-saw Matching

The case of Jigsaw puzzles is probably the most common case of puzzles and the literature on computational methods of reconstruction goes back to 1964 to the work of Freeman et al. [147]. In this section we will review only the piece-wise matching process of the puzzle pieces, as the global solution of puzzles is presented in Section 11.

The first automatic solver that could handle large puzzles (100 or more pieces) appeared in 1988 by Wolfson et al. [148]. The authors used the curve matching algorithm of Schwartz-Sharir [149] in order to match the boundary curves of the pieces. Kosiba et al. [150] are the first to exploit both the shape and the image color information. Color similarity of adjacent pieces is favored, by using a color compatibility metric along the matching contour. In a similar approach, Chung et al. [151] give penalty to color mismatch across the boundary by calculating the squared color disagreement. The work of Goldberg et al. [152] in 2002 while utilizing only shape information has been for several years the state of the art in this area, achieving reassembly of a 204-piece puzzle and it was only after several years (2008) that Nielsen et al. [153] managed to solve a 320-piece puzzle, through exploitation of image features along with the shape of pieces. After that point, the focus of jigsaw puzzles was shifted from the traditionally-shaped pieces to square pieces and as a consequence all later methods focus only on the color-textural information and not the shape of pieces. The square-piece puzzles are categorized in three types based on the assumptions they make:

- Type 1 puzzles scramble the location of the pieces while the orientation is known.
- Type 2 puzzles consider both location and orientation of the pieces unknown.
- Type 3 puzzles have unknown rotation but known location for their pieces.

The work of Cho et al. [154] focuses on square-piece type 1 puzzles and exploits graphical models for the achievement of a solution. Pomeranz et al. [155], while still addressing type 1 square-piece puzzles and using compatibility metrics based on the work in [154], showed improved performance and

[147] Freeman, Herbert, and L. Garder. "Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition." *Electronic Computers, IEEE Transactions on* 2 (1964): 118-127.

[148] Wolfson, Haim, et al. "Solving jigsaw puzzles by computer." *Annals of Operations Research* 12.1 (1988): 51-64.

[149] Schwartz, Jacob T., and MichaSharir."Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves."*The International Journal of Robotics Research* 6.2 (1987): 29-44.

[150] Kosiba, David A., et al. "An automatic jigsaw puzzle solver." *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*. Vol. 1.IEEE, 1994.

[151] Chung, Min Gyo, Margaret M. Fleck, and David A. Forsyth."Jigsaw puzzle solver using shape and color."*Signal Processing Proceedings, 1998.ICSP'98. 1998 Fourth International Conference on*. Vol. 2.IEEE, 1998.

[152] Goldberg, David, Christopher Malon, and Marshall Bern. "A global approach to automatic solution of jigsaw puzzles." *Proceedings of the eighteenth annual symposium on Computational geometry*.ACM, 2002.

[153] Nielsen, Ture R., Peter Drewsen, and Klaus Hansen. "Solving jigsaw puzzles using image features." *Pattern Recognition Letters* 29.14 (2008): 1924-1933.

[154] Cho, Taeg Sang, ShaiAvidan, and William T. Freeman. "A probabilistic image jigsaw puzzle solver."*Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*.IEEE, 2010.

[155] Pomeranz, Dolev, Michal Shemesh, and Ohad Ben-Shahar."A fully automated greedy square jigsaw puzzle solver."*Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*.IEEE, 2011.

achieved solution of a 3300-piece puzzle by mainly altering their global reconstruction step. The first paper that addresses the harder type 2 square-puzzle problem is the work of Gallagher [156] that uses the so called *"Mahalanobis Gradient Compatibility"* measure. This work is also discussed in section 11, for its multi-piece reassembly approach. In the rest of this sub-section we will present two matching methods that are based on the color criteria of [154] and [156], since curve matching is thoroughly studied in *Section 10.2* that focuses on 2D fragment matching.

10.1.1. A Probabilistic Image Jigsaw Puzzle Solver

Cho et al. [154] focus on solving image jigsaw puzzles with square pieces of type 1 (jig swap puzzle) and evaluate pair compatibility using color metrics, including natural image statistical measures.

Compatibility $P_{i,j}(x_j|x_i)$ is the measure that represents the likelihood of a patch x_j to appear next to x_i. There are four types of compatibility for each pair of patches corresponding to the placement direction of x_j next to x_i (left/right/top/bottom). Five types of compatibility measures were tested:

- **Dissimilarity-based Compatibility:** The dissimilarity of patches is defined as the squared color difference along the adjoining boundaries of two patches.

$$D_{LR}(x_j, x_i) = \sum_{k=1}^{K} \sum_{l=1}^{3} \left(\mathbf{x}_j(k, u, l) - \mathbf{x}_i(k, v, l)\right)^2$$

  where $\mathbf{x}_j$, $\mathbf{x}_i$ are regarded as $K \times K \times 3$ matrices, where K is the number of pixels per column (side), u indexes the last column of $\mathbf{x}_j$ (patch x_j), and v indexes the last column of $\mathbf{x}_i$ (patch x_i). The color difference is computed in the normalized LAB color space (chrominance components are normalized to have the same variance as the luminance component) and the squared difference is converted to a probability by exponentiation of the color difference:

$$P_{i,j}(x_j|x_i) \propto \exp\left(-\frac{D(x_j, x_i)}{2\sigma_c^2}\right)$$

  where σ_c is adaptively set as the difference between the smallest and the second smallest $D(x_j, x_i)$ among all x_j. This compatibility measure, although naïve in its approach, outperforms the other more sophisticated ones.

- **Boosting-based compatibility:** In this method, a boosting classifier [157] is trained to identify matching edges by deriving a feature vector from boundary pixels. A two pixel band is taken from each patch and the sum of the squared difference of all pairwise 2-pixel bands is calculated. This captures the correlation between pixels at the adjoined boundary. For K pixels per column the feature vector is $3 \times 4K^2$. The classifiers are trained using a Gentle Boost algorithm and the margin is used at the compatibility measure.

- **Set-Based compatibility:** This measure is inspired by the bidirectional similarity measure introduced in [158]. The set dissimilarity is the minimum sum of squared color differences of the adjoined boundary of two patches x_i, x_j, and all other patches in the database. The distance is exponentiated as in the Dissimilarity-based Compatibility measure, in order to be used as a compatibility measure. Under this measure, a patch pair is compatible if the boundary of the

[156] Gallagher, Andrew C. "Jigsaw puzzles with pieces of unknown orientation." *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*.IEEE, 2012.

[157] Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. "Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)." *The annals of statistics* 28.2 (2000): 337-407.

[158] Simakov, Denis, et al. "Summarizing visual data using bidirectional similarity." *Computer Vision and Pattern Recognition, 2008.CVPR 2008.IEEE Conference on*.IEEE, 2008.

patches is similar to one of the patches in the database and the authors sample half the boundary region from the first patch and half the boundary from the second one.

- **Image statistics-based compatibility:** The $K \times K$ patch at the adjoining boundary is convolved using the set of image filters presented by Weiss and Freeman [159]. Patch pairs with a small filter response at the boundary are given high score of compatibility. This and the Set-Based compatibility did not perform well in the authors' test. That can be attributed to the fact that learning-based compatibility metrics measure how natural the boundary regions are and do not necessarily preserve the likeliness rating.
- **Compatibility metric used by Cho et al. [**160**]:** This compatibility metric combines the dissimilarity-based compatibility and the image statistics-based compatibility by multiplication. This metric is useful for finding visually pleasing patch matches other than the correct match and is also useful for image editing purposes.
-

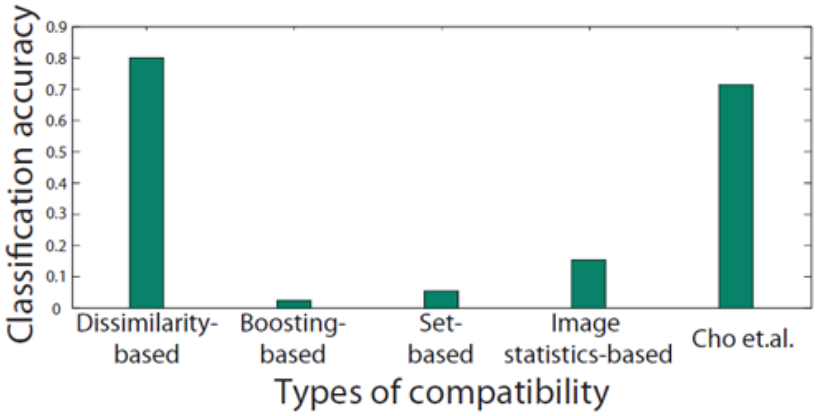


Figure 39: Evaluation of the five compatibility metrics. The dissimilarity-based metric is the most discriminative. Figure taken from [154].

10.1.2. Jigsaw Puzzles with Pieces of Unknown Orientation

The work of Gallagher [156] is the first to address the harder type 2 square-puzzle problems, where no assumption is made over the final orientation and location of the pieces. The measure used to validate adjoined jigsaw pieces is the so called *"Mahalanobis Gradient Compatibility"* (MGC). MGC compares the color similarity near the boundaries of two patches by penalizing changes in the intensity gradients between adjacent pieces, rather than penalizing the intensity dissimilarity itself. Furthermore, instead of using the Euclidean distance and penalize all deviations from a constant gradient uniformly, Gallagher uses the Mahalanobis distance to take into account the non-uniform distribution of the color gradients across the three color channels. In essence, the method tries to favor patch configurations where the intensity gradient at the boundary is maintained as we move towards the interior of the patches.

In order to compute the compatibility metric $D_{\mathrm{LR}}(x_i, x_j)$ of a jigsaw piece x_i to another piece x_j that lies on the right side of x_i, the authors calculate first the distribution of the color gradients near the right edge of x_i. G_{iL} is defined as the array of gradients with 3 columns (one per color) and P rows. In essence, G_{iL} describes the intensity changes along the right side of x_i:

[159] Weiss, Yair, and William T. Freeman. "What makes a good model of natural images?." *Computer Vision and Pattern Recognition, 2007.CVPR'07.IEEE Conference on*.IEEE, 2007.

[160] Cho, Taeg Sang, et al. "The patch transform and its applications to image editing." *Computer Vision and Pattern Recognition, 2008.CVPR 2008.IEEE Conference on*.IEEE, 2008.

$$G_{iL}(p,c) = x_i(p,P,c) - x_i(p,P-1,c)$$

The mean distribution of these gradients on the right side of x_i is:

$$\mu_{iL} = \frac{1}{P}\sum_{p=1}^{P} G_{iL}(p,c)$$

The 3×3 covariance estimated from G_{iL} captures the relationship of the gradients near the edge of the piece between the color channels and is referenced to as $\mathbf{S}_{iL}$. The compatibility metric is defined as:

$$D_{LR}(x_i,x_j) = \sum_{p=1}^{P}\left(G_{ijLR}(p) - \mu_{iL}\right)\mathbf{S}_{iL}^{-1}\left(G_{ijLR}(p) - \mu_{iL}\right)^T$$

where $G_{ijLR}(p)$ is the gradient from the right side of piece x_i to the left side of piece x_j at row p:

$$G_{ijLR}(p,c) = x_j(p,1,c) - x_i(p,P,c)$$

In the same way, $D_{RL}(x_j,x_i)$ evaluates the distributions from the x_j side of the boundary. The symmetric compatibility measure $C_{LR}(x_i,x_j)$ is the sum of the two:

$$C_{LR}(x_i,x_j) = D_{LR}(x_i,x_j) + D_{RL}(x_i,x_j)$$

For two adjacent pieces the authors calculate all 16 possible configurations (four possible positions, multiplied by four rotational permutations). The author calculates these configurations for all pairs of pieces and stores them in a 3D array $\mathbf{S}(x_i,x_j,r)$ with r indicating the pairwise configuration of size $K \times K \times 16$.

Error! Reference source not found. and **Error! Reference source not found.** show that the proposed similarity metric outperforms the LAB and RGB similarity metrics presented in [154] and [155], respectively. P is the size of the piece in pixels (side length) and K is the total number of pieces. The ratio corresponds to the correct matches found.

	P = 14			**P = 28**		
	K = 221	**K = 432**	**K = 1064**	**K = 221**	**K = 432**	**K = 1064**
RGB SSD	0.682	0.649	0.621	0.828	0.790	0.863
LAB SSD	0.676	0.634	0.606	0.826	0.788	0.859
MGC	0.816	0.785	0.771	0.919	0.902	0.942

Table 13: Performance of compatibility measures of [155]-*RGB* [154]-*LAB* [156]-*MGC on type 1 puzzle pieces. Data taken from* [156].

	P = 14			**P = 28**		
	K = 221	**K = 432**	**K = 1064**	**K = 221**	**K = 432**	**K = 1064**
RGB SSD	0.596	0.569	0.542	0.782	0.740	0.832
LAB SSD	0.591	0.554	0.525	0.780	0.738	0.827
MGC	0.757	0.712	0.703	0.902	0.879	0.933

Table 14: Performance of compatibility measures of [155]-*RGB* [154]-*LAB* [156]-*MGC on type 2 puzzle pieces. Data taken from* [156].

10.2. Two-Dimensional Matching

While all real objects have a third dimension, for certain flat objects such as frescos and stone tables it is safe to make a simplification and reduce the problem to two-dimensions (2D). Many methods have been proposed for the 2D reassembly problem and most of them address it as a 2D planar curve matching, using the outline as viewed from the top side of the fragment. Stolfi and Leitao [161] uniformly sample the contour of the fragments in various scales storing for each point a curvature value. Matching is performed using a coarse-scale representation and later it is iteratively refined using an elastic curve matching approach. The multi-scale elastic curve matching approach was initially used by Kong and Kimia [162] but in that work, the representation of the contour was based on a polygonal non-uniform approximation. Similar to [161] is the work of Amigoni et al. [163], which also use a uniform sampling of the contour and a curvature representation; however in this approach, the authors also utilize the color of the boundary of the fragments to validate the matching results. McBride and Kimia [164] identify the contour corners as critical points and use them to guide the elastic curve matching. Papaodysseus et al. [165] also use a polygonal approximation to represent a fragment's contour and calculate the area left between two contours in terms of pixels, as the matching metric. Tsamoura and Pitas [166] exploit the contours' color in order to quickly discard several potential matching pairs. Sagiroglu and Ercil in [167] and [168] propose a different solution to the problem using only color and textural features by exploiting texture synthesis and in-painting techniques. The matching and alignment of the pieces is carried out using an FFT-based image registration technique.

A lot of research in these types of problems has been also carried in the field of forensics, where the similar problem of shredded/torn-up document reassembly is encountered [169], [170], [171]. Biswas

[161] da Gama Leitão, Helena Cristina, and Jorge Stolfi. "A multiscale method for the reassembly of two-dimensional fragmented objects." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.9 (2002): 1239-1251.

[162] Kong, Weixin, and Benjamin B. Kimia. "On solving 2D and 3D puzzles using curve matching." *Computer Vision and Pattern Recognition, 2001.CVPR 2001.Proceedings of the 2001 IEEE Computer Society Conference on*.Vol. 2.IEEE, 2001.

[163] Amigoni, Francesco, Stefano Gazzani, and Simone Podico. "A method for reassembling fragments in image reconstruction." *Image Processing, 2003.ICIP 2003.Proceedings.2003 International Conference on*.Vol. 3.IEEE, 2003.

[164] McBride, Jonah C., and Benjamin B. Kimia."Archaeological fragment reconstruction using curve-matching."*Computer Vision and Pattern Recognition Workshop, 2003.CVPRW'03.Conference on*.Vol. 1.IEEE, 2003.

[165] Papaodysseus, Constantin, et al. "Contour-shape based reconstruction of fragmented, 1600 bc wall paintings." *Signal Processing, IEEE Transactions on* 50.6 (2002): 1277-1288.

[166] Tsamoura, Efthymia, and Ioannis Pitas. "Automatic color based reassembly of fragmented images and paintings." *Image Processing, IEEE Transactions on* 19.3 (2010): 680-690.

[167] Sağiroğlu, M. Ş., and AytülErçil. "A texture based approach to reconstruction of archaeological finds." *Proceedings of the 6th International conference on Virtual Reality, Archaeology and Intelligent Cultural Heritage*.Eurographics Association, 2005.

[168] Sagiroglu, MahmutSamil, and AytulErcil. "A texture based matching approach for automated assembly of puzzles." *Pattern Recognition, 2006.ICPR 2006.18th International Conference on*.Vol. 3.IEEE, 2006.

[169] Biswas, Arindam, ParthaBhowmick, and Bhargab B. Bhattacharya."Reconstruction of torn documents using contour maps."*Image Processing, 2005.ICIP 2005.IEEE International Conference on*.Vol. 3.IEEE, 2005.

[170] Justino, Edson, Luiz S. Oliveira, and CinthiaFreitas."Reconstructing shredded documents through feature matching."*Forensic science international* 160.2 (2006): 140-147.

[171] Zhu, Liangjia, Zongtan Zhou, and Dewen Hu. "Globally consistent reconstruction of ripped-up documents."*Pattern Analysis and Machine Intelligence, IEEE Transactions on* 30.1 (2008): 1-13.

et al. [169] initially detect the corner points on the boundary. Then, for each pair of corners the chain code and the Minkowski Sum of the edge defined are calculated. Using these, they search for matching edges by initially aligning corner points and checking for containment of the edge under test inside the envelope defined by the Minkowski sum. Justino et al. in [170] apply initially a polygonal approximation to reduce the complexity of the fragment contour. For these polygonal curves, geometrical features are extracted and a similar approach to [161] is used in order to perform the reassembly of the document. Zhu et al in [171] estimate shape features from every fracture contour and utilize them in order to discover matching contour segments.

Below, we present the main contributions to the field of 2D reassembly. Although the reader should keep in mind that the goal of our research is generic 3D reassembly, certain ideas applied to 2D matching can inspire related algorithms in the 3D domain or assist in cases where 3D constrained matching can operate on contours embedded in space (e.g. fracture lines).

10.2.1. A Multiscale Method for the Reassembly of Two-Dimensional Fragmented Objects

Leitao and Stolfi [161] describe a procedure that encodes the fragment contours using a curvature metric. They propose a multi-scale approach by comparing the curvature-encoded fragment outlines at progressively increasing scales of resolution, using an incremental dynamic programming sequence-matching algorithm. In this way, they reduce the computational cost of a standard dynamic approach of $O(N^2L^2)$ to about $O(N^2L\log L)$, where N is the number of fragments and L is the number of samples on each one of the contours.

The metric used for the encoding of fragment outlines is the well-known curvature graph representation; it is a one-dimensional function $\kappa(t)$ of the curve arch length t that has been extensively studied for curve matching [172] and is proven from differential geometry that under sufficiently dense sampling it is invariant under rotations and translations. The authors use the same sampling step for all outlines.

The matching approach in this work is not a classical curve matching technique as the fragments might be weathered or there might be errors in the acquisition of the data (scanning). An elastic curve matching approach is used, which has been proved to be much more robust in these types of data. The matching defines a correspondence between two outlines as a collection of pairs of samples. An index pair (r, s) defines a point-to-point correspondence between the two outline segments but allows a many-to-one mapping between curve points (see *Figure 40*).

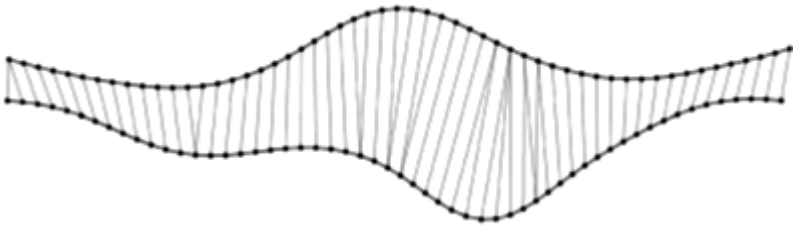


Figure 40: Pairing between samples of two fragments outlines. Figure taken from [161].

Given two segments $\alpha = (a_0 \dots a_m)$ and $b = (b_0 \dots b_m)$ a pairing between them is defined as a pair (r, s) of index sequences $r_k \in \{0, \dots, m\}$, $s_k \in \{0, \dots, n\}$, $k \in \{0, \dots, p\}$, such that

$$(r_{k+1} - r_k, s_{k+1} - s_k) \in \{(0,1), (1,1), (1,0)\} \quad \forall k \in \{0, \dots, p-1\}$$

To perform the matching the authors calculate the quadratic mismatch of a pair:

$$S^2(a, b, r, s) = Y^2(a, b, r, s) + Z^2(r, s)$$

[172] Wolfson, Haim J. "On curve matching." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 12.5 (1990): 483-489.

where:

$$Y^2(a,b,r,s) = \frac{1}{4}\sum_{k=0}^{p-1}(\varepsilon_k + \varepsilon_{k+1})(\tau_k + \tau_{k+1})$$

$$Z^2(r,s) = \frac{\zeta^2}{2}\sum_{k=0}^{p-1}|(r_{k+1} - r_k) - (s_{k+1} - s_k)|$$

ζ^2 is a constant for the penalty for each asymmetric step, ε_k is some distance metric and $\tau_k = r_k + s_k$.

The term $Y^2(a,b,r,s)$ measures the total difference between corresponding sample values of the two segments and the term $Z^2(r,s)$ is used to penalize pairings that are too irregular (zero for one-to-one pairing).

Given a fixed number n, the smaller the value of $S^2(a,b,r,s)$ the more likely it is that the segments a,b are adjacent and that (r,s) is the true correspondence between their samples. Assuming a fixed segment $n = m$, with one-to-one pairing the authors using Bayesian analysis determine a critical value Ξ^2 of S^2 such that the candidate is more likely to be false if $S^2(a,b,r,s) > \Xi^2$ and true if $S^2(a,b,r,s) < \Xi^2$.

The authors found that the critical value of S^2 is of the form:

$$\Xi^2 = (n - n_{min})\xi^2,$$

where ξ^2 is a constant value depending only on the nature of the fragments and n_{min} is proportional to $\log N$ and represents the minimum candidate length for reliable matching. The parameter ξ^2 is the critical sample mismatch and is the value that separates true candidates from false ones provided they are sufficiently long ($n \gg n_{min}$).

The authors' experiments showed that a critical value Ξ^2 exists, at which the candidate is equally likely true or false and still varies according to the formula:

$$\Xi^2 = \left(\frac{m+n}{2} - n_{min}\right)\xi^2,$$

for suitable constants ξ^2 and n_{min}. Based on that analysis the matching decision criterion is defined as:

$$\Delta_*(a,b) = S_*^2(a,b) - \xi^2\big((m+2)/2 - n_{min}\big)$$

A candidate (a,b) is considered valid if $\Delta_*(a,b) < 0$.

To minimize the computational complexity that would arise from many finely sampled segment contours, the authors utilize a multi-scale approach. Coarse fragment fracture curve representations, which exhibit a lower computational cost, are used in order to quickly discard incorrect pairs. With geometrically smaller sampling steps, the results are refined until a final set of candidates are identified. A Gaussian filter bank is used in all progressive refinement steps, with increased cutoff frequency, as the step of sampling gets smaller to avoid aliasing artifacts.

10.2.2. Archaeological Fragment Reconstruction Using Curve-Matching

McBride and Kimia [164] present another method for the 2D fragment reassembly. In order to address the fact that unconstrained curve matching is computationally expensive, the authors consider only matches that begin on fragment corners and perform curve-matching with normalized energy to determine how far the matching extends. To further reduce the cost of the matching operations, a multi-scale approach is used. Using coarse scales, possible matches are generated and only the best of them are used for matching at finer scales.

For all contours, curvature is calculated and by locating the extrema the authors extract the corner points. The matching of the sub-segments defined by the corner points is performed in four scales. The coarser representation of the contour contains only the corner points. Second and third representations

are obtained by re-sampling the sub-segments at even intervals of arc-length across the original contour, while the fourth representation is the original set of points from the fragment's contour.

The matching on the first level (coarser) representation is performed using the elastic curve matching method described in the work of Sebastian et al. [173]. On the second level of representation, given the start points, (obtained by the first level matching) the authors determine a set of end points by balancing similarity and extent of the matching to obtain locally optimal sub-contours (see *Figure 41*). In order to achieve that, the method of Kong and Kimia [162] that uses predefined energies for stretching and bending is used.

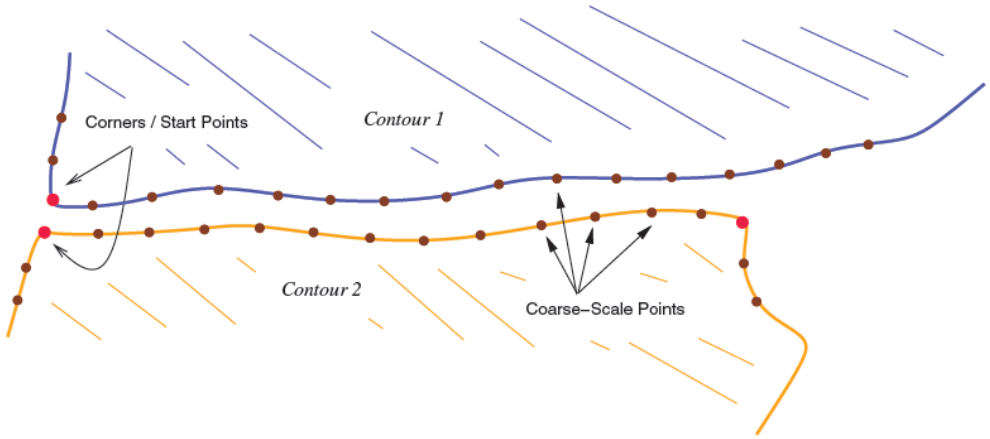


Figure 41: Second level sub-segment matching. Image taken from [164].

From the set of possible candidates obtained so far, the top *k* are selected for the third level representation matching. The goal at this level is to get a more accurate measure of similarity and that is performed by using the open curve matching method of [80]. At the fourth representation level (finest scale) an optimal Euclidean registration is obtained using least-squares on the already alignment segments. Once the optimal registration is obtained, a cost metric is applied to measure the pairwise affinity. The metric consists of three distinct parts and in general favors length and complexity of matching. The formula is:

$$c_{total} = \lambda_1 c_{distance} + \lambda_2 \sqrt{c_{length}} + \lambda_3 \sqrt{c_{diagnostic}}$$

where $c_{distance}$ measures the similarity of two sub-contours and is equal to the average distance between corresponding points. c_{length} is given by the arc length of the common boundary and accounts for the fact that longer matches give higher confidence that the matching is correct. $c_{diagnostic}$ measures the complexity of the common boundary and accounts for the fact that more complex boundaries give higher confidence. The square-root relation is used on c_{length} and $c_{diagnostic}$ so that the length or the complexity of a poor match do not obscure the matching score.

10.2.3. Globally Consistent Reconstruction of Ripped-Up Documents

The work of Zhu et al. [171] on the reconstruction of ripped-up documents performs curve matching, similar to [161] and [164], but considers a global approach to disambiguate the candidate matches, instead of relying on best pairwise matches. The rationale behind this approach is based on the fact that, although some incorrect pairwise matches may score higher than a correct one, the global score of the correct configuration of the entire piece set will probably be much higher than that of an invalid one. The relationships among all candidate matches are exploited to search for a globally consistent solution for reconstructing the original document. This way, the problem of disambiguating candidate

[173] Sebastian, Thomas B., Philip N. Klein, and Benjamin B. Kimia."On aligning curves."*Pattern Analysis and Machine Intelligence, IEEE Transactions on* 25.1 (2003): 116-125.

matches is then reduced to global optimization problem, which is solved as a nonlinear optimization procedure with boundary constraints. More specifically, the relaxation procedure is a gradient projection method, implemented in an iterative manner over the L ripped up pieces. After the maximization procedure converges, the N_a candidate matches with confidences of 1 are merged. The overall process starts again with $L-N_a$ fragments and continues until N_a=L or none fragments have been merged.

The piecewise matching itself is interesting too, since it relies on a histogram-based approach to spot congruent curve segments. In more detail, the partial contour matching is as follows. Let $F = \{F_1, \dots, F_L\}$ be the set of L document fragments. Initially, turning functions $\theta(s) = \{\theta_1(s), \dots, \theta_L(s)\}$, i.e. curve turning angles as a function of arch length, are defined for each one of the fragment contours and the matching segments between two fragment contours are found by using a turning function-based curve matching.

Let $M = \{M_1, \dots, M_N\}$ be the properly matched fragments. Let $\varphi_A = (\alpha_1, \dots, \alpha_m)$ and $\varphi_B = (b_1, \dots, b_n)$ be two strings to be compared where $m \leq n$. The turning-function-based partial curve matching is performed as follows in four steps:

- Step 1: The shortest string φ_A is circularly shifted by d positions and the corresponding shifted trail substring in φ_B is denoted by $\varphi_B^d = (b_{1+d}, \dots, b_{m+d})$. Then, the differences $\Delta\varphi_{AB}^d$ between φ_A and φ_B^d are computed as:

$$\Delta\varphi_{AB}^d = \varphi_B^d - \varphi_A \triangleq (b_{1+d} - \alpha_1, \dots, b_{m+d} - \alpha_m)$$

  Subsequently, a histogram of $\Delta\varphi_{AB}^d$ is generated by sampling it with equal spacing Δt and counting the number of points that lie in each interval.

- Step 2: Search for the pairs of starting and ending points. This boils down to searching for the clear-cut peaks in the histogram (see *Figure 42*). The rationale for this is that if two contour segments match, there should be a large amount of points clustered near or within (depending on the histogram granularity) a high-scoring difference bin. Since longer or jagged matching pairs usually convey greater confidence that the shorter or the almost straight ones, only those pairs of starting and ending points with their length greater than t_l and $N_c > t_c$ are selected where:

$$N_C = \sum_{k=w}^{w+l-1} \zeta_\kappa, \text{ where } \zeta_\kappa = \begin{cases} 1, & \text{if } |a_{k+1} - a_k| > t_a \\ 0, & \text{otherwise} \end{cases}$$

  t_l, is a constant that controls the minimum length of the segments. t_a, t_c are constants that control the tolerance of segment straightness allowed for curve matching. $\alpha_w, \dots, \alpha_{w+l}$ the set of points under evaluation. N_c corresponds to the number of "interesting" (jugged) features along the segment.

- Step 3: The Euclidean transformation and similarity between the possible matching segments is calculated. Let X, Y be the two segments. The authors use the approach of [172] in order to calculate the Euclidean transformation. Using the optimal transformation $\mathbf{E}_{opt}$, segment X is transformed to X' where X' and Y are evenly sampled and represented by two sequences $(u_1, \dots, u_{k1})$ and $(v_1, \dots, v_{k2})$. The metric evaluating the matching score of the curves is:

$$S_{X'Y} = \frac{\sum_{i=1}^{k_1} d(u_i - Y) + \sum_{j=1}^{k_2} d(v_j - X')}{(\min\{l_1, l_2\})^2}$$

  where $d(u_i - Y) = \min_{\forall u_j \in Y} \|u_i - v_j\|$, and l_1, l_2 the length of each segment respectively.

- Step 4: Shift φ_A, one position further along φ_B in counter clockwise direction $(d \leftarrow d + 1)$ and repeat Step 1 to Step 3 until d is equal to the length of φ_B.

The best candidates from each one of the above iterations are accumulated and ranked according to matching score. The described matching method requires $O(n^2)$ operations for comparing two

fragment contours where n the number of sample points of the longest contour. Therefore $O(L^2n^2)$ is the computational complexity of the method for finding the candidate matches for L fragments.

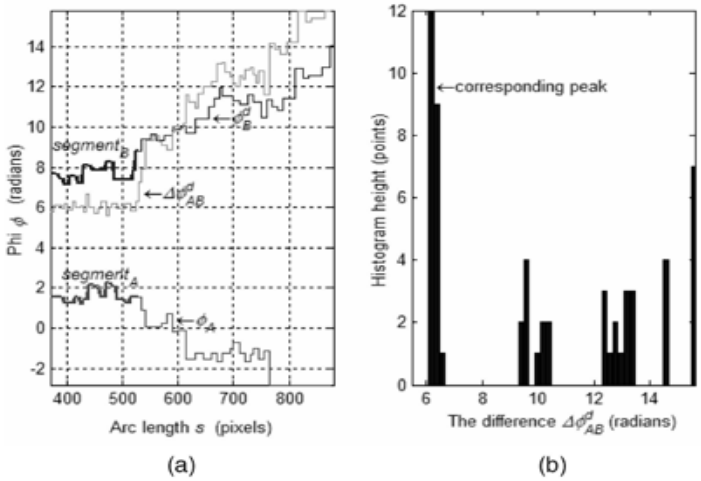


Figure 42: Search for the pairs of starting and ending points. a) Difference between $\boldsymbol{\varphi_A}$, $\boldsymbol{\varphi_B^d}$. b) Clear-cut peaks in the histogram. Figure taken from [171].

10.2.4. A Texture Based Matching Approach for Automated Assembly of Puzzle

Sagiroglu and Ercil [168] present an approach to the 2D fragment assembly that is based on a texture prediction algorithm, inspired by the fact that in many situations, although geometric detail at fracture lines may be missing, continuity of textural patterns can still guide the (manual) matching process. Their method predicts the pixel values in a band outside the boundary of the pieces. Features obtained from the expanded texture outside a fragment are correlated with original pictorial patterns of potentially neighboring pieces. First, a band of pixels around the border of each fragment is predicted using a mixture of in-painting and texture synthesis techniques. Texture features are derived from both the original region and the predicted one and, using FFT shift theory, the authors try to find a solution that maximizes the correlation between the predicted parts of a piece and other pieces. The main idea behind this approach is that the texture on the expanded part of a piece will match the original pattern of the matching piece(s).

The authors use the method proposed in [19] to predict the pixel values of the band around the fragment's border: Given a source region I_m^0 for the *m*-th piece, a target band I_m^+ extending outwards from the *m*-th piece is defined so that a new, larger image area is produced: $I_m = I_m^0 + I_m^+$. The border δI_m between I_m^0 and I_m^+ evolves outwards as an inpainting algorithm progresses (*Figure 43*). The three main steps of the inpainting algorithm that are presented below, are iterated until a target region is filled:

- Step 1: A priority P that determines the order in which the pixels are filled is calculated. It mainly depends on a bias towards the continuation of strong edges D and the confidence of neighbor pixels C:

$$P(p) = D(p).C(p)$$

where

$$C(p) = \frac{\sum_{q \in \Psi_p \cap I_m^0} C(q)}{|\Psi_p|}, \quad D(p) = |\nabla I_p^{\perp}.n_p|$$

Ψ_p is the patch centered at point p and $|\Psi_p|$ is the area of it. n_p is the orthogonal unit vector to the front δI_m at point p and $\perp$ is the orthogonal operator. This confidence metric measures the reliability of a region or a pixel and affects the filling order during the in-painting.

- Step 2: After all priorities have been computed, the propagation process begins in order to fill the target band. Direct sampling of the source region is used with the most similar patch for sampling given as:

$$\Psi_{q'} = arg\ max_{\Psi_p \in I_m^0} d(\Psi_{p'}, \Psi_q)$$

with $d(\Psi_{p'}, \Psi_q)$ being the distance between the already filled pixels of patches at points p' and q.

- Step 3: The last step is the update of the confidence values that are affected by the filling of the patch. This region is limited by the neighbors of the point p'.

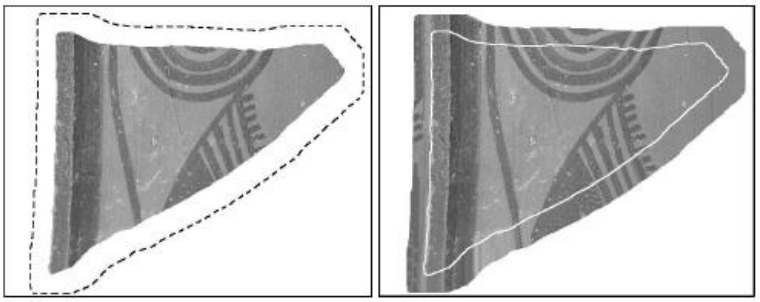


Figure 43: The original fragment and the expanded one Figure taken from [168].

For the fragment matching procedure, to avoid a pixel-by-pixel comparison the authors extract first and second moments (mean and variance of pixel intensity) from the source and expanded regions, for each piece after the prediction of the target band. The features are calculated in a window with size proportional to the resolution of the pictures of the fragments. The confidence of the feature depends on the confidence of all the pixels in the window used for its computation. The metric used to evaluate the matching of fragments is the Euclidean distance for all features.

The authors use the FFT shift theory to find a solution that will maximize the correlation between the predicted pairs of pieces. The solution for a pair of fragments consists of the base fragment I_o^0 and the transformed version of the second fragment I_1^0, where the transformation consists of both translation and rotation $\mathbf{T}_i = (\Delta x_i, \Delta y_i, \Delta\theta_i)$. The best match between the two pieces is given by:

$$S_2 = \left\{ \underset{T_1}{\text{argmax}} \sum_k^{n_k} C\left(f_{k0}, \mathbf{T}_1(f_{k1})\right) | C\left(I_0^0, \mathbf{T}_1(I_1^0)\right) = 0 \right\}$$

where C denotes the correlation operator, f_{ki} denotes the *k*-th feature of the *i*-th piece and n_k is the number of features. $C\left(I_0^0, \mathbf{T}_1(I_1^0)\right) = 0$ expresses the physical constraint that two pieces cannot overlap and $C\left(f_{k0}, \mathbf{T}_1(f_{k1})\right)$ denotes the correlation between I_1 and I_0. To find the solution set of correlations the authors use FFT operations and the solution is transformed into:

$$S_2 = \text{imax}\left[\sum_k^{n_k} \bar{F}\left(\frac{F(f_{k0}) \cdot F^*(f_{k1})}{|F(f_{k0})| \cdot |F^*(f_{k1})|} \right) \cdot L\left[\bar{F}\left(F(I_0^0) \cdot F^*(I_1^0) \right) \right] \right]$$

where $L[\text{x}] = \begin{cases} 0 & \text{if x} \neq 0 \\ 1 & \text{if x} = 0 \end{cases}$, F, F^* and $\bar{F}$ denote the Fourier operator, its complex conjugate and the inverse Fourier operator in respect. The steps of the process can be seen in *Figure 44*, *Figure* ***45*** and *Figure* ***46***.

As the FFT solution can only solve the translation transformation, the rotation is solved iteratively with the help of polar coordinates as described in the work of Wolberg and Zokai [174].

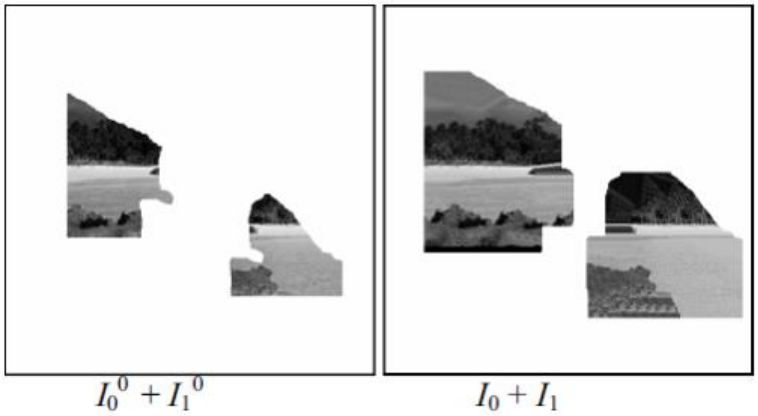


Figure 44: Original (left) and expanded (right) pieces. Image taken from [168].

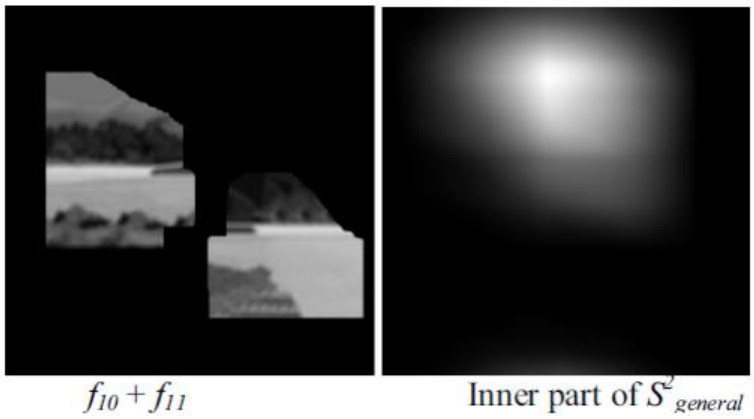


Figure 45: Mean feature (left) and correlation matrix (right) without the overlap constraint. Image taken from [168].

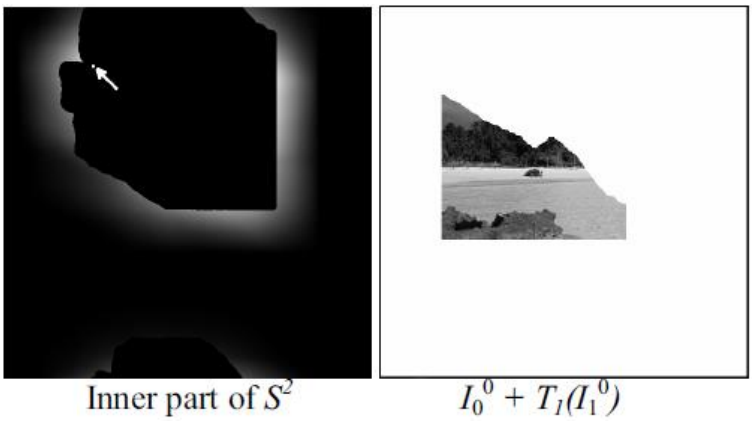


Figure 46: Correlation after constraint [left]. The arrow indicates the maximum point in the matrix. The solution after translation is applied to the second piece. Image taken from [168].

[174] Wolberg, George, and Siavash Zokai. "Robust image registration using log-polar transform." *Image Processing, 2000. Proceedings. 2000 International Conference on*. Vol. 1. IEEE, 2000.

10.2.5. Automatic Color Based Reassembly of Fragmented Images and Paintings

Tsamoura and Pitas [166] propose a three-step method for the reassembly of fragmented images. They first perform a combination reduction based on the content of the image pieces and then contour matching, using color similarity is used for the matching evaluation of the possible pairs.

In the first step, *L* potentially adjacent image fragments for each image piece are identified in order to reduce the computational cost of the next step (pairwise matching). This is performed by utilizing techniques that are widely used in content based image retrieval systems (CBIR). In the second step, matching contour segments of adjacent image fragments are discovered. The authors use an approach based on the work of Smith and Waterman [175]. In the third and final step, the alignment of the fragments contour is performed using a variant of the ICP algorithm.

For the discovery of potentially adjacent image fragments, the authors utilize high color similarity measures. Initially, colors are quantized using the commercial color palette Gretagh Macbeth Color Checker [176]. Four matching metrics were tested in order to identify the potentially adjacent fragments. These are the first and second norms, a histogram intersection and the Spatial Chromatic Histogram proposed in [177] which provides information of color presence and color spatial distribution. Both norms and the histogram intersection measures were scaled to the range $[0, 1]$, where 1 denotes a perfect similarity.

Given an image I_1, the normalized histogram h_1 counts in each bin $h_1(i)$ the number of pixels having color i divided by the total number of pixels. Let $b_1(i)$ be a 2D vector expressing the center of mass of an image piece and $\sigma_1(i)$ be the standard deviation of the *i*-th color label. The expressions of the utilized metrics follow:

Scaled L_1 norm:

$$d_{L_1}(h_1, h_2) = 1 - 0.5 \sum_{i=1}^{C} |h_1(i) - h_2(i)|$$

Scaled L_2 norm:

$$d_{L_2}(h_1, h_2) = 1 - \frac{1}{\sqrt{2}} \sum_{i=1}^{C} (h_1(i) - h_2(i))^2$$

Scaled histogram intersection (HI):

$$d_{H_1}(h_1, h_2) = \sum_{i=1}^{C} \min\big(h_1(i), h_2(i)\big)\big(1 - h_1(i) - h_2(i)\big)$$

Spatial chromatic distance (SCD):

$$d_{SC}(I_1, I_2) = \sum_{i=1}^{C} \min\big(h_1(i), h_2(i)\big) \times \left(\frac{\sqrt{2} - \|b_1(i) - b_2(i)\|^2}{\sqrt{2}} + \frac{\min\big(\sigma_1(i), \sigma_2(i)\big)}{max\big(\sigma_1(i), \sigma_2(i)\big)} \right)$$

[175] Smith, T. F., and M. S. Waterman."ªIdentification of Common Molecular Subsequences, º J."*Molecular Biology* 147 (1981): 195-197.

[176] Gavrielides, Marios A., Elena Sikudova, and Ioannis Pitas."Color-based descriptors for image fingerprinting."*Multimedia, IEEE Transactions on* 8.4 (2006): 740-748.

[177] Cinque, Luigi, et al. "Color-based image retrieval using spatial-chromatic histograms."*Multimedia Computing and Systems, 1999.IEEE International Conference on*.Vol. 2.IEEE, 1999.

From the authors' experiments, the metric that returned the best results was the HI. Using the HI metric, a list for each fragment is created containing the most chromatically similar fragments, which correspond to the potentially adjacent image fragments.

The fragment contour matching is performed based exclusively on information regarding the color of the contour. To avoid noise, the authors perform a quantization preprocessing step on the contour pixel values. Four quantization methods were tested: Mean Shift algorithm [178], Macbeth Palette [176], the method of Amigoni et al.[163] and Kohonen Neural Networks [179] (KNN), with the later giving the most successful results. The KNN is an unsupervised neural network that clusters input vectors without external information by using an iterative procedure based on competitive learning. In KNNs two node layers exist, the input and the output. In the former, the number of nodes is equal to the dimension of the input vectors and in the latter the number of nodes is equal to the amount of produced clusters. Each node s_i in the input layer has a weighted connection w_{ik} with every node c_k in the output layer, which is organized by means of a lattice. Given an input vector, the output node with the highest response as well as its neighboring nodes update their weight vectors.

The training procedure is performed by initially selecting a random number of pixels N_p from the input fragments and mapping them to La^*b^* color space. Subsequently, a $[3 \times C]$ KNN is defined where 3 corresponds to the dimensions of the input La^*b^* space and C to the predefined color clusters. Let $x = [x_1, x_2, x_3]$ be one of the N_p sampled pixels, the iterative learning procedure is applied as follows:

1. The output node c_j is selected, whose weight vector w_j has the highest similarity with the input vector x. The metric used in the paper is the Euclidean distance.
2. The weight vector w_k of an output node c_k is updated using:

$$\Delta_{w_k} = \gamma \Omega_{c_j}(c_k)(\|x - w_k\|)$$

where $\Omega_{c_j} = e^{\left(-\|p_k - p_j\|^2\right)/2\sigma^2}$, γ is the learning parameter $(0 < \gamma < 1)$, σ is the spread of the neighborhood around the winning node c_k and p_k, p_j correspond to places inside the lattice of an output node c_k and the winning c_j respectively. γ, and σ decrease gradually for better convergence.

The learning procedure stops either after a predefined number of iterations or when Δ_{w_k} is very small. When the training stops, the weight vector of every output vector corresponds to a cluster center.

The similarity function for the contour pixels is defined based on the color values. For the matching, a variant of the Smith Waterman dynamic programming algorithm [175] is used. Given two input sequences, the algorithm identifies the mapping function between them. For an image partitioned in N fragments the computational complexity of the algorithm is $O(N^2 \tilde{n}^2)$ where $\tilde{n}$ is the average contour length of the input image fragments.

In the end, for each image fragment f, one matching contour segment with K $(0 \leq K \leq L)$ other image fragments is retained, representing the true adjacent fragment couples among the L candidate matches from the first (statistical) step. For the final alignment, the authors evaluate 4 ICP algorithm variations to find the best geometrical transformation that aligns the contours with their matching counterparts. For more information on the ICP algorithm, please see Section 8.

[178] Cheng, Yizong. "Mean shift, mode seeking, and clustering." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 17.8 (1995): 790-799.

[179] Carpenter, Gail A., and Stephen Grossberg.*Pattern recognition by self-organizing neural networks*.The MIT Press, 1991.

10.3. Restricted Three-Dimensional (2.5D) Matching

In this section we present methods that solve the three dimensional problem of fragment matching on three-dimensional objects, for which either the actual dimensionality is less than 3 (e.g. contours or surfaces embedded in 3D space) or the degrees of freedom for the matching reduces the pose estimation transformation to a two-dimensional one. In the literature, the first set of problems usually regards *"thin-walled fragments"*, which in most cases are pottery sherds, while the latter targets *"flat pieces with thickness"*, which are fresco fragments in their majority.

The area of *"thin-walled fragments"* has been extensively researched in the past years. Several methods focus on the estimation of the principal characteristics of the pottery objects (axis of rotation and profile). Willis et al. [180] try to achieve that using an algebraic model of the surface; Cao et al. [181] describe an approach using spheres of curvatures, while Yacoub et al. [182] and Kampel and Sablatnig [183] use variations of the Hough transformation. In another approach, Halir [184] proposes a multistep optimization technique using M-estimators and circle and line fitting to obtain the estimation of the principal characteristics. Other works focus on the reassembly of potteries from their thin-walled fragments. Sablatnig and Menard [185] propose a method, where fragments are classified by shape features and properties and by using a graph similarity approach they try to find matching pairs. Kampel and Sablatnig [186] presented an extension of [185] that utilizes curve matching in order to align the matching pairs. Willis and Cooper, [187] also propose a method that uses the global constraint of axial symmetry, and through a Bayesian approach, they exploit the outside surface of each sherd along with the break curves of the fragments, in order to achieve matching and alignment of fragments. The system was developed through a sequence of works [180], [188], [189], and the

[180] Willis, Andrew, Xavier Orriols, and David B. Cooper."Accurately estimating sherd 3D surface geometry with application to pot reconstruction."*Computer Vision and Pattern Recognition Workshop, 2003.CVPRW'03.Conference on*.Vol. 1.IEEE, 2003.

[181] Cao, Yan, and David Mumford."Geometric structure estimation of axially symmetric pots from small fragments."*Signal Processing, Pattern Recognition, and Applications, IASTED International Conference*. 2002.

[182] Yacoub, S. Ben, and Christian Menard."Robust axis determination for rotational symmetric objects out of range data."*21 th Workshop of the Oeagm*. Hallstatt, Austria, 1997.

[183] Kampel, Martin, and Robert Sablatnig."An automated pottery archival and reconstruction system."*The Journal of Visualization and Computer Animation* 14.3 (2003): 111-120.

[184] Halır, Radim. "An automatic estimation of the axis of rotation of fragments of archaeological pottery: A multi-step model-based approach." *Proc. of the 7th International conference in Central Europe on computer graphics, Visualization and Interactive Digital Media (WSCG'99)*.1999.

[185] Sablatnig, Robert, and Christian Menard."3d reconstruction of archaeological pottery using profile primitives."*Proc. of Intl. Workshop on Synthetic-Natural Hybrid Coding and Three-Dimensional Imaging*.1997.

[186] Kampel, Martin, and Robert Sablatnig."3D puzzling of archeological fragments."*Proc. of 9th Computer Vision Winter Workshop*.Vol. 2.Slovenian Pattern Recognition Society, 2004.

[187] Willis, Andrew R., and David B. Cooper. "Bayesian assembly of 3d axially symmetric shapes from fragments." *Computer Vision and Pattern Recognition, 2004.CVPR 2004.Proceedings of the 2004 IEEE Computer Society Conference on*.Vol. 1.IEEE, 2004.

[188] Cooper, David B., Andrew Willis, Stuart Andrews, Jill Baker, Yan Cao, Dongjin Han, Kongbin Kang et al. "Assembling virtual pots from 3D measurements of their fragments." *Proceedings of the 2001 conference on Virtual reality, archeology, and cultural heritage*.ACM, 2001.

[189] Cooper, David B., Andrew Willis, Stuart Andrews, Jill Baker, Yan Cao, Dongjin Han, Kongbin Kang et al. "Bayesian pot-assembly from fragments as problems in perceptual-grouping and geometric-learning." *Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3-Volume 3*. IEEE Computer Society, 2002.

authors still continue to present variants [190]. Oxholm and Nishino [191] use the iterative uniform sampling method of Leitao and Stolfi [161] in order to match 2D representations of the sherd contours. Additional to the torsion and curvature of the original method, they store color values of the contour as well, in order to refine the alignment of the matching segments. The reason this method is presented here and not in the traditional 2D matching is the validation of the matching segments. To evaluate the resulting surface continuity, the authors extract the surface normals at regular intervals and evaluate them in terms of gradients.

While the research in the area of *"thin-walled fragments"* is extended, there are only few methods that treat them as three dimensional objects and most of them simplify the problem to two dimensions and solve a contour matching problem. We have found two methods in the literature that exploit the thickness of the 3D fragments and both base their approach on the fact that the top-flat surface can be recognized during the scanning process. Papaodysseus et al. in [192] and [193] present a method that matches the fragments using five criteria. The first criterion is the volume of the gap between the two considered fragments, the second and third criteria measure the overlap in each possible matching position and finally the fourth and fifth criteria use calculus principles to calculate the bound for the area of the contact surfaces and the length of the contact curves. In the current report, we focus in the work of Brown et al. [194] where the authors first extract and uniformly sample the ribbon of each fragment and subsequently search for matching pairs using a 3D contour matching approach. In [195], the same authors propose a machine learning approach to the problem of matching frescos, where they evaluate the usage of several local and global descriptors of color, normal and shape. Belenguer and Vidal [196] present another approach for fresco matching that uses projective GPU depth maps to find the best rigid transformation that maximizes the contact area using the *Largest Common Point-set* (LCP) as a matching criterion. The authors pre-calculate depth maps for a set of possible orientations and, using an LOD search-scheme, perform the matching calculations in a hierarchical way, for finding both the matching orientation and the displacement distance between them.

10.3.1. Bayesian assembly of 3D axially symmetric shapes from fragments

Willis et al. [187] present a system for the automatic assembling of 3D pots from their fragments (potsherds) using the global constraint of axial symmetry. The solution uses a Bayesian approach

[190] Andrew Willis and Cooper David B. "Estimating a-priori unknown 3d axially symmetric surfaces from noisy measurements of their fragments."*Proc. 3rd Int. Symp*. 2006.

[191] Oxholm Geoffrey, and Ko Nishino. "Reassembling thin artifacts of unknown geometry."*Proceedings of the 12th International conference on Virtual Reality, Archaeology and Cultural Heritage*.Eurographics Association, 2011.

[192] Papaodysseus Constantin, Arabadjis Dimitris, Panagopoulos Michail, Rousopoulos Panayiotis, Exarhos Michalis "Automated reconstruction of fragmented objects using their 3D representation-application to important archaeological finds." *Signal Processing, 2008.ICSP 2008.9th International Conference on*.IEEE, 2008.

[193] Papaodysseus, Constantin, Dimitris Arabadjis, Michalis Exarhos, Panayiotis Rousopoulos, Solomon Zannos, Michail Panagopoulos, and Lena Papazoglou-Manioudaki. "Efficient solution to the 3D problem of automatic wall paintings reassembly." Computers & Mathematics with Applications (2012).

[194] B Brown, Benedict J., Corey Toler-Franklin, Diego Nehab, Michael Burns, David Dobkin, Andreas Vlachopoulos, Christos Doumas, Szymon Rusinkiewicz, and Tim Weyrich. "A system for high-volume acquisition and matching of fresco fragments: Reassembling Theran wall paintings." In ACM Transactions on Graphics (TOG), vol. 27, no. 3, p. 84. ACM, 2008..

[195] Toler-Franklin, Corey, Benedict Brown, Tim Weyrich, Thomas Funkhouser, and Szymon Rusinkiewicz. "Multi-feature matching of fresco fragments." In ACM Transactions on Graphics (TOG), vol. 29, no. 6, p. 185. ACM, 2010.

[196] Belenguer, Carlos Sanchez, and Eduardo Vendrell Vidal. "Archaeological fragment characterization and 3D reconstruction based on projective GPU depth maps." *Virtual Systems and Multimedia (VSMM), 2012 18th International Conference on*. IEEE, 2012.

exploiting the outside surface as well as the break curves of each potsherd. In order to match pairs of potsherds, the authors exploit a Maximum Likelihood Estimation (MLE) approach that exploits the break curve features.

The geometric information used in their approach consists of a) the outer-surface break curves, b) break curve vertices at junctions, c) axis/profile curve (see *Figure 47*) for the entire pot and portions of such a curve for individual sherds and finally d) the Euclidean transformations that align potsherds in optimal assemblies.

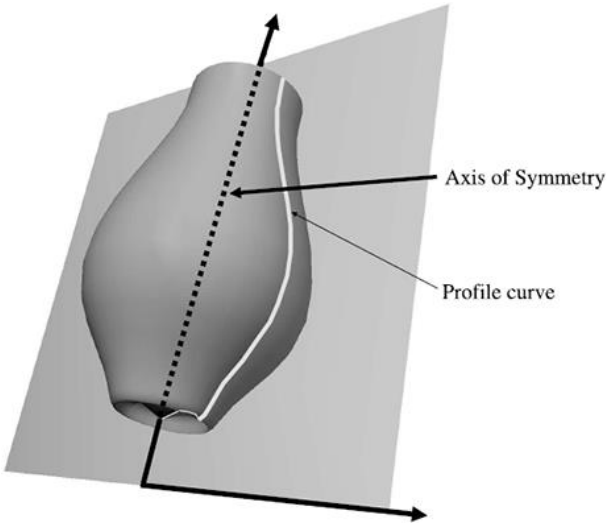


Figure 47: Axis of symmetry and profile curve. Figure taken from [190].

Break curve parameters are the locations on the pot surface along which the pot breaks. These locations include vertices of T-junctions and Y-junctions, or *break points*, as in [164] that are high-curvature points where sherd boundaries meet. The authors also define sequences of K points of increasing Euclidian distance from these vertices, the *break-point segments*, along the two halves of the contour extending from each vertex. The break-curve segments are created in a multi-scale approach using the K as parameter. For the alignment, the authors use all scales for each segment and keep only the one providing best matching. In the described approach, the authors select the possible location of junctions (vertices) manually after the scanning process.

The profile curve is modeled as an algebraic curve of degree six, using the estimation method proposed by the authors in [180]. The method parameterizes the pot axis of symmetry l using the parametric equation of a 3D line:

$$x = m_x z + b_x,$$
$$y = m_y z + b_y,$$

Therefore

$$l = (b_x, b_y, m_x, m_y)$$

where m_x, m_y specify the slope of the line when it is projected on the xz-plane and the yz-plane, respectively and b_x, b_y specify where the line intercepts the xy-plane at $z = 0$. The profile curve $a(r, z)$, with respect to l defines a $3D$ axially symmetric algebraic surface with axis l.

$$a(r, z) = \sum_{0 \le j+k \le d;\ j,k \ge 0} a_{jk} r^j z^k = 0$$

where $r = [0, \dots, K-1]$. Hence $a = (\cup_{j,k} a_{jk})$ is the vector of coefficients for the implicit polynomial curve of degree $d = 6$.

Sherd pairs are the basic building block of the assembly algorithm. In order to estimate the parameters for a sherd, a hypothesis is made on pairs of sherds. Let two distinct sherds (i, j) share a pair of common break curve segments (m, n) respectively. If the hypothesis is true, the segments are a portion of the global break curve β and the surface data from each sherd provides estimates of the pot axis l and a portion of the global profile curve α. In order to estimate these parameters one of the two coordinate systems is selected as global. Let that be the coordinate system of potsherd i, which is initialized with $\mathbf{T}_i = (t = (0,0,0), \mathbf{R} = \mathbf{I})$. $\mathbf{T}_j$ is estimated given that break-point segments m, n can be aligned by computing the MLE of $\mathbf{T}_j, l, \alpha, \beta$ for the entire set of measurement data that can be used on the common boundary of sherds i, j. Assuming a statistical independence of the noise on the sampled sets, this is expressed by the estimation of the following probability:

$$\mathrm{P}(D_i|l,\alpha)\mathrm{P}(B_{ci}|\beta)\mathrm{P}\big(D_j|l,\alpha,\mathbf{T}_j\big)\mathrm{P}\big(B_{cj}|\beta,\mathbf{T}_j\big) \qquad (1)$$

where B_{ci}, B_{cj} denote the break-point segment data for sherd i and j, respectively and D_i, D_j are the sampled data of the outer surfaces of potsherds i and j . $\mathrm{P}\big(D_j|l,\alpha,\mathbf{T}_j\big)$ denotes the probability of the data set D_j transformed by $\mathbf{T}_j$, given the surface parameterized by l and α. Similarly, $\mathrm{P}\big(B_{cj}|\beta,\mathbf{T}_j\big)$ is the probability that of B_{cj} is indeed a break curve segment after the transformation $\mathbf{T}_j$ given break-point parameters β.

Since this is a computationally expensive nonlinear problem the authors solve a simpler problem for the matching of pairs at this point and Eq. (1) is solved only when a pair is about to be combined with another one. The simple solution corresponds to computing the MLE of a projection of the higher-dimensional joint distribution:

$$\tilde{\mathbf{T}}_j = \arg\max_{T_j} \ln\Big(\mathrm{P}(B_{ci}|\beta)\mathrm{P}\big(B_{cj}|\beta,\mathbf{T}_j\big)\Big)$$

Unlike the previous approach, the maximization of the above equation is solved as a least-squares problem and has an explicit solution, which is not computationally expensive. Then the most probable values of the parameters $\tilde{l}, \tilde{\alpha}, \tilde{b}$ given $\tilde{\mathbf{T}}_j$are computed:

$$\tilde{l}, \tilde{\alpha}, \tilde{\beta} = arg\max_{l,\alpha,\beta} \ln\Big(\mathrm{P}\big(D_i, D_j, B_{ci}, B_{cj}|l,\alpha,\beta,\tilde{\mathbf{T}}_j\big)\Big)$$

The pair is assigned a preliminary match cost, which is the negative log-likelihood of the aligned sherd data given the estimated pot parameters.

$$\epsilon_{(i,j),(m,n)} = -\ln\Big(\mathrm{P}\big(D_i, B_{ci}|\tilde{l},\tilde{\alpha},\tilde{\beta}\big)P\big(D_j, B_{cj}|\tilde{l},\tilde{\alpha},\tilde{\beta},\tilde{\mathbf{T}}_j\big)\Big)$$

This quick solution is suboptimal but allows quick pruning of wrong matches.

The same approach is generalized in order to extend the matching from pairs of sherds to configurations of size *N* with the following cost function:

$$\epsilon_{\cup_w(w,cw)} = -\ln\left(\mathrm{P}\big(D_1, B_{c1}|\tilde{l},\tilde{\alpha},\tilde{\beta}\big)\prod_{w=2}^{N} P\big(D_w, B_{cw}|\tilde{l},\tilde{\alpha},\tilde{\beta},\tilde{\mathbf{T}}_w\big)\right) \qquad (2)$$

For the alignment of sherds, the authors seek to find the 3D transformation that minimizes the sum of 2 terms: a) the squared distance between the transformed points and normals on the matched break-curve segments and b) the average approximate squared Euclidean distance of the two surfaces to the axially symmetric virtual pot surface defined by the axis/profile-curve pair. This is solved using a Levenberg-Marquardt minimization scheme that is similar to the fast-ICP method, where the distance transform is replaced with the algebraic distance and the addition of an error term that involves the break parameters.

10.3.2. A System for High-Volume Acquisition and Matching of Fresco Fragments: Reassembling Theran Wall Paintings

Brown et al. [194] present a three dimensional matching algorithm for the matching of fresco fragments using only the scanned geometry. The approach takes advantage of the fragments' flat front surfaces to limit the search space to planar transformations (*Figure 48*). This way, the authors extract for each fragment the *ribbon* between the front and the back face, sample it regularly and use it for the pairwise matching.

Figure 48: A typical fresco fragment. Figure taken from [194]

In order for the matching process to be efficient, fragment edges are uniformly re-sampled into a "ribbon" (*Figure 49*). A contour is first extracted from a vertical slice below 2mm relative to the front surface, to avoid capturing the severely eroded part of the fragment's perimeter. Samples are placed every 0.25mm along the length. In order to account for noise, the extracted contour is smoothed using a standard deviation of 2.5mm and the smoothed points are re-projected back on the fragment edge using their normals. Subsequently, samples are added every 0.25mm along the z-axis (vertically) by walking from each contour point in a direction perpendicular to the arc length. The process stops when it reaches a face pointing away from the normal of the contour point. While this procedure places isolated points on the fragment's front and back surfaces, these are pruned by triangulating the ribbon and keeping only the connected components which contain a contour point.

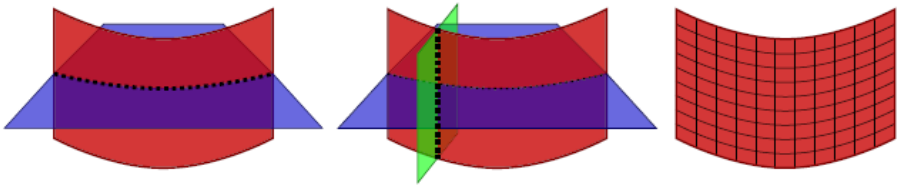


Figure 49: The described Ribbon construction process. Figure taken from [194]

Each ribbon is indexed by row and column. A strip of samples of fixed width on each fragment is picked and tested for match. The correspondences are determined by the regular sample structure and the alignment and an associated error are calculated. The overlap region is shifted then by a single sample and the process is repeated. The alignment and error metrics are incrementally computed from the previous ones using the sum of squared distances between the points of the strip. The computational cost of this process is $O(nm)$ where n and m are the edge lengths of the two fragments. The authors report an average of 2 seconds irrespective of the fragment's thickness and the width of the matching strip. They also propose and apply the following optimizations to their procedure, which take advantage of the specific geometry of the problem.

To avoid wrong correspondences due to erosion, the authors require the normal of the corresponding point to have opposite z-component even before the alignment, as the transformation is always planar. Specifically the required condition is $|n_z + n'_z| \leq 0.5$.

Based on the observation that the matching fragments have almost the same thickness by counting the vertical positions along the corresponding columns an estimate of the difference in thickness is obtained and a fixed penalty is assigned to each unmatched sample beyond a fixed per-column threshold (16 samples = 4mm).

10.4. Three-Dimensional Matching

In this section we focus on methods that handle free-form three dimensional (3D) fragments. Several approaches exist both in virtual archeology and in computer-assisted surgery but most of them are not fully automated and expect manual initial alignment or even region selection for matching, something that reduces the problem to that of surface alignment. Scheuering et al. in [197] expect manual coarse positioning and utilize a voxel-base metric for the optimal alignment. Willis et al. [198] and Zhou et al. [199] initially segment the surfaces to intact and fractured using analysis of the bone density and afterwards expect user interaction for the selection of fractured patches that coarsely correspond. Finally, ICP variants are utilized to achieve optimal alignment. Mellado et al. in [200] also expects user-specified initial position and orientation and validates the pose through a k-d tree ICP variant.

Papaioannou et al. [201] were the first to present an automatic solution to the three-dimensional object matching problem. Fragments are initially segmented to intact and fractured faces and pairwise matching for all pairs of fragments is tested using as metric the curvature difference. In [202] the same authors presented a variation of their method that utilizes contour matching and alignment in order to reduce the computational complexity, where possible. Huang et al. in [98] propose another solution for three dimensional fragment matching, where integral invariant features of multiple scales, clustered into patches are used both for segmentation and pairwise matching of fragments. Winkelbach et al. [203] propose a method for the pairwise fragment matching that initially segments the fragment into a binary tree based on the 6D coordinate-normal space and then searches for pairwise matches in depth-first order using as maximization metric the contact surface. Li et al. [204] present another method for pairwise matching of fragments. “Curvedness” is their descriptor selected for the pairwise matching, calculated at several scales using the notion of a patch. Each patch is represented using a 3D histogram of its descriptor values. Based on the histograms, similar matching point pairs are extracted and used for alignment, with a hierarchical greedy algorithm that avoids backtracks but might result in an incorrect matching. Furnstahl et al. in [205] propose a method that focuses on humerus fractures.

---

[197] Scheuering, Michael, Christof Rezk-Salama, Christian Eckstein, Kai Hormann, and Günther Greiner. "Interactive Repositioning of Bone Fracture Segments." In VMV, pp. 499-506. 2001.

[198] Willis, Andrew, Donald Anderson, Thad Thomas, Thomas Brown, and J. Lawrence Marsh. "3D reconstruction of highly fragmented bone fractures." In Medical Imaging, pp. 65121P-65121P. International Society for Optics and Photonics, 2007.

[199] Zhou, Beibei, Andrew Willis, Yunfeng Sui, Donald D. Anderson, Thomas D. Brown, and Thaddeus P. Thomas. "Virtual 3D bone fracture reconstruction via inter-fragmentary surface alignment." In Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, pp. 1809-1816. IEEE, 2009.

[200] Mellado, Nicolas, Patrick Reuter, and Christophe Schlick. "Semi-automatic geometry-driven reassembly of fractured archeological objects."*VAST 2010: The 11th International Symposium on Virtual Reality, Archaeology and Cultural Heritage*. 2010.

[201] Papaioannou Georgios, Evaggelia-Aggeliki Karabassi and Theoharis Theoharis. "Automatic Reconstruction of Archaeological Finds–A Graphics Approach." *Proc. of the 4th International Conference on Computer Graphics and Artificial Intelligence (3IA’00)*. 2000.

[202] Papaioannou Georgios and Evaggelia-Aggeliki Karabassi. "On the automatic assemblage of arbitrary broken solid artefacts."*Image and Vision Computing* 21.5 (2003): 401-412.

[203] Winkelbach Simon and Friedrich M. Wahl. "Pairwise matching of 3D fragments using cluster trees."*International Journal of Computer Vision* 78.1 (2008): 1-13.

[204] Li, Qunhui, Mingquan Zhou, and GuohuaGeng."Pairwise matching of 3D fragments."*Information Management, Innovation Management and Industrial Engineering (ICIII), 2012 International Conference on*.Vol. 3.IEEE, 2012.

[205] Fürnstahl, Philipp, Gábor Székely, Christian Gerber, Jürg Hodler, Jess Gerrit Snedeker, and Matthias Harders. "Computer assisted reconstruction of complex proximal humerus fractures for preoperative planning." Medical Image Analysis 16, no. 3 (2012): 704-720.

Initial alignment of fragments is achieved using a reference healthy bone template and the exterior surfaces of the fragments using only the exterior surface of the fragments. The initial alignment is refined using ICP. The maximization metric used is the largest connected region. In the rest of this section [202] [98] and [203] are presented in detail.

10.4.1. On the Automatic Assemblage of Arbitrary Broken Solid Artefacts

Papaioannou et al. [202] presented a method for the 3D free form fragment matching that combines curve matching and surface matching techniques. The method operates on object surface meshes of arbitrary topology that are first segmented into areas of adjacent polygons using a region growing approach and fractured sides are detected. The fractured surfaces are classified as "external" or "internal", depending on whether one or more intact surfaces are adjacent to them or not. A different matching approach applies to each category, allowing the region boundary curve to be used as a constraint in the case of external contact configurations. Pairwise matching error is calculated for all fractured facets combinations.

The "facets", i.e. the segmented regions, created by the segmentation process, are centered in an orthographic projection frustum using the average normal for alignment with the projection plane and the depth buffer for each one is rendered (*Figure 54*). From the depth map and using a Laplace image operator, the authors calculate the *bumpiness* of the surface in image space and characterize them as fractured or intact based on a predefined threshold. This approach falsely marks engraved surfaces as well, although the authors argue that this is not a problem as these facets will not be compatible with any other during the pair-wise matching stage. The bumpiness measure B_{R_k} for a facet R_k is defined as:

$$B_{R_k} = \frac{1}{N_{\text{depth}}} \sum_{\substack{(u,v)| \\ d_{R_k}(u,v)\neq\infty}} \left|\nabla^2 d_{R_k}(u,v)\right|$$

where $d_{R_k}(u,v)$ is the depth map value at image parametric location (u,v) corresponding to a pixel (i,j) of the non-background rendered region of facet R_k. N_{depth} is the number of non-background values in the depth map and $\nabla^2 d_{R_k}(u,v)$ is the common Laplacian image operator. B_{R_k} reflects the average steepness per surface facet.

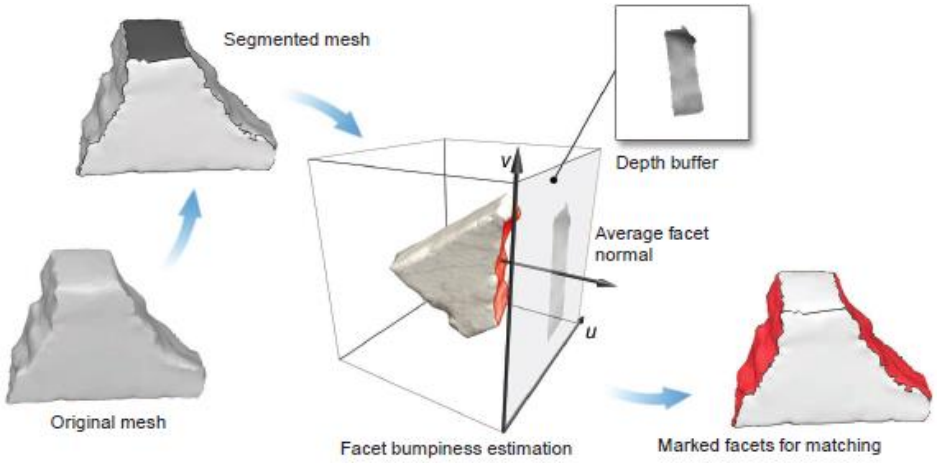


Figure 50: Fragment Segmentation – Facet Classification.

The proposed method performs a different pairwise matching approach depending on whether both fragments are external or not. In the first case, a boundary-constrained method is used, while in the latter case the unconstrained surface matching algorithm initially proposed in [206] by the same authors is utilized.

A fragment and the respective (fractured) facet is characterized as external or internal based on the fractured surface under examination. This way a fragment with multiple fragmented surfaces can be categorized both as external and internal depending on the currently selected facet. A fragment behaves as an external when the facet under examination is adjacent to an intact surface region, i.e. they share a fracture line as a region boundary. Otherwise, it is regarded as an internal part (see *Figure 51*).

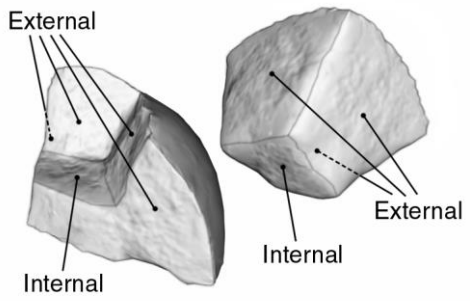


Figure 51: Characterization of facets.

For the unconstrained surface matching, the authors follow the method described in [206]; the best match between two fragments is sought and a corresponding matching error is determined. For each fractured facet combination of the two pieces, the fragments are roughly aligned according to the corresponding region normals and their relative pose is optimized based on a point-to-point curvature difference. The resulting matching error and final pose is assigned to this facet configuration and the algorithm moves to process the remaining facet combinations in the same manner.

In the proposed method, the point-to-point difference is calculated with respect to a regular grid discretization on a plane parallel to the aligned surfaces. For the measurement, the depth buffer of the orthographically projected regions on the reference plane is used. The depth buffer can be regarded as a discrete uniform distance field $d_1(u,v)$, $d_2(u,v)$ and the curvature at each node is computed in image space as the gradient of the depth buffer at this point. Therefore, the method can regularly sample the fractured surfaces at a fixed rate, regardless of their initial topology and sample density, exploiting for the curvature measurements the hardware-accelerated distance extraction (depth buffer) of the GPU.

Initially, the maximum diameter of the two fragment facets is calculated and they are moved so that their centers reside on the coordinate system origin. The fragments are rotated so that the X axes of their local reference frame point in opposite directions. Each object is rendered separately in a right-handed coordinate system with the Z-axis pointing towards plane p (see *Figure 52*). Obviously, the resolution of the depth buffer $N_u \times N_v$ at which the objects are rendered, represents the coarseness of the discretization approximation with which the computation will be performed. The resulting matching error e_d for two facets uses surface derivatives in image space:

[206] Papaioannou Georgios, Evaggelia-Aggeliki Karabassi and Theoharis Theoharis. "Reconstruction of three-dimensional objects through matching of their parts." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.1 (2002): 114-124.

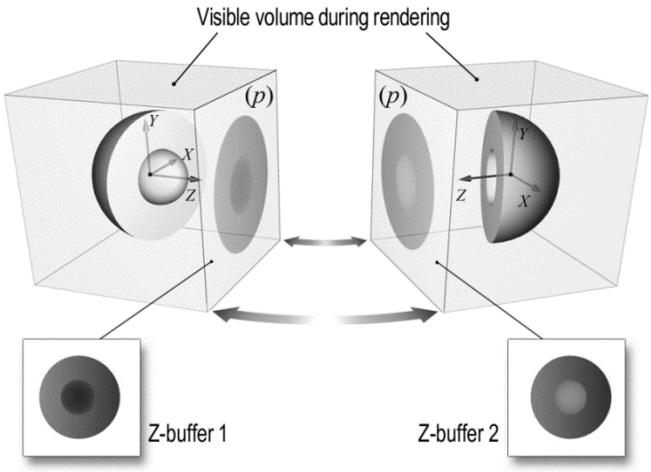


Figure 52: Use of z-buffer to calculate the point-to-point distances for the pairwise matching.

$$\varepsilon_d = \frac{1}{A_S} \iint_S \left(\left| \frac{\partial d_1(u,v)}{\partial u} + \frac{\partial d_2(u,v)}{\partial u} \right| + \left| \frac{\partial d_1(u,v)}{\partial v} + \frac{\partial d_2(u,v)}{\partial v} \right| \right) dS$$

where d_1, d_2 are the depth buffers of the two fractured surfaces, S is the buffer region where the two surfaces overlap and A_S is the corresponding area of overlap in image space. Note that the above criterion does not necessarily favor large areas of overlap.

In order to find the optimal pose, a non-linear global optimization method is used to minimize the matching error over a set of parameters. Given a fixed distance between the two fragment centers $\bar{O}_1$,$\bar{O}_2$, seven degrees of freedom are defined. Each model may rotate arbitrarily in its local orthogonal reference frame $<\theta_1, \varphi_1, \rho_1, \bar{O}_1>$ and $<\theta_2, \varphi_2, \rho_2, \bar{O}_2>$ (see *Figure 53*). ρ_2 is discarded as it is redundant and the relative displacement on the uniform grid is modeled by the translation of one of the two object by $\vec{t}_1 = (x_1, y_1, 0)$ along the sampling plane. The rotations around x and y axes are limited to small range as they represent expected deviation of the facet normal and the authors use a range of ± 10 degrees. Note that the choice of the distance between the fragment centers does not affect the measurements, since the error metric decouples the measurements for the two fragments and operates on distance derivatives instead of absolute distances.

The set of relative pose parameters is aggregated in a parameter vector $\bar{\omega} = [\theta_1, \varphi_1, \rho_1, x_1, y_1, \theta_2, \varphi_2]$. In order to minimize the matching error, the authors use the Enhanced Simulated Annealing (ESA), variation of the well-known SA non-linear optimization technique, proposed in [207], which in their experiments had an average optimum pose detection rate of 85%.

When two external fractured facets are compared, the closed boundary curve of the two facets is extracted using an image-based technique that exploits as before the depth buffer. For a fragment Obj_k with an external fractured facet $F_{k,m}$ the average normal vector $n_{ave}(F_{k,m})$ is:

$$n_{ave}(F_{k,m}) = \left\| \sum_{j|P_j \in F_{k,m}} A_j n_j \right\|^{-1} \cdot \sum_{j|P_j \in F_{k,m}} A_j n_j$$

[207] Siarry, Patrick, Gérard Berthiau, François Durdin, and Jacques Haussy. "Enhanced simulated annealing for globally minimizing functions of many-continuous variables." ACM Transactions on Mathematical Software (TOMS) 23, no. 2 (1997): 209-228.

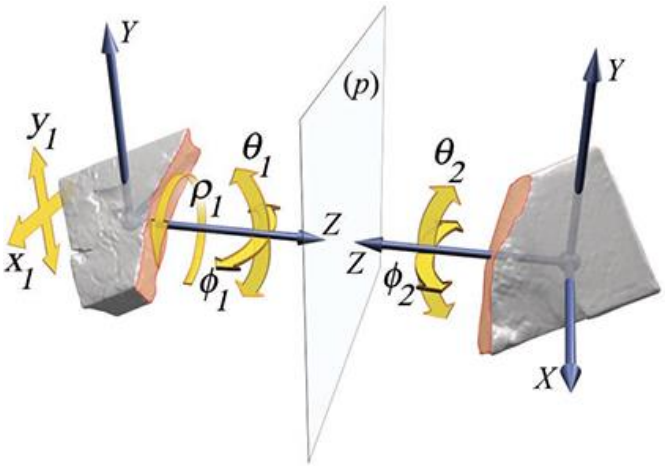


Figure 53: Relative pose of two meshes during the pairwise matching process.

where $A_j$is the area of a polygon P_j and n_j its normal. The facet $F_{k,m}$ is transformed according to $\mathbf{M}\left(n_{ave}\left(F_{k,m}\right)\right)$ so that $n_{ave}\left(F_{k,m}\right)$ coincides with the z axis and then the (orthographic) depth buffer is acquired. The outer boundary is then extracted for all non-background pixels from the depth buffer. The corresponding polyline $H_{k,m}^{(proj)}$ is extracted and transformed back to the standard coordinate system using the inverse transform: $H_{k,m} = \mathbf{M}^{-1}\left(n_{ave}\left(F_{k,m}\right)\right) H_{k,m}^{(proj)}$. To remove noise and achieve a smoother result, since the calculation of the boundary line is performed on discrete data, the authors apply a Gaussian filter on the boundary nodes resulting in the smoothed curve $B_{k,m}$.

In order to compare the boundary lines of two fragments each curve $B_{k,m}$ is described by a signature based on the discrete $v_{k,m}$ curvature and torsion, a parameterization described in [208].

$$v_{k,m}(i) = [k_{k.m}(s)\tau_{k.m}(s)]^T$$

where $k(s)$ is the discrete curvature and $\tau(s)$ is the torsion with s being the arc length at curve node i. The boundary matching process again follows the string matching approach described in [208] and found in other methods as well, with some variations. The first variation is that while in [208] gaps are permitted; here authors do not opt for that as that might imply substantial differences between the respective boundary segments. The second difference is that while in the original approach the search is for matching non-overlapping closed-curves each one describing an entire potsherd, here the search is for overlapping surface regions that the boundary lines enclose. Therefore, the boundary lines here are compared mirrored in a search for the largest strings of similar consecutive elements $v_{k,m}(i)$ (*Figure 54*). The matching measure is the Euclidean distance of the local boundary features $v_{1,m}(i)$ and $v_{2,n}(j)$:

$$\Lambda(i,j) = \frac{1}{3}\sum_{q=-1}^{1}\left\|v_{1,m}(i+q) - v_{2,n}(j-q)\right\|.$$

The authors do not expect ideal matches between the segments and for that reason, a similarity tolerance is defined so that $\Lambda(i,j) < \mathrm{tol}_\Lambda$. Furthermore, matching facets are required to share edges of substantial support (at least $1/4$ the arc-length of the shortest boundary) for the matching to be valid. Once all similar signature segments are identified, a clean-up process discards substrings contained in larger ones. The remaining matching segments are sorted in descending length order and for each pair a rigid motion transformation is computed in order to align the segments. This is performed using a

[208] Üçoluk Göktürk, and I. HakkI Toroslu. "Automatic reconstruction of broken 3-D surface objects." *Computers & Graphics* 23.4 (1999): 573-582.

quaternion-based rigid body estimation method as suggested in [209]. Each transformation $\mathbf{M}_{m,n}^{i}$ that aligns the *m*-th facet of the first fragment with the *n*-th facet of the second fragment is associated with a matching error $e_b^{(i)}$, equal to the average distance between the aligned boundary segment points. This error is used to eliminate incompatible relative poses of the fragments with regard to their facet boundaries if:

$$e_b^{(i)} > a_b e_{min} + (1 - a_b) e_{max}$$

where a_b is a constant defined by the authors in the range of $(0 \ldots 1)$. e_{min} and e_{max} are the minimum and maximum boundary matching errors.

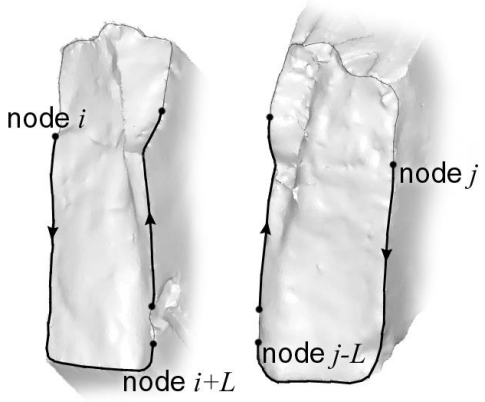


Figure 54: Matching of boundary segments.

Finally, in order to align the segments the authors use a closed-form solution that operates on consecutive triads of corresponding boundary points. In that way, a set of $N - 2$ transformations is derived and the one that minimizes the average distance between the two entire point sets is kept. This approach was preferred instead of an ICP-based alignment, since reliable boundary matches (if any) are usually localized. An ICP algorithm would be biased toward an overall alignment, on the other hand, which in the case of fractured lines, could be far from the desired pose.

The transformation $M_{m,n}^{i}$ calculated from the segment alignment is not sufficient for correct matching between facets and a full surface comparison has to be performed. Using $M_{m,n}^{i}$ the facets are initially aligned and oriented so that each one faces the other and their depth buffers are extracted (for more details see previous section). Using the depth values the authors initially test for penetration and if the maximum penetration surpasses a predefined tolerance value the pairing is discarded. Otherwise, full surface matching is performed using the unconstrained surface matching method and utilizing $M_{m,n}^{i}$ as a constraint. This way, from the initial vector of parameters $(\bar{\omega})$, ρ_1, x_1 and y_1 are eliminated and only the limited set of parameters have to be determined in order to minimizing the error metric.

10.4.2. Reassembling Fractured Objects by Geometric Matching

Huang et al. [98] proposed another solution to the problem of fragment reassembly. The method works on a set of point clouds and initially computes *integral invariants* for surfaces and 3D curves in multiple scales. These descriptors are utilized for both the segmentation of the fragments and the selection of surface features that will be used in the matching process. The fragments are segmented into a set of faces bounded by sharp curves of feature extrema and are classified into "original" faces and fractured surfaces based on the *"surface roughness"* descriptor. Initially, in the matching process,

[209] Horn Berthold KP. "Closed-form solution of absolute orientation using unit quaternions." *JOSA A* 4.4 (1987): 629-642.

only the fractured surfaces are considered and in a second step the robustness is increased by enforcing consistent alignment of the original faces. Pairwise matching is performed using patch surface features named *"feature clusters"*. All pairwise matches are simultaneously locally registered in order to find a consistent set of matching faces and mutual fragment positions such that they do not penetrate each other. Local registration uses the *Forward Search Method*, introduced by Atkinson et al. in [210], for a coarse alignment as it proved according to the authors both more efficient and robust to the presence of incorrect correspondences. The initial alignment is improved by utilizing an ICP variant using as constraint the avoidance of mutual penetration of the fragments.

The segmentation and pairwise matching in this work are based on comparing local curve and surface descriptors computed for points on the surfaces of the fragments. *Integral invariants* are defined by integrating spatial functions over moving domains centered at surface points. These descriptors are extensively explained by Pottmann et al. in [211] and are briefly discussed below.

- **Surface Integral Invariants:** Given a surface Φ, the boundary of a domain D in $\mathbb{R}^3$, χ_D is defined as a characteristic function that is 1 for points of D and 0 elsewhere. Let also $d^2(x,\Phi)$ be the square distance function between point x and a surface Φ and let also $B_r(p)$ denote a ball of radius r centered at point p with bounding sphere $S_r(p)$. For a point p of surface Φ the volume descriptor $V^r(p)$ and volume distance descriptor $VD^r(p)$ are defined as:

$$V^r(p) = \frac{3}{4\pi r^3}\int_{B_r(p)} \chi_D dx\,, \qquad VD^r(p) = \frac{15}{4\pi r^5}\int_{B_r(p)} d^2(x,\Phi)dx$$

  $V^r(p)$ is the ratio between the volume of the intersection $B_r(p) \cap D$ and the volume of the entire ball $B_r(p)$. $VD^r(p)$ is the weighted integral of the squared distance function of the entire ball $B_r(p)$. If surface Φ is a planar patch with respect to the extents of $B_r(p)$, $V^r(p) = 1$ and $VD^r(p) = 1/2$. As $r \to 0$ these relations are expressed as:

$$V^r(p) = \frac{1}{2} - \frac{3}{16}H \cdot r + O(r^2), \qquad VD^r(p) = 1 - (\kappa_1 - \kappa_2)^2 \cdot \frac{r^2}{28} + O(r^3)$$

  κ_1, κ_2 are the principal curvatures at point p and H is the mean curvature.

- **Spatial Curve Integral Invariants:** Given $c \subset \mathbb{R}^3$, a spatial curve, such as an edge of a fragment, the deviation descriptor $D^r(p)$ at a curve point p with respect to radius r is defined as:

$$D^r(p) = \frac{1}{r^2}\int_{x=0}^{r} \|c_x(p) - d_x(p)\|dx$$

  $c_x(p)$ and $d_x(p)$ are the first two points obtained by intersecting the sphere $S_r(p)$ with the curve c when going left and right from p. For a straight line c, $D^r(p) = 1$. As $r \to 0$ again from [211] the relation is expressed with respect to the curvature κ as:

$$D^r(p) = 1 - \frac{\kappa^2}{16} \cdot r^2 + O(r^3)$$

For the data segmentation a set of multi-scale surface features are calculated using the above integral invariants:

- **Surface Sharpness:** Based on the $V^{r_i}(p)$ the surface sharpness $s_{vol}(p)$ is defined as:

$$s_{vol}(p) = \left[\frac{1}{N}\sum\nolimits_{i=1}^{N}\left(V^{r_i}(p) - \frac{1}{2}\right)^2\right]^{\frac{1}{2}}$$

[210] Atkinson, Anthony, Marco Riani, and Andrea Cerioli. "Random start forward searches with envelopes for detecting clusters in multivariate data." In Data analysis, classification and the forward search, pp. 163-171. Springer Berlin Heidelberg, 2006.

[211] Pottmann Helmut, Johannes Wallner, Qi-Xing Huang and Yong-Liang Yang. "Integral invariants for robust geometry processing." Computer Aided Geometric Design 26, no. 1 (2009): 37-60.

r_i is set as a linearly increasing scale feature radius $r_i = r_{min} + i * (r_{max} - r_{min})/(N-1)$. Using $V^{r_i}(p)$, mean curvature information of the surface at multiple scales is implicitly taken into account. This metric is used to segment fragments into a set of faces, using the fact that points p on the break curves between faces have high value of $s_{vol}(p)$.

- **Surface Roughness:** To discriminate the original from the fractured faces the term of *surface roughness* is introduced. Let q_i be the *k*-nearest neighbors of a point $p \in \Phi$, and n_p and n_{q_i} be the surface normal vectors at these points. The local bending energy at p is defined as:

$$e_k(p) = \frac{1}{k}\sum_{1}^{k} \frac{\|n_p - n_{q_i}\|^2}{\|p - q_i\|^2}$$

The bending energy is averaged over the local neighborhood of a point p to estimate the roughness of the surface:

$$\bar{e}_{k,r}(p) = \frac{1}{|N_r(p)|}\sum_{q \in N_r(p)} e_k(q)$$

This metric is used for the classification of surfaces to fractured and original. In order to ensure that $\bar{e}_{k,r}(p)$ reflects the desired surface class, k (number of nearest neighbors) and r have to be chosen with care. The authors employ a supervised learning method [212] in order to calculate the optimal values. Finally, each point is classified into a fractured or an original surface using a binary function called surface roughness characteristic: $\rho(p) = 1$ for original and $\rho(p) = 0$ for fractured surfaces.

For the fractured surfaces and the boundaries between the fragments' faces the authors compute the so called *feature clusters.* Feature clusters are neighboring points with similar descriptor values that overlap with each other. This way, the authors keep track of coherent clusters and use this cluster structure in the matching algorithm to discard or verify feature correspondences.

In order to use the feature clusters for the matching process, a concise representation is created; for each such cluster a principal component analysis on all points in feature cluster C is performed and the feature vector of a cluster contains the following:

1. Barycenter $b(C)$ of the points in the cluster
2. Principal directions n_1, n_2, n_3
3. Points $p_k^{\pm}(C) = b(C) \pm l_d n_k(C)$, where $k = 1,2,3$ and $l_d = r_{max}/2$. These points are used for rough registration of feature clusters
4. A collection of representative points from C, $R(C) = B_{r_{max}}\big(b(C)\big) \cap \Phi$ that will be used for the fine registration of surface features
5. Cluster signatures *size* $sig_S(C) = (\lambda_1 + \lambda_2 + \lambda_3)^{\frac{1}{2}}$, and *anisotropy* $sig_A(C) = |\lambda_2/\lambda_3|$ where λ_k are the principal component eigenvalues with ascending order

Similarly, the representation of the edge features consists of an *anisotropy* signature $sig_A(C) = |\lambda_1/\lambda_3|^{1/2}$ and an *angle* signature $sig_a(C) = \angle(n_1 + \bar{n}_0).\bar{n}_0$ is the average surface normal of points in C that belong to the adjacent original surface.

The initial set of correspondences (*Figure 55*) is created by finding for each cluster C on fragmented surface S all clusters D on surface T. Two features C_{g_i} and D_{g_i} are potentially corresponding, if they were computed using the same descriptor g_i at the same scale. For edge features an additional correspondence between their angle signatures must exist: $\left|sig_a\big(C_{g_i}\big) + sig_a\big(D_{g_i}\big)\right| < \varepsilon_\theta$.

[212] Duda, Richard O., Peter E. Hart, and David G. Stork. *"Pattern classification."* John Wiley & Sons, 2012.

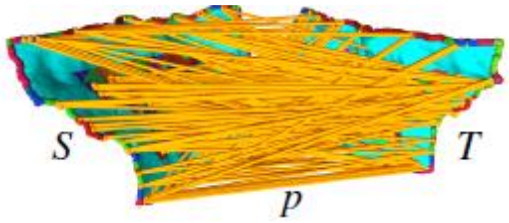


Figure 55: Potential feature correspondences. Figure taken from [98].

This initial set of correspondences is pruned using the associated shape and topology:

- **Shape Pruning**: A feature correspondence is considered further if and only if size and anisotropy deviations of the two are below specific limits

$$SD = \left|\frac{sig_S(C) - sig_S(D)}{sig_S(C) + sig_S(D)}\right| < \varepsilon_s \wedge \text{AD} = \left|\frac{sig_A(C) - sig_A(D)}{sig_A(C) + sig_A(D)}\right| < \varepsilon_A$$

- **Topological Pruning:** This is a two-step approach that tries to discard redundant and false correspondences while verifying the correct ones using the multi-scale nature of the descriptors and the respective cluster regions that are formed at a given location on a surface (*Figure 56*). In the first step, for each correspondence, the parent correspondence in terms of scale is evaluated for overlap. If no overlap exists the correspondences are discarded. Furthermore, correspondences whose features are not verified by their children are likely to be incorrect and are removed as well.

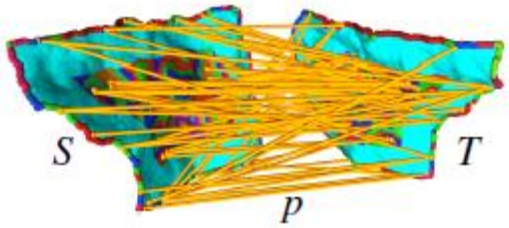


Figure 56: After shape and topological pruning. Figure taken from [98].

Since the Forward Search Method (FSM) is utilized in order to find possible matches between two fractured surfaces, the corresponding pairs of features found need to be tested for consistency as FSM needs a noise-free initial subset in order to operate successfully (*Figure 57*).

- **Geometric Consistency:** A set of pair features $p_1 = (C^1, D^1)$, $p_2 = (C^2, D^2)$ is considered geometrically consistent if the displacement vectors between the feature positions $b(C^1)$-$b(C^2)$ and $b(D^1)$-$b(D^2)$ are of similar length and the angles between their principal directions do not differ more than a certain threshold. For each pair of feature correspondences p_1, p_2 where C^1 is neighbor of C^2, D^1 neighbor of D^2 and p_1, p_2 are geometrically consistent, the one with the larger average signature is removed. The authors point that having both pairs of features is redundant and always keeping the larger one would reduce the set of potential correspondences too much for successful matching.
- **Registration Consistency:** A geometrically consistent set of pairs $p_1 = (C^1, D^1)$, $p_2 = (C^2, D^2)$ are furthermore tested via registration of the cluster points by minimizing the sum of squared distances between the corresponding points. The quaternion method of [209] is used to find the optimal Euclidean transformation that maps points $b(C^1), p_k^{\pm}(C^1)$ of feature C^1 and $b(C^2), p_k^{\pm}(C^2)$ of feature C^2 to the corresponding points of D^1 and D^2, respectively. Furthermore, local registration is used as in [213] to improve the alignment, based on a collection

of representative points $R(C)$ from the features. Only the corresponding pairs, where the registration process converges and lead to a final positioning are kept.

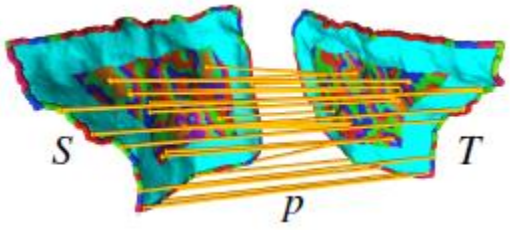


Figure 57: Consistent pairs used as initial sets for the FSM. Figure taken from [98].

Finally, the FSM is used to build a set of consistent feature correspondences from all the remaining pairs of features. Let $P = \{p_i\}$ be the set of features remaining. The correspondences are ordered by increasing score of:

$$\left|\frac{sig_S(C) - sig_S(D)}{sig_S(C) + sig_S(D)}\right| * \left|\frac{sig_A(C) - sig_A(D)}{sig_A(C) + sig_A(D)}\right|$$

The set of consistent pairs $M_{ij} = (p_i, p_j)$ is also ordered in the minimum $i + j$ order. The forward search iteratively builds the set of matching features. Let E^m be the subset of selected feature correspondences at iteration m, with the initial E^1 containing only the first element of M_{ij} that satisfies $\left|b(C^i) - b(C^j)\right| > 2l_d$. To form E^{m+1}, the corresponding pairs of E^m are used to compute a rigid transformation a^m that aligns the centers and the normals of its feature cluster pairs using again the method proposed in [213]. The residual of a feature correspondence $p_l = (C^l, D^l)$ is defined as $\left\|a^m\left(b(C^l)\right) - b(D^l)\right\|$ and the feature correspondences with the $m + 1$ smallest residuals are included in E^{m+1}. The process continues until the largest residual of E^m exceeds $2\varepsilon_{dev}$. The process is repeated with the next valid item of M_{ij} until all pairs in the set are marked.

Each set E^{m+1} is treated as a possible matching pose and the matching quality is evaluated incorporating three parameters. w^d, w^e and w^f. The two first denote the average deviation of the overlapping regions of the fractured faces and fractured edges respectively. The overlapping regions are computed using the bidirectional closest point search method introduced in [214]. w^f denotes the integral of the surface sharpness over the fragment surfaces' overlapping regions. The quality of match w tries to evaluate the minimum surface deviations for maximum surface size:

$$w = \log\left(\frac{w^f}{w^e w^d}\right)$$

Finally, when matching faces contain edges between original and fractured faces the authors require those edges to be within a threshold distance as a surface consistency measure.

[213] Pottmann, Helmut, Qi-Xing Huang, Yong-Liang Yang, and Shi-Min Hu. "Geometry and convergence analysis of algorithms for registration of 3D shapes." International Journal of Computer Vision 67, no. 3 (2006): 277-296.

[214] Pauly, Mark, Niloy J. Mitra, Joachim Giesen, Markus H. Gross, and Leonidas J. Guibas. "Example-Based 3D Scan Completion." In Symposium on Geometry Processing, pp. 23-32. 2005.

10.4.3. Pairwise Matching of 3D Fragments Using Cluster Trees

In [203] Winkelbach and Wahl propose a method for the reassembling of 3D fragment pairs, which initially segments two oriented point clouds to be matched using a decomposition of the point set into a binary tree structure using a hierarchical scheme on a combined 6D position-normal space. The matching process then scans through the space of possible contact poses of the fragments and tries to reassemble them using a tree search strategy that does not rely on surface features; pairwise matching is performed in depth-first order using as metric the maximization of the contact surface. It is critical to point out that the authors make the assumption of reasonably large fractured surfaces.

The proposed method expects *oriented point clouds* as input, i.e. point clouds with attached surface normal vectors. To obtain the normal, the authors rely on the method proposed by Mitra and Nguyen [215], either because normals are not provided or because the existing ones suffer from heavy noise.

The main principle of the method is to generate alignment hypotheses between clusters of points on the two point clouds. Since an exhaustive procedure operating on the flat point sets has an impractical computation time for typical data, the authors devised an elegant hierarchical algorithm that operates on a hierarchical decomposition of the point clouds.

At each level, pairs of clusters from one point cloud are compared for alignment with pairs from the second point set in terms of centroid position and local reference frame alignment. Since at the high levels of the hierarchy there are more aggregated oriented points than on the lower ones, the variance within the clusters drives the tolerance of the cluster pair congruence. The more clusters at a specific decomposition level successfully match, the more the matching area (of the clusters) is accumulated, reflecting a better score. However, although fast, high level matching tests are unreliable and the method tries to verify a matching hypothesis by proceeding down the binary tree to check recursively the hypotheses at finer levels. If the high level hypothesis breaks, then the sub-tree is pruned. Going down the tree, narrows the clusters and allows for more precise evaluation, up to the point level, where local point pair frames are aligned with the point normals and cluters become the actual point locations.

Initially, each point set (fragment) is decomposed into a binary tree structure (*Figure 58*). Their approach clusters coordinates and surface normals simultaneously in the combined 6D coordinate-normal space after the normals have been scaled to roughly match the position variance, in order to avoid bias of the splitting method. The splitting rule is based on the *k*-means clustering approach with k=2. The 6D splitting scheme is used because the coordinate and the normal orientation variance of each cluster are crucial factors for the pose tolerance hypothesis of the authors.

The authors propose a method for the generation of orientation-invariant pairwise relations between the oriented points of two fragments. This formulation will be used in order to evaluate relative poses of the two fragments that leave the selected pairs of points co-tangent. Points are considered co-tangent when they coincide and the corresponding normal vectors are facing each other. Given two oriented point sets A, B where:

$$A \coloneqq \{\mathrm{u} = [p_u, n_u] \mid p_u \in P_A \text{ and } n_u \in N_A\},$$
$$B \coloneqq \{\upsilon = [p_\upsilon, n_\upsilon] \mid p_\upsilon \in P_B \text{ and } n_\upsilon \in N_B\},$$

The subset C of points of fragment A that form a tangential contact with fragment B given a pose ${}^{A}\mathbf{T}_{B}$ is expressed as:

$$C \coloneqq \left\{a \in A \mid \exists\, b \in B\colon \left\|p_a - {}^{A}\mathbf{T}_{B} A p_b\right\| < e_p \wedge \left(n_a \cdot {}^{A}\mathbf{T}_{B} n_b\right) + 1 < e_n\right\},$$

Where the $\cdot$ operator is the dot product of two vectors and e_p, e_n are tolerance values required to handle numerical errors, noise and non-perfect matches. These values depend on the application and the authors adapt them to the surface accuracy manually.

[215] Mitra, Niloy J., An Nguyen, and Leonidas Guibas. "Estimating surface normals in noisy point cloud data." International Journal of Computational Geometry & Applications 14, no. 04n05 (2004): 261-276.

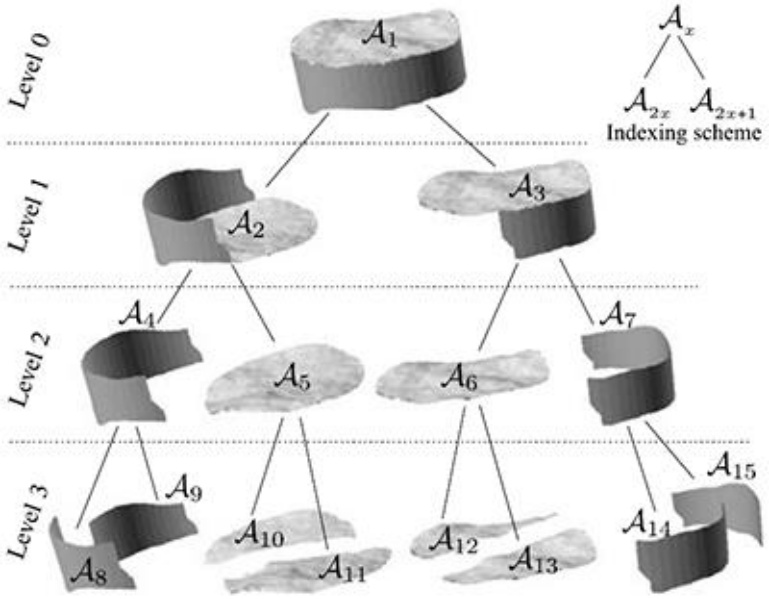


Figure 58: Cluster tree example. Figure taken from [203].

To obtain the relative pose ${}^{A}\boldsymbol{T}_{B}$ while avoiding the exhaustive search through the 6D space of all relative poses, the authors consider only pose hypotheses with a contact between the fragments. Pose hypotheses are constructed by assuming contact between some surface points on each fragment. Given four oriented surface points $a, c \in A$ and $b, d \in B$ a tangential contact between a, b and c, d is assumed thus constraining all degrees of freedom of the relative transformation. ${}^{A}\mathbf{T}_{B}$ is determined by means of two predefined frames:

$$ {}^{A}\mathbf{T}_{B}(a,b,c,d) = \mathbf{F}(a,c)^{-1}\mathbf{F}(b^{*},d^{*}) $$

where the superscripted star indicates the surface normal inversion and the function $\mathbf{F}(\mathrm{u},\mathrm{v})$ is a homogeneous 4x4 transformation matrix representing a coordinate system located between the oriented points u and υ:

$$ \mathbf{F}(\mathrm{u},\upsilon) \coloneqq \begin{bmatrix} \dfrac{p_{uv} \times n_{uv}}{\|p_{uv} \times n_{uv}\|} & p_{uv} & \dfrac{p_{uv} \times n_{uv} \times p_{uv}}{\|p_{uv} \times n_{uv} \times p_{uv}\|} & \dfrac{p_u + p_v}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

with the difference vector p_{uv} and the combined normal vector being:

$$ p_{uv} \coloneqq \frac{p_v - p_u}{\|p_v - p_u\|}, \quad n_{uv} \coloneqq n_u + n_v $$

The congruent relation between the oriented pairs is approximated using the tolerance values e_p, e_n.

$$ \mathrm{rel}(\mathrm{u},\upsilon) \approx \mathrm{rel}(\mathrm{q},\mathrm{r}) \iff \left|d_{uv} - d_{qr}\right| < \varepsilon_p \ \wedge \left|\cos\alpha_{uv} - \cos\alpha_{qr}\right| < \varepsilon_n $$
$$ \wedge \left|\cos\beta_{uv} - \cos\beta_{qr}\right| < \varepsilon_n \wedge \left|\cos\delta_{uv} - \cos\delta_{qr}\right| < \varepsilon_n $$

d expresses the distance between points, α is the angle between the normal n_u and vector p_{uv}, β is the angle between the normal n_v and vector p_{uv}, and δ is the signed angle between n_u and n_v around p_{uv} (dihedral angle between the plane with normal $n_u \times p_{uv}$ and the plane with normal $n_v \times p_{uv}$).

Since an n-point surface has n^2 different point pairs and n^4 point pair combinations the authors discard a naïve algorithm that would check all point-pair combinations and rather perform a top-down coarse-to-fine approach, where for the coarse levels a simple relative transformation is not used. The approach tries to solve the problem at a low resolution with small amount of data and later the resolution is increased to refine the solution. At the high levels of the cluster tree, a small set of pose hypotheses is calculated in order to prune the tree search early. The computed measure of quality of

the high-level hypotheses serves as a conservative upper bound for the low-level hypotheses. In order to prune the depth traversal, whenever the upper bound of a coarse high-level is worse than the last best match so far (lower tree level) the search is in depth-first fashion. For that reason, the high-level hypothesis must allow pose tolerances. This means that the congruence check cannot use a specific pose transformation for aligning the pairs first and then measuring the matching.

In order to estimate the amount of contact of a pose hypothesis without using an explicit relative transformation, the authors extend the congruence relation of point pairs described in the previous section to point triples:

$$(u, v, w) \cong (q, r, s) : \Longrightarrow$$

$$|n_{uv} p_{uv} p_{wu}| = |n_{qr} p_{rs} p_{sq}| \wedge (u, v) \cong (q, r) \wedge (v, w) \cong (r, s) \wedge (w, u) \cong (s, q)$$

The first sub-condition compares the relative pair orientation to avoid mirror symmetrical solutions, and all further sub conditions verify whether the two oriented point triangles are geometrically congruent.

For high-level comparisons, the notion of relations between point pairs is extended to cluster pairs, where the positions and orientations of the individual points within the clusters define the bounds of the tuple congruence and therefore the expected tolerance (see *Figure 59*).

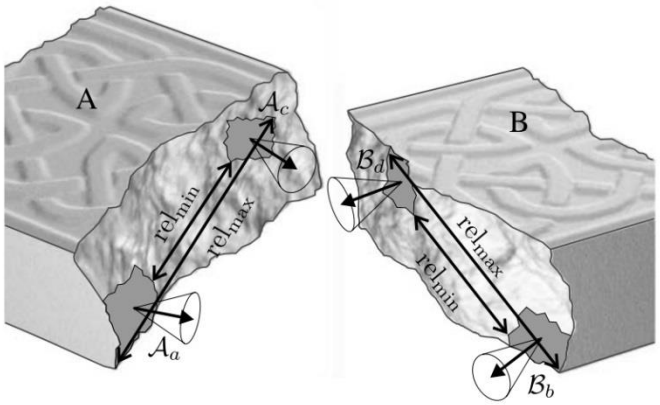


Figure 59: A simplified illustration of two high-level cluster pairs with overlapping relation intervals over position and orientation. Figure taken from [203].

10.5. Method Comparison

Table 15 provides a general overview and comparison of the methods presented in the categories of 2D, 2.5D and 3D that focus on fragment matching and are related to our research. Jigsaw methods are omitted from this comparison as the non-square piece matching methods are simplifications of the 2D fragment matching problem and the square piece matching methods focus only on textural color comparison of pictorial puzzles. Furthermore, we also omit from the comparison table several methods that focus on the reconstruction of symmetric pots as these methods do not address the generalized problem of fragment matching.

The criteria used in the table were chosen in order to highlight the differences of the methodologies.

Fragment Type category is self explanatory and provides a simple distinction on the used fragment type of each method. It is important to note that we focus on the representation and usage of the fragment and not its real form. For example, a method that treats fresco fragments as completely flat objects and exploits only the 2D top surface, is categorized as *"2D free form"* in our table.

Alignment informs about the used methodology in order to align a fragment pair regardless of how the pair was chosen. As expected, most methods use either a curve matching approach or an ICP based one.

Matching gives information on how each method finds the fragment pairs used in the Alignment process. *"Manual"* in this category means that the authors expected user interaction in the selection process of the fragments.

Descriptor is also self explanatory as a category and holds information about the descriptors used by each method in the matching process. *"Manual"* matching methods do not have an entry in this category.

Sampling category refers to the representation of the input data (surface, outline etc) used in the matching process.

Exhaustive tests is a score-based category with range from one to five (*), that is used to denote how thoroughly a method is been tested by its authors. Please note that the results in this category can be subjective as there is a lack of a benchmark set of models that would make the task objective.

Input type shows the input data type on which each method operates.

Preprocessing finally is a binary category that shows whether the matching process expects pre-calculated data or whether it operates directly on the input data.

Method		Fragment Type	Allignment	Matching	Descriptor	Sampling	Exhaustive Tests	Input Type	Preproc
Leitao et al.	[161]	2D free form	Elastic curve matching	Contour based	Curvature	Uniform	***	Image	Yes
Kong and Kimia	[162]	2D free form	Elastic curve matching	Contour based	Curvature, Torsion	Non-uniform	***	Image	Yes
Amigoni et al.	[163]	2D free form	Curve matching	Contour based. Evaluation of match using color	Curvature, Color	Uniform	**	Image	Yes
McBride et al.	[164]	2D free form	Elastic curve matching	Contour matching based on length distance and complexity of matching segment	Edge points	Non-uniform	****	Image	Yes
Papaodysseus et al.	[165]	2D free form	Curve matching	Contour based	Area between matching contour segments	Non-uniform	****	Image	No
Tsamoura and Pitas	[166]	2D free form	I——CP	Scaled histogram intersection of contour/color values	Color	Non-uniform	**	Image	Yes
Sagiroglu and Ercil	[168]	2D free form	FFT shift theory & Polar coordinates	Contour and synthesized extension texture matching using Euclidean distance	Color	Non-uniform	****	Image	Yes
Biswas et al.	[169]	2D free form	Iterative edge matching	Edge matching using the envelope created by the chain coded contour and the Minkowski sum	Chain code of contour edges	Non-uniform	***	Image	Yes
Justino et al.	[170]	2D free form	Elastic curve matching	Contour based	Angle of vertex, Euclidean distance of neighbour vertices	Non-uniform	****	Image	Yes
Zhu et al.	[171]	2D free form	Curve matching	Depict differences in histogram and search for clear-cut peaks	Turning function of arc length	Non-uniform	***	Image	Yes
Kampel and Sablatnig	[186]	Pottery	Curve matching	Classification of surface features and properties	Dimensions, Profile, Relief, Edge, Color, Type of Surface	Non-uniform	****	Triangulated Mesh	Yes
Willis and Cooper	[187]	Pottery	Levenberg-Marquardt (Similar to ICP)	Maximum likelihood estimation of parameters and Bayesian framework	Vessels axis, Axially symmetric surface, break curves	Non-uniform	***	Oriented Point Cloud	Yes
Oxholm and Nishiro	[191]	Pottery	Elastic curve matching	Contour/Color based, Evaluation of match using Normal vectors	Torsion, Curvature, Color, Surface normal	Uniform	****	Triangulated Mesh	Yes

Method		Fragment Type	Alligment	Matching	Descriptor	Sampling	Exhaustive Tests	Input Type	Preproc
Papaodysseus et al.	[193]	Fresco	Curve matching	Contour based	Length of contact, Contact surfaces, Volume of gap between fragments	Non-uniform	****	Triangulated Mesh	No
Belenguer and Vidal	[196]	Fresco	LCP	LCP over discretized surface representation	Contact area	Uniform	*	Triangulated Mesh	Yes
Brown et al.	[194]	Fresco	3D curve matching	3D contour based	Ribbon	Uniform	****	Triangulated Mesh	Yes
Toler-Franklin et al.	[195]	Fresco	3D curve matching	Use of machine learning to combine several descriptors-features	Color, Normal, Thickness, Ribbon	Uniform for Ribbon, Non-uniform for other features	*****	Triangulated Mesh	Yes
Scheuering et al.	[197]	3D free from	Voxel based method	Manual	-	Uniform	***	Volume Data	No
Willis et al.	[198]	3D free from	ICP	Manual	-	Non-uniform	***	Volume Data	No
Zhou et al.	[199]	3D free from	ICP	Manual	-	Non-uniform	****	Volume Data	No
Mellado et al.	[200]	3D free from	ICP	Manual	-	Non-uniform	*	Triangulated Mesh	No
Papaioannou et al.	[202]	3D free from	Enhanced simulated annealing	Search all possible relative poses using enhanced simulated annealing to minimize matching error	Curvature	Uniform	***	Triangulated Mesh	No
Huang et al.	[98]	3D free from	ICP	Forward search on feature clusters of multiple scales	Surface Integral Invariants, Spatial curve integral, Surface sharpness & roughness	Non-uniform	*****	Oriented Point Cloud	Yes
Winkelbach et al.	[203]	3D free from	Triplets of congruent points	Split fragments in a hierarchical scheme. Use depth-first order search to maximize contact surface, searching for triplets of congruent points based on their normal and the distance between them	Contact surface	Non-uniform	****	Oriented Point Cloud	Yes
Li et al.	[204]	3D free from	ICP	Match histogram values on patches around salient points	Curvature	Non-uniform	**	Oriented Point Cloud	Yes
Furnstahl et al.	[205]	3D free from	ICP	Search all pairs for maximization of contact surface	Contact surface	Non-uniform	***	Volume Data	No

Table 15. General overview and comparison of pair-wise matching methods.

10.5.1. Discussion

The work that will be conducted during the WP4 is closely related to that of 3D free form object matching and thus the corresponding pairwise matching methods are the ones that are more suitable to our problem case.

From the approaches focusing on 3D free-form object matching [197] [198] [199] [200] rely on manual matching and cannot offer a solution to our problem case.

The approach of Papaioannou et al. [202] performs robustly on the presented data and exploits the hardware in order to extract surface information and avoid more expensive computations. The matching metric presented could be easily implemented using CUDA on the GPU as the calculations are uncorrelated. The method also tries to exploit information of the boundary and perform initial alignment based on 2D curves, thus reducing greatly the search space of the 3D matching approach.

Huang et al.[98] show robust results with the data sets that are fine-fragmented and erosion free. That is because the approach relies on sharp curves and tight fitting-matching of the fragments. This is something that comes in contrast with the usual archaeological findings and is proved by the latter research of the authors [216] where human interaction was used in the process of matching blocks identification and on the selection of features for alignment. The calculation of the multi-scale descriptors could benefit from a GPU accelerated implementation, but the matching steps would be very difficult to implement efficiently.

The approach of Li et al. [217] is similar to [98], relying on sharp curves identified by multi-scale curvature estimation. Thus it is expect to perform similarly under the same test cases.

The work of Winkelbach et al. [203] uses the matching surface area as the metric and presents robust results for the used test cases but raises some doubts about the effectiveness of the hypothesis propagation on highly eroded archaeological data or objects with missing parts, where matches at high cluster tree levels may not be validated by the irregular and divergent surface landscape at finer levels of detail. The implementation of this method could benefit from a GPU implementation on the search of congruent clusters as these calculations are repetitive and uncorrelated.

The work of Fürnstahl et al. [205] also uses the surface area as the matching metric, but requires a template, almost similar to the final reconstructed object and a minimal user interaction in order to obtain an initial alignment of the fragments. Due to these restrictions their method cannot be adopted for our problem.

Since we don’t know yet the exact approach that we will be using, it is possible that other methods could provide possible solutions. Contour matching is a generic matching approach that can be rather easily applied also in three dimensions and the descriptors of curvature and torsion present robust results along with elastic curve matching algorithms. These could be easily combined with some additional information about the remaining volume gap and the length of matching. Finally should we require comparison of color/texture information several metrics discussed in the area of the square pieced jigsaw matching approaches could be evaluated.

[216] Thuswaldner, Barbara, Simon Flöry, Robert Kalasek, Michael Hofer, Qi-Xing Huang, and Hilke Thür. "Digital anastylosis of the octagon in ephesos." Journal on Computing and Cultural Heritage (JOCCH) 2, no. 1 (2009): 1.

[217] Li, Qunhui, Mingquan Zhou, and GuohuaGeng."Pairwise matching of 3D fragments."*Information Management, Innovation Management and Industrial Engineering (ICIII), 2012 International Conference on*.Vol. 3.IEEE, 2012.

11. MULTI-PIECE MATCHING (ASSEMBLY) METHODS

Now that we reviewed the process of pairwise matching through all the categories dictated by the type of fragments, we will focus on the multi-piece reassembly process of the original object. The problem is typically represented as a graph with fragments as nodes and possible pairings as edges. Most authors avoid proposing a solution based on global criteria evaluation, as this has been proven to be an NP-complete problem [218], and an optimal algorithm that would solve the problem in a reasonable execution time does not exist. Most of the methods discussed in the next paragraph perform the assembly in a hierarchical way by considering only local criteria and merging fragments with a best-first approach based only on per-pair compatibility. These heuristic approaches of course, result in quasi-optimal solutions to the problem. Other approaches try to incorporate a more global choice of pairing selection, by considering multiple matches at the same time, or introduce global-relaxation steps to optimize the fragment alignment in each step.

Wolfson et al. [148] propose a two-step approach for the global assembly problem. In their problem case (jigsaw puzzle), they consider the final shape of the object as known and so they first assemble the frame of the puzzle using a heuristic solution to the Traveling salesman problem (TSP). Then, they fill the interior puzzle by processing corners sequentially, beginning with the lower left side of the puzzle and advancing within each row to the right. An improved version of this approach is used by Goldberg et al. [152], where the interior pieces are filled using a greedy algorithm that considers all neighbors for each subsequent filling (global criterion) and the partial solutions are re-optimized by distributing the matching error evenly among all the border pieces (relaxation-step). A greedy best-first strategy is utilized by Papaodysseus et al. [165], Kano et al. [219] and Willis et al. [187] where using local-only criteria the authors select the most probable matching pair in order to solve the global reconstruction problem. Ucoluk et al. [208] apply a best-first strategy using also local evidence but also utilize backtracking, in order to account for possible invalid matches. Kong and Kimia [162] also use best-first strategy with back-tracking, using a global criterion that considers triplets of fragments. McBride and Kimia [164] introduce another global matching criterion by rewarding triple junctions (*Y* and *T*-junctions) that they found to be dominant in archaeological puzzles. Their approach is also best-first but in order to avoid backtracking they employ a beam-searching. Alajlan [220] also use a beam-search approach and for the general matching strategy utilize the Hungarian process which is an optimization algorithm that simultaneously selects all neighbors for each piece added in the puzzle using local criteria. Huang et al [98] apply a greedy search algorithm using forward search in order to merge the fragments until all are merged or no other valid candidates exist. Zhu et al [171] perform a best-first approach that utilizes a matching relaxation technique, where the global match confidence of all candidate matches is exhaustively searched. Castañeda et al. [221] also propose another global relaxation scheme, where a physically based non-linear solver is used to minimize all local alignment errors. Huber [222] in order to solve a multi-view registration problem utilizes Kruskal's [223] algorithm that compute the minimum spanning tree in the formed graph. The same approach is used by

[218] Demaine Erik D., and Martin L. Demaine. "Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity." *Graphs and Combinatorics* 23.1 (2007): 195-208.

[219] Kano, M., Shogo Yasuhara, Shohei Kato, and H. Itohi. "Similarity-based approach to earthenware reconstruction." In *Image Analysis and Processing, 2001. Proceedings. 11th International Conference on*, pp. 478-483. IEEE, 2001.

[220] Alajlan, Naif. "Solving square jigsaw puzzles using dynamic programming and the hungarian procedure." *American Journal of Applied Sciences* 6.11 (2009): 1941.

[221] Castañeda, Antonio García, Benedict Brown, Szymon Rusinkiewicz, Thomas Funkhouser, and Tim Weyrich. "Global consistency in the automatic assembly of fragmented artefacts." *Eurographics Association* (2011): 73-80.

[222] Huber, Daniel F. *Automatic three-dimensional modeling from reality*. Diss. Carnegie Mellon University, 2002.

[223] Kruskal, Joseph B. "On the shortest spanning subtree of a graph and the traveling salesman problem." *Proceedings of the American Mathematical society* 7.1 (1956): 48-50.

Gallagher [156] for solving jigsaw puzzles, which achieves state-of-the-art performance in the specific area. Finally, Toyama et al. [224] and Papaioannou et al. [201] utilize genetic-like algorithms, as they have the ability to find relatively quickly good solutions to optimization problems that operate in large spaces.

11.1. A Global Approach to Automatic Solution of Jigsaw Puzzles

Goldberg et al. [9] propose a global matching method based on the work of Wolfson et al. [148] that can solve big apictorial jigsaw puzzles that obey three standard rules. For a less restricted form of puzzles where the sides are not well defined a fourth rule is required:

1. The puzzle has a rectangular outside border
2. Pieces form a rectangular grid. In that way, each interior piece has four neighbors, one at each main direction (right, left, above, below)
3. Pieces interlock with their neighbors by tabs. A tab consists of an "indent" of a piece, matching an "outdent" on the neighboring piece
4. Each piece has no neighbors except its primary ones. In more detail, the cutting lines between pieces meet only at right angles (+-junctions) rather than a mix of +-, T- and Y-junctions

The algorithm first solves the border of the puzzle using a heuristic Traveling Salesman Problem (TSP) and then fills the interior pieces. For the filling process, the authors maintain at each step an optimized planar embedding of the partial solution and each interior piece is fitted in that, using a greedy placement algorithm that allows any number of neighbors around the pocket.

The sides of a piece are classified in three categories, sides with tabs, which are further classified as having an indent or outdent, and straight sides. The proposed solution first locates the indents, marks the boundaries near them and afterwards searches for outdents on the unmarked sides. Finally, in-order to identify straight sides, the method searches for stretches of a piece's boundary which are characterized as straight segments with an "inward" turn at each end and from which neither has been previously identified as an indent.

The border pieces are distinguished between the ones with one straight side and the corner ones, which have two straight sides. Those with one straight side are all oriented with the straight side down and the corner ones with the sides down and right. A score function $s(A,B)$ is defined, measuring the matching compatibility of the right side of A to the left side of B. The best ordering is solved using an asymmetric TSP with $s(A,B)$ depicting the distance from A to B and $s(A,B) \neq s(B,A)$.

All scores are calculated in polynomial time and in order to convert this to a path signifying a connected chain of pieces at the perimeter of the puzzle, a random piece P is picked. From P the next best match is selected and connected and the iterative process continues until it returns to P. This process may not return a complete traversal of the puzzle's boundary as the path may return to P in less than n steps, with n being the total number of boundary pieces. In their experimentation the authors obtained either a single n-long cycle or two cycles. The first case is a correct result while in the second they test all the possible ways of stitching the two cycles together in $O(n^2)$ steps. The ordering is easily checked as the corner pieces must be symmetrically placed, based on the first rule (rectangular outside border).

For the scoring function the authors take parallel lines to the border and for each line they compute the length between the two pieces. In the case of perfect matching, these lines have equal length (see *Figure 60*).

[224] Toyama, Fubito, Yukihiro Fujiki, Kenji Shoji, and Juichi Miyamichi. "Assembly of puzzles using a genetic algorithm." In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 4, pp. 389-392. IEEE, 2002.

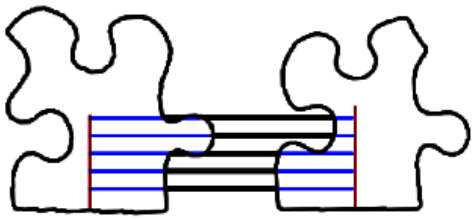


Figure 60: Perfectly matching pieces. Lengths of bold horizontal lines should be equal. Figure taken from [152].

In order to align the adjacent pieces, the authors introduce special points, called *fiducial points* (robust canonical locations). Details on the alignment will not be presented here as that is beyond the scope of our section theme. After the alignment, a global relaxation is also used in order to distribute the error of alignment evenly among all border pieces.

A greedy algorithm is applied for the interior pieces. There are three main parts in this step. The score measuring how well a piece fits into an eligible pocket, the strategy in which pieces are placed and an optimization step that readjusts the embedding after each new piece is placed. An eligible pocket is defined as an unfilled piece slot that is adjacent to at least two existing pieces.

The score of fitting an interior piece *P* consists of two steps. First, the position of *P* that minimizes the sum of squares of distances between *P*'s fiducial points and the pocket's tabs is calculated. Next, the score for *P* is computed. The boundary of *P* is walked and for each one of its vertices the closest boundary point on any of the neighbor pieces defining the pocket is calculated. The score is defined as the average distance between a vertex of *P* and its closest pocket point. The walking across the boundary of *P* is performed across the tangent points of the neck of the tab with the radius emanating from the ellipse center. It is crucial to point out that while the above metric was sufficient for a 100-piece puzzle, the authors had to modify the score computation for a larger puzzle set (204-pieces) with a one-step look-ahead approach. After fitting piece *P*, for each one of the newly created eligible pockets, the piece with the best fitting is found, and the score of *P* is calculated using the temporary neighbors. It is obvious that the look-ahead step greatly increases the computational cost of the method although it provides more robust results.

11.2. Reassembling Fractured Objects by Geometric Matching

For the multi-piece object reassembly, Huang et al. [98] propose a greedy approach that initially merges compatible pairs of fragments until all parts are merged or no valid matches remain. A graph is built with the fragments as nodes and the pairwise matches between the fragments as edges, whose weight is the quality of match described in the pairwise matching process. The goal is to compute the set of edges that result in the best reassembly of the object.

The greedy algorithm uses the observations that incorrect matches lead to penetration and that the matching algorithm has to be iterative as there is no confidence in the pairwise matches for a single-step solution.

In more detail, based on the results of pairwise matching, a graph $G(F,E)$ is defined, where nodes $F_i \epsilon F$ represent fragments and edges $e = (F_i, F_j)$ are the possible matches between two fragments. Each edge is associated with a corresponding weight and the relative transformation calculated during the pairwise matching. A second weighted graph is defined $M(\Gamma, L)$ where each node $G^k \epsilon \Gamma$ is a weighted sub-graph $G^k(F^k, E^k)$ of G for a group of fragments. In order for two sub-graphs G^1, G^2 to be connected in M, two fragments $F_i \epsilon F^1$, $F_j \epsilon F^2$ with $(F_i, F_j) \epsilon E$ should exist. The weight of the connecting edge is defined as the sum of the weights of all edges:

$$e = (F_i, F_j) \epsilon E, F_i \epsilon G^1, F_j \epsilon G^2.$$

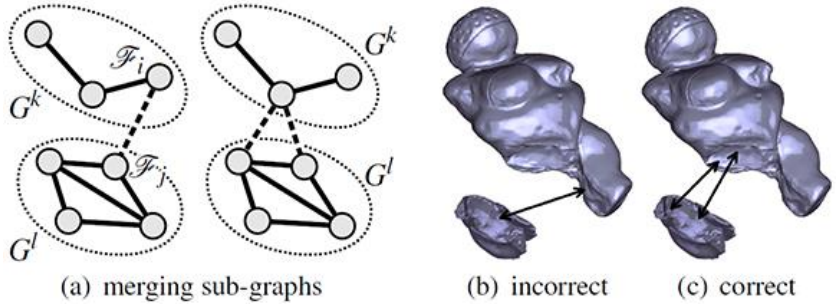


Figure 61: Merging of sub-graphs. The authors prefer to merge based on the maximum total weight over the highest weighted edge. Figure taken from [98].

The initialization of $M(\Gamma, L)$ is performed by considering each fragment as a single sub-graph. The merging takes place in an iterative manner as long as there exist more than one fragments and at each step, a group of edges of M is selected using the forward search algorithm and the corresponding sub-graphs are merged, provided the merged nodes do not inter-penetrate. If penetration is detected the set of edges is discarded and the next set is examined.

The approach tries to merge sub-graphs (fragment islands) in order to gradually build up a solution until only one node exists. The merging process is performed similarly to the grouping of cluster correspondences using forward searches. Instead of iterating through the edges connecting the two sub-graphs and relying on the stronger one to bring the two clusters together, the authors opted for a global consistency check (*Figure 61*); they observe that a valid merge procedure should result in a near-identity relative transformation of the fragments over an edge loop and use the maximum total weight of the edges interconnecting the sub graphs of such configurations as a scoring function. In each merging step wrong matches are detected and removed using the collision detection algorithm presented in [225].

During the fragment merging the points of all matching fractured faces are removed using the bidirectional closest point search method of [214] and in a second step, any holes are filled using a modification of the method presented in [226] so that the new fragment is a single closed surface.

11.3. Archaeological Fragment Reconstruction Using Curve-Matching

McBride and Kimia in [164] use a best-first strategy for the multi-piece assembly. The process is guided by a global metric that utilizes two conjectures concerning the structure of puzzles composed of natural materials. These conjectures are used as a measure of confidence for the arrangement of the puzzle as it is constructed.

- **Conjecture 1:** The majority of junctions in archaeological fragment re-assembly problems are triple junctions. These are the "*T"* and "*Y"* junctions. Based on their experiments, 70% to 90% of all interior junctions were "T" junctions and 6% to 9% of the remaining junctions weree "Y" junctions. Higher-order junctions occur very infrequently.
- **Conjecture 2:** The majority of matching segments in archaeological fragment re-assembly begin at a pair of corners on the fragments. Based on this observation, they consider the matching contours that begin at a pair of matching corners, which reduces $O(n^2)$ complexity to $O(m^2)$ where n is the number of sample points on a fragments segment while m is the average number of corners per fragment (4-5 in the authors' data).

[225] Lin, Ming C., and Dinesh Manocha. "Collision and proximity queries." (2003).

[226] Amenta, Nina, and Yong Joo Kil. "Defining point-set surfaces." *ACM Transactions on Graphics (TOG)* 23.3 (2004): 264-270.

Search for the global arrangement is initialized by taking the highest ranking match of the ordered list of pair-wise matches and registering the corresponding pair of fragments. At each step the best match which connects a new fragment is used. During this best-fit strategy, the authors use the logical constraint of ruling out matches with significant overlap. The global confidence metric is introduced at this stage based on whether the fragments form adjacencies and triple junctions ("T", "Y"). The local confidence metric used in the first stage of pair-to-pair matching is completely replaced afterwards with the global one. Every time a fragment is added all matches are re-evaluated using the global confidence and the matching that results in the best possible contribution to the global solution is preferred.

In order to avoid back-tracking due to potentially incorrect matches, the authors use a beam-search technique: k different solutions are considered in parallel by starting the initial process with different fragments. To avoid infinite execution of the algorithm, when extra non-matching pieces exist in the set, the authors set a threshold in the confidence required for a matching operation.

11.4. Globally Consistent Reconstruction of Ripped-Up Documents

Zhu et al. [171] use a relaxation scheme for the global reassembly, in which the definition of compatibility between neighbor matches is used and global consistency is defined as the object reassembly criterion.

In more detail, global matching confidences $x = \{x_1, \dots, x_N\}$ are assigned to each node and the adjacency information implied by them is applied to define the respective neighborhoods $G = \{G_1, \dots, G_N\}$. A global consistency $\mathrm{C}(x)$ evaluates the propagation of local interactions between the neighboring matches. The problem is then reduced to that of maximizing the global consistency, which is a nonlinear optimization problem with boundary constrains. A gradient projection method is employed to maximize the global consistency. After the maximization procedure converges, confident matches are merged and the same procedure runs again for the rest of the fragments.

The neighborhood for any candidate match M_i is defined as:

$$G_i = \{M_j \in M | M_j \cap M_i \neq \emptyset, i \neq j\}$$

where $M_j \cap M_i = \emptyset$ indicates that there is no common fragment between the two matches. Two measures of compatibility are defined: the area compatibility and the segment-based compatibility. Both are used to designate inconsistencies of matching results. Let F_r be the common fragment between two neighbor matches and F_s and F_k be the other two fragments (one from each match). The area compatibility measure is defined as:

$$\gamma_{ij} = \begin{cases} 1 - \left(\frac{A}{A_i}\right)^2 & \text{if } \frac{A}{A_i} \leq \alpha_1 \\ -\frac{A}{A_i} & \text{otherwise} \end{cases}$$

α_1 is a constant that control the maximum tolerance of overlapping with suggested value 0.2, A_i the total area of fragments F_r, F_s and A the overlapped area of F_s and F_k.

The segment-based compatibility measure is defined as:

$$\gamma_{ij} = \begin{cases} 1 - \left(\frac{d_2}{d_1}\right)^2 & \text{if } \frac{d_2}{d_1} \leq \alpha_2 \\ -\frac{d_2}{d_1} & \text{otherwise} \end{cases}$$

α_2 is also a constant in the similar sense to α_1 with the same suggested value and d_2 and d_1 denoting the matching segment of F_r on segment M_i and M_j respectively.

In practice, the authors found the area compatibility measure to be a better choice when the number of fragments becomes greater than 50.

The support that M_i obtains from all its neighbors G_i is defined as:

$$\theta_i = \sum_{j \in G_i} \left(\gamma_{ij} x_j - \alpha_1 e^{\alpha_2 (x_j - \gamma_{ij} - \beta)} \right)$$

where β is a constant defined by the authors. The suggested values for the constants are $a_1 = 3$, $a_2 = 5$, $\beta = 1.8$. The global consistency measure is defined as:

$$C(x) = \sum_{i=1}^{N} x_i \theta_\iota$$

where x_i is the global matching confidence and is estimated via maximization of the global consistency evaluator $C(x)$. The optimization problem to be solved is expressed as $\max_{x: x_i \in [0,1]} \{C(x)\}$. Although this maximization problem is stated as an optimization problem over a set of parameters, the authors transform it to an optimization problem over the controller of a dynamic system by defining an adaptive gradient descent searching rule over the space of variables $\Omega = \{x: 0 \leq x_i \leq 1\}$:

$$x_i^{(0)} = \alpha_3 - \frac{S_i - S_{min}}{S_{max} - S_{min}} \times w, i = 1, \dots, N$$

$$x_i^{(n+1)} = \min\left\{ max\left\{ x_i^{(n)} + \rho_n^{opt} \frac{\partial C(x)/\partial x}{\|\partial C(x)/\partial x\|}, 0 \right\}, 1 \right\}$$

where ρ_n^{opt} is a parameter in the interval between 0 and 1 that is computed by exhaustive search with goal $C(x_n) \leq C(x_{n+1})$.

The relaxation approach can be described as a four-step process:

- Step 1: Evaluation of the global match confidence $x_i^{(0)}$ for all candidate matches
- Step 2: Determine the compatibility γ_{ij} between neighboring candidate matches
- Step 3: Compute the modified $\frac{\upsilon}{\|\upsilon\|}$ and search for the optimal value ρ_n^{opt}, then calculate the new $x_i^{(n)}$
- Step 4: Stop if $\rho_n^{opt} = 0$. Repeat Step 3, otherwise

Candidate matches with confidences of 1 are merged and the overall process continues until all fragments have been merged or no merging has taken place.

11.5. Jigsaw Puzzles with Pieces of Unknown Orientation

Gallagher [156] proposes a tree-based global assembly solution inspired by Kruskal's algorithm [223] for finding the minimum spanning tree of a graph $G(V_G, E_G)$. The graph is constructed by using each jigsaw piece as a vertex and the compatibilities as edge weights. Each graph edge has also an associated geometric configuration r between the pair of jigsaw pieces. The only constrain imposed on the search for the MST of this graph is that there should be no overlap (based on the geometric configurations) as the assembled puzzle should be flat. Using this constraint, no vertex can have greater than four degree and thus the problem is NP-hard.

In order to find the MST the authors propose a heuristic method based on Kruskal's algorithm that has three stages: A constrained version of Kruskal's algorithm is performed in order to find a tree in that results in a flat assembly. Each piece is initially a forest by itself using no rotation. A forest records the relative spatial location of the member vertices and the absolute rotations to apply to each piece. Edges are examined and the one with the lowest cost e_{min} is found and removed from the set of remaining edges. In case vertices of e_{min} belong to the same forest, e_{min} is discarded as that would form a loop. Otherwise the forests joined by e_{min} are merged according to the geometric relationship r defined by it. If the merge results in overlap of pieces, both the merge and e_{min} are discarded.

Since the authors have not imposed constrains on the size of the puzzle, the resulting tree from the previous stage, might not result to a rectangular frame. If the dimensions of the puzzle are known, the tree is trimmed by finding the position of the frame that trims the fewest pieces. In case the orientation of the puzzle is not given, the trimming must be performed for both orientations.

11.6. Automatic Reconstruction of Archaeological Finds – A Graphics Approach

Papaioannou et al. [201] in order to achieve multi-part assembly present an optimization scheme based on the pair-wise matching errors and geometrical transformations that are used to arrange the fragment collections in a set of reconstructed objects. The algorithm is a genetic-like algorithm that generates and rearranges the fragment pairs interactively. Four rules are used in order to minimize the global reconstruction error, which is the sum of matching errors of individual combinations. a) Fragments can be attached to as many objects as the number of their fragmented facets, b) the bond between two fragments is unique, c) some fragments may not belong to a valid assembly and d) fragment pairs that yield smaller matching error are favored

Every valid combination of facets is assigned a fitness value in the range [0.0 ... 1.0] that reflects the suitability of the pair. A set of such combinations conforming to the first and second rule, form a string that represents the active bonds among the fragments. This string is iteratively mutated and the fitness values are adjusted until convergence is achieved.

The fitness values are initiated to 0.5 and in the combination string every fragment appears at least once. The combinations with small pairwise matching error are inserted into the string and their fitness values are increased. The fitness of a pair is adjusted according to the global error that the current instance of the combination string produces. If the addition of a combination increases the matching error, the string is discarded (see *Figure 62).*

Permutation of the combinations is performed by crossing-over the participants of some pairs of combinations by selecting less fit fragment pairs. The algorithms terminates when fitness values are stabilized in a small range.

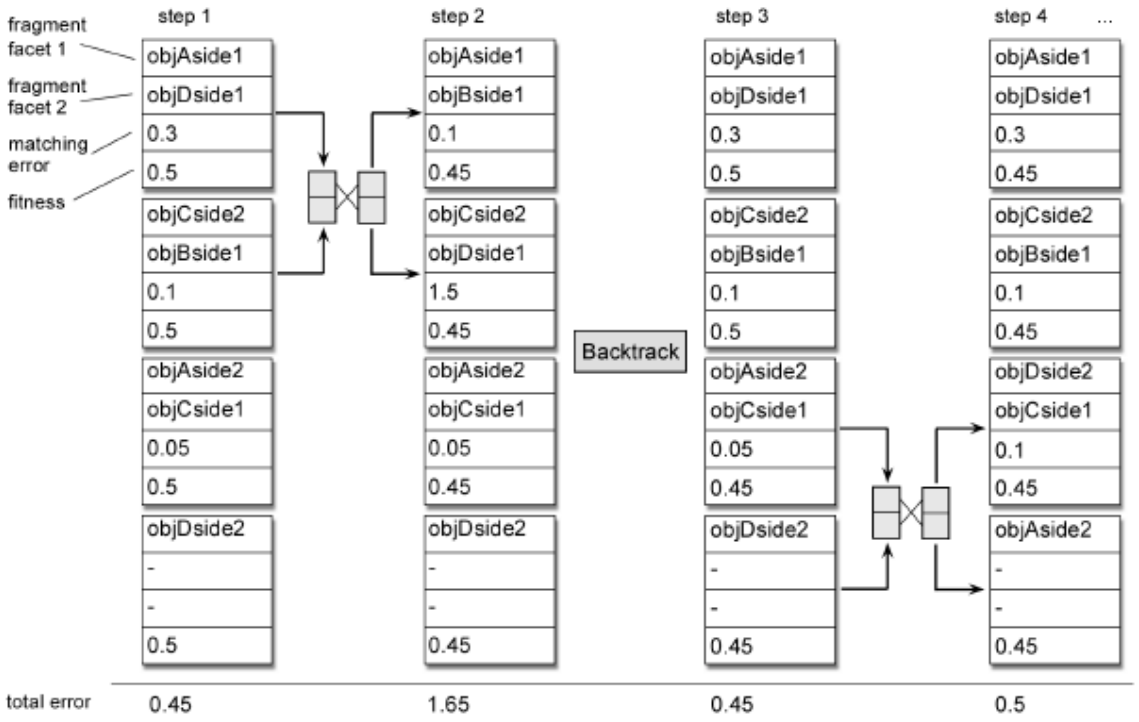


Figure 62: Genetic-style object reassembly algorithm

11.7. Method Comparison

Table 16 offers a general overview and a brief comparison of the multi-piece matching methodologies. We could base this comparison on several specific characteristics of each method. One example could be whether the methods are exploiting parallelism, but this could be argued for all of them, given enough effort on their implementations. To avoid this problem, we are mostly focusing on the general ideas of each method and not the specifics of each implementation. The general criteria used to categorize the methods in Table 16 are the following:

- **Strategy** shows the algorithmic approach followed in each method in order to solve the problem.
- **Criteria** describe whether Local or Global criteria are used in the puzzle solving process.
- **Mixed Puzzles** describes whether the proposed approach manages to handle cases of mixed sets of fragments. For example Golberg et al. [152], when initially constructing the boundary of the puzzle, expect to find strictly one solution and so their approach cannot handle mixed puzzles.

Method		Strategy	Criteria	Mixed Puzzles
Goldberg et al.	[152]	Heuristic TSP for Border, Greedy Best First for Interior Pieces	Global	No
Papaodysseus et al.	[165]	Greedy Best First	Local	Yes
Kano et al.	[219]	Greedy Best First	Local	Yes
Willis et al.	[187]	Greedy Best First	Local	Yes
Huang et al.	[98]	Greedy Best-First	Local	Yes
Ucoluk et al.	[208]	Best First with Backtracking	Local	No
Kong and Kimia	[162]	Best First with Backtracking	Global	No
McBride and Kimia	[164]	Best First with Beam-Search	Global	Yes
Alajlan	[220]	Hungarian Method with Beam-Search	Local	No
Zhu et al.	[171]	Best First with Global Matching Relaxation	Global	No
Castañeda et al.	[221]	Best First with Global Matching Relaxation	Global	Yes
Gallagher	[156]	Greedy (Kruskal)	Global	Yes
Toyama et al.	[224]	Genetic Algorithm	Global	No
Papaioannou et al.	[201]	Genetic Algorithm	Global	Yes

Table 16. General overview and comparison of multi-piece matching methods.

11.7.1. Discussion

Due to the nature of the problem (*NP-Hard*) the methodologies for the object reassembly have to make a compromise between the execution time and exhaustiveness of the search space. Initial approaches used greedy "best first" schemes with either global or local criteria, but would easily result on local minima. In order to address this issue, later approaches utilized backtracking, beam-searching and relaxation schemes that expand the search space for the global solution. Genetic algorithms offer another approach to the global matching problem but they cannot guarantee convergence and the stop criterion cannot be easily set.

While it is still early to discuss the solution we will be adopting, a heuristic based approach that would exploit outer surface criteria could prove efficient. In addition to that, as described in the DoW, our approach will be able to utilize automatically predicted shapes resembling the complete object. While one can note a similarity with the jig-saw puzzle solving problem, approaches such as [152] that try to solve initially the exterior of the puzzle cannot be adopted, as in our case the predicted shape will be a rough estimate prone to be updated and thus it cannot be treated as a shell of the final re-assembled object, but rather as a base guideline imposing extra rules on the process.

12. CURRENT ADVANCES IN GENERAL PURPOSE GPU ARCHITECTURES AND TECHNIQUES

The processing throughput (floating-point operations per second) of *Graphics Processing Units* (GPUs) has grown at a faster rate than that of CPUs, as shown in Figure 63. It is worth noting that many supercomputers in the top 500 list [227] use GPUs and similar stream processors in order to achieve high performance levels. Therefore, a natural choice for many computationally intensive problems is to design algorithms that get executed on the GPU instead of the CPU. In this section we provide a quick overview of the main GPU characteristics and we describe their influence in our design decisions and research directions.

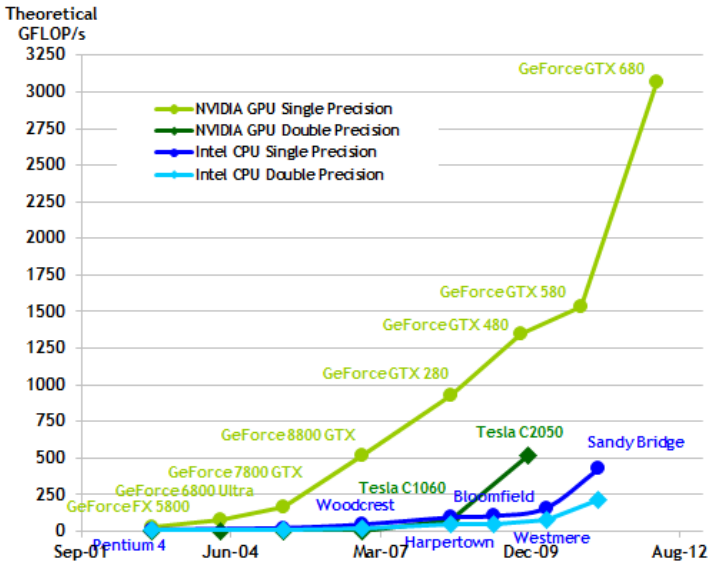


Figure 63: Floating-Point Operations per Second for the CPU and the Graphics Processing Unit. (Source: NVIDIA Corporation)

12.1. A Brief GPU History

Rendering is an embarrassingly parallel problem. The same set of operations must be completed on a large set of vertices, during the vertex processing stages of the rendering pipeline, while another set of operations must be completed during the fragment processing on a large number of pixel samples, in order to compute the final color of the fragments that were generated during the rasterization. This set of operations is commonly referred to as a *shader*.

Since the number of vertices and pixels are very high on typical scenes, this workload is extremely high for general purpose CPUs to handle at interactive rates. Therefore, hardware manufacturers developed specialized graphics hardware that is designed to accelerate these operations. Early designs by Silicon Graphics only supported a fixed shader to be executed on the vertices and another one for the pixels, both parameterized by changing some variables on the graphics API. This was necessary, because the rather limited set of operations that was supported by these "fixed" shaders was implemented in fixed-function hardware. It is worth noting that at the time, programmable shaders were only popular in Pixar's RenderMan (Reyes) software rendering pipeline, while most hardware designs featured fixed-function shaders. Furthermore, it is worth noting that the first consumer 3D hardware, such as the 3Dfx Voodoo chips and other similar designs of the mid 90s, completely skipped the fixed-function vertex shading part, which had to be performed on the CPU, while the hardware only implemented fast triangle rasterization and shading. This design choice made sense at the time, since the number of pixels was much higher that the number of vertices, as the typical 3D scenes only consisted of a few thousands of polygons.

[227] http://www.top500.org/.

The first consumer graphics card to perform the complete vertex and pixel shading pipeline in hardware was the NVIDIA GeForce. The term GPU (Graphics Processing Unit) was popularized by NVIDIA during the introduction of this card and is used until today in order to describe specialized graphics hardware.

12.2. Programmable Shaders

The next big step in GPUs was the introduction of programmable shading with the advent of OpenGL 2.0 and Direct3D 9.0 APIs. Instead of using a fixed (but customizable) shader for the processing of the vertices and pixels, now the software developer could write an arbitrary set of operations that influence a vertex or a pixel, using a high level shading language.

Today's GPU architectures consist of a large pool of compute units, in order to execute the programmable shaders, and a very small set of fixed-function units exist for graphics-specific operations, like triangle setup and rasterization. The block diagram of a contemporary GPU is shown in *Figure* 64.

The main restriction that shaders have is that they are not allowed to have side effects, meaning that they have a specific set of outputs, without random write access to a global shared memory. This restriction was necessary in order to keep the design of the underlying hardware implementation simple and at the same time allow fast and efficient parallel execution of the shaders. Interestingly, the same restriction on the side effects of the shaders also exists in the RenderMan Shading Language (RSL), so it is clear that the same principles apply also for efficient software implementations. For completeness, we should mention that this restriction on side effects was lifted in OpenGL 4.2, but the software developers should manually manage conflicts and race conditions in the global memory using a set of atomic operations, that generally incur a performance overhead.

Figure 64: Block Diagram of a modern GPU (Nvidia Kepler Architecture). The chip consists of a large pool of computation cores, shown as green. Memory buffers and cache memory are shown in blue. (Source: NVIDIA Corporation).

GPUs don't have a fixed *Instruction Set Architecture (ISA)*, like the x86 instruction set on CPUs. Instead, each hardware design uses a different ISA, optimized for the underlying architecture that is used. The operation of GPUs is controlled by graphics APIs, like OpenGL, which was pioneered by Silicon Graphics and DirectX, which was introduced by Microsoft. Shaders are written in a high level language, like GLSL (a mix of ANSI C and the RSL) and are compiled by the graphics driver to the native hardware ISA.

12.3. Stream Processing Model

In the *stream processing* model, a common set of instructions generally are executed independently on a large set of input elements, commonly referred to as *input stream*. In the general case multiple kernels and multiple input streams can exist in parallel. The conceptual diagram of this processing model is shown in *Figure 65*.

The most common problem that is solved with a stream processing model is the creation of digital images with *rendering* and other *image synthesis* algorithms. Many problems outside the image synthesis domain are embarrassingly parallel and have similar demands in terms of computations, where the same set of operations should be performed on an enormous set of individual elements. Modern GPUs can be used as general stream processors, in order to accelerate algorithms for these problems, too. In this case, the small program that gets executed on each data element is often called a *kernel* instead of shader, since the word shader implies that some form of shading operations are executed, which is not always the case for general purpose computing. Kernels are defined using the so called *compute APIs*, such as CUDA and OpenCL.

Each compute API uses a completely different terminology to describe the work that is performed on the GPU. The instances of the running kernels are often aggregated in groups, to allow for more efficient execution. We will refer to these groups of active kernels as *warps*. One warp on a typical GPU consists of 32 or 64 running kernel instances on contemporary Nvidia and AMD GPU architectures respectively. Xeon Phi/Larabee, a more general stream processor by Intel, operates on 16 floating point elements, using 512-bit wide SIMD units.

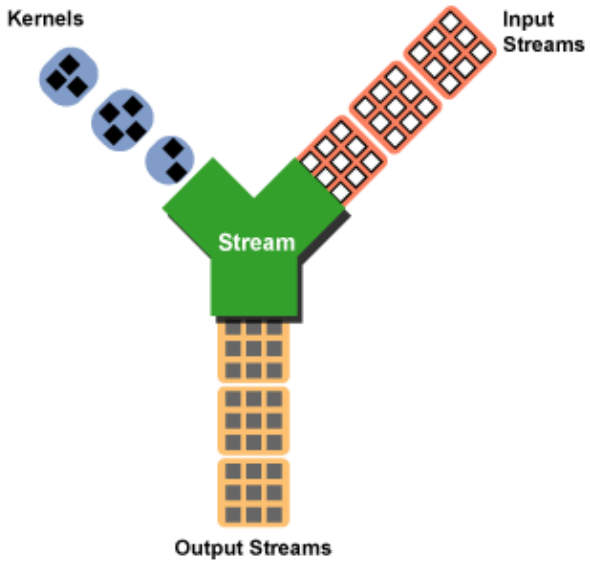


Figure 65: The stream processing model. The same set of operations (kernel) is executed on an array of elements (input stream) in order to produce an output stream. (Source: http://arstechnica.com/)

12.4. Resource utilization and latency hiding

A typical GPU or similar stream processor can achieve high throughput by having many warps running on the fly. When a warp needs to wait for a memory fetch to complete, the GPU can switch to the execution of another warp, thus increasing the total utilization of the hardware resources. However, this requires a much larger *register file* to hold data from multiple threads. The number of threads (shaders or kernels) that can be executed in parallel is highly dependent on the number of general registers that are used by the thread. Threads that utilize fewer registers tend to be executed more efficiently, because the hardware can keep a large number of them "in flight" (concurrently active, paused or running).

The scheduling of the different threads can be either dynamic, performed on-the-fly by the hardware, or the instructions can be pre-scheduled by the driver in software. By having thousands of warps available for execution, modern hardware can hide the latency introduced by memory transactions and increase the floating point throughput. Furthermore, in order to increase the total efficiency, the latest GPU architectures can execute warps of more than one kernel concurrently.

It should be noted that a similar measure that increases the utilization of the hardware resources and hides memory latency and other pipeline bubbles has been also successfully deployed on general purpose CPUs, using the *Simultaneous Multi-Threading (SMT)* technique, which is known as "hyper-threading" on Intel architectures. However, since general purpose CPUs often deal with problems that are not massively parallel, the SMT implementation on these architectures is limited to a small number of threads (2 in the case of Intel general-purpose CPUs and 4 on the Intel Xeon Phi stream processor).

12.5. Common Implementation Characteristics

In most common implementations of the stream processing model, each warp has one program counter and the same instruction is executed at the same time on every kernel instance in the warp. This is not mandated by the specification of any compute API, but it is how most stream processing hardware implementations work, especially on GPUs and other massively parallel architectures. This execution of kernels in an SIMD-fashion allows the amortization of the cost of instruction fetching between many kernel instances, and furthermore allows a more efficient hardware implementation with SIMD units. It is worth noting that even the software implementation of the Reyes algorithm, as detailed by Apodaca and Gritz [[228]], uses a similar approach in order to shade the micropolygon grids that are produced in this algorithm and has similar performance characteristics. Therefore, this design is common in both software and hardware implementations of the stream processing model.

In order to allow an efficient implementation of this SIMD-like execution models, most stream processing architectures includes memory *gather* and *scatter* instructions for vectorized I/O. These operations are generally very useful for vectorizing code. A memory gather instruction takes as an operant a vector of memory addresses and returns a vector with the contents of the memory at these addresses. Similarly, a scatter operation takes as an argument a vector of data and a vector of data addresses and writes the data to the corresponding memory addresses. Please note that gather instructions (but not scatter) has also been included in commodity general purpose CPUs that support the AVX2 instruction set, like Intel's Haswell architecture.

12.6. Conditional Statements and divergent branches

The most significant implication of the execution model that we have outlined above is that conditional statements with *divergent* branches are rather inefficient in stream processing architectures that follow this design, because the hardware typically has to execute both sides of a branch. Each time a code path is executed, only the kernel instances inside the warp that follow this specific code path should be affected. This is typically implemented using a mask of active kernel instances inside a

[228] A. Apodaca and Larry Gritz, Advanced renderman: Creating cgi for motion picture, 1st ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

warp. However, the specification of the existing compute APIs does not specify how this is performed or which side of the branch is executed first. And in fact, these characteristics are known to be different depending on the underlying hardware architecture.

In such architectures, it is generally more efficient to design algorithms without branches, or use conditional moves, when possible. A conditional move is an operation that performs a single move/copy only if a specific condition is met. This type of operation is a common type of optimization and is found in many ISAs, like x86, ARM, Itanium. Nevertheless, blindly turning a branch of code in a large set of conditional moves will not always result in a performance gain. It is the algorithms that should be designed/adapted in order to use minimal branching and not just the code.

12.7. Memory Architecture

Another aspect of modern GPUs that will certainly influence the direction of our research is their memory architecture. Existing high performance GPUs are physically separate chips that communicate with the CPU through a physical high-performance bus (PCI-express). Furthermore, the CPU and GPU have a separate set of memory chips directly attached to them, and each one operates on a separate memory address space. Sending data from the CPU to the GPU or reading data back requires an expensive copy operation over a physical bus. This relatively slow memory transfer, compared to the speed of the local memory that is directly attached to the GPU, can quickly become a performance bottleneck. Further overhead can be induced when the transferred data should be decompressed or converted to another data type (for example most CPUs do not directly support half-float data types). Additionally, depending on the architecture, some API calls require switching from *user mode* to *kernel mode*, which is very expensive on most platforms. For these reasons, any excessive communication between the CPU and the GPU should be avoided. High performance algorithms should be designed to be executed mostly on the GPU, with minimum CPU intervention. In this processing model, the CPU usually just feeds the GPU with data and commands, keeping the CPU-side processing minimal.

Stream processors and GPUs, much like other processing units, typically include a large amount of cache memory and other buffers, in order to improve the performance or memory fetches. For instance, the NVIDIA GK110 (Titan) architecture features 1.5MB of L2 cache memory and 64KB of L1. In order to better take advantage of the cache memory, any memory fetches should be spatially coherent.

12.7.1. Integrated Architectures and shared memory model

The separation of high performance CPUs and GPUs in two different dies (chips) that use two different memory pools is currently mandated by the limitations of the manufacturing technology. Merging these two chips in one die and creating a *system-on-chip* (*SoC)* will increase the total number of transistors and the total area of the chip, something that would negatively affect the amount of working chips that could be manufactured per silicon wafer (lower yields).

However, we should note that these restrictions will not always be true. For low-performing cores with relatively small transistor counts, we already have processors that integrate the general purpose CPU and GPU in one die. These processors are commonly used today (mid 2013) in mobile devices and other highly integrated platforms, like low-end desktop systems and game consoles. Further advances in manufacturing technology will allow the creation of higher performing integrated architectures in the future, as implied by the *Moore's law*. The so-called *Moore's law* indicates that the number of transistors in integrated circuits rises exponentially with time, so it is reasonable to expect that in the future increasingly complex CPUs and GPUs will be integrated in one die, providing increasingly higher performance.

Such integrated systems can share a common memory space for both CPU and GPU, completely avoiding the overhead of copying data between the two. This would allow a closer collaboration of these two processing units. In fact, many integrated architectures that have been announced for the future by both AMD (Jaguar architecture, used in PS4 and Xbox One) and Intel (Haswell architecture) follow this shared memory model. However, please also note that a common memory space between

the CPU and the GPU does not necessarily require the integration of the two processors in one die, but can also be implemented by using a common memory controller for the two chips.

It should also be noted that the performance of an integrated system, apart from the manufacturing challenges that we have discussed earlier, is also limited by the thermal characteristics and the power envelope of the system-on-chip module. The amount of heat that can be removed from the small surface of the chip using a cooling system (which typically consists of a heat spreader and a fan) is limited by the laws of physics. Concentrating more chips and transistors in such a small area makes the required cooling harder and more expensive to achieve.

For the reasons discussed above (limited power envelope and manufacturing capacity), existing integrated systems typically have fewer transistors than their non-integrated counterparts. However, having a common memory space is a big advantage for many algorithms. Therefore, we expect that in the future, many architectures, integrated or not, will follow this memory model.

12.7.2. Growth of memory access speed and processing power

As often stated in the bibliography [229], for the past decades, computational speed is advancing at a faster rate than memory speed, as shown in Figure 66. This is a general trend in computing that we expect to continue in the future. Based on this observation of past trends, we conclude that it is more important to minimize the memory bandwidth of an algorithm than the ALU operations.

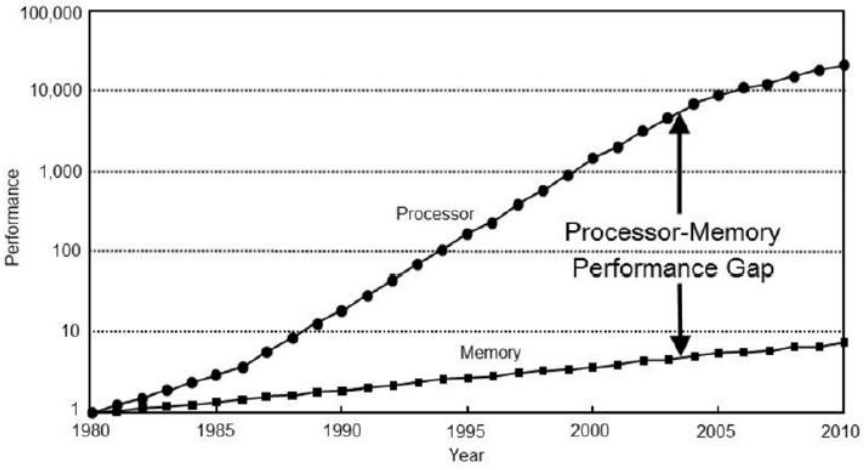


Figure 66: For the past decades computational power has grown at a faster rate than the available memory bandwidth. Source: [230].

12.8. General strategies for synergistic CPU and GPU algorithms

Since most typical computing platforms, from mobile devices to super-computers, include both a general purpose CPU and a massively parallel GPU, the general strategy in order to exploit the tremendous computational power of these two processing units is to determine the most appropriate parts of an algorithm to run on each side (CPU / GPU) with minimal data exchange between the two sides.

The data exchange should be kept at a minimum level for two reasons. First, as we have discussed earlier in this section, on existing systems the communication cost between the CPU and GPU is rather high. But most importantly, even if this was not true, any excessive communication between parallel processes can easily lead to circumstances where one process is idling, waiting to receive data from

229 John Owens, Streaming architectures and technology trends, GPU Gems 2: Programming Techniques, Tips, and Tricks for Real-Time Graphics, Addison Wesley, 2005, pp. 457–470.

230 John L. Hennessy and David A. Patterson, *Computer architecture, fourth edition: A quantitative approach*, Morgan Kaufmann Publishers Inc.,San Francisco, CA, USA, 2006.

another one (or in the general case waiting for another one to release a shared mutex), something that leads to sub-optimal utilization of the available resources. For this reason, efficient parallel algorithms generally strive to minimize the communication between the instances that are executed in parallel.

The ideal case, and the most effective strategy for existing hardware architectures, is to create an algorithm that can be executed efficiently, from the beginning to the end, on a stream processor. Such an algorithm would never require a round-trip of data from the CPU to the GPU and back. In this ideal case, even if the algorithm is executed using many different kernels and performs many passes over the input stream, the actual data will always stay on the GPU, until the final results are produced. The CPU is used to "feed" additional commands (kernels) and data to the stream processor, but never reads intermediate results back. Most existing research on GPU-accelerated algorithms strives to implement this processing model, which guarantees minimal data exchange.

However, some parts of an algorithm might not map very well on the stream processing model, because they rely heavily on branching or they are generally serial in nature. Currently, the best way to deal with these cases is to find an alternative (or new) algorithm that performs the required task and is more suitable for execution on a stream processor. When this is not possible, one can execute different parts of a method on a CPU and GPU, but the communication overhead between the two units should never be higher that the speedup from executing part of the algorithm on the GPU.

A representative example of such an algorithm is the "*Screen space photon mapping*" method by McGuire and Luebke [231]. In the first pass of this photon mapping algorithm, photons are traced from the light sources into the scene and stored in a *k*-d tree. This is typically performed on the CPU, but the authors demonstrate that the first bounce of photons, in the case of point lights, can be efficiently computed using hardware rasterization on the GPU, since in this case all the photon paths are coherent and originate from the same point. The rest of the bounces are computed on the CPU using ray-tracing and the final positions of the photons are transferred back to the GPU, in order to render the final scene. However, this roundtrip of information between the CPU and GPU incurs a high performance overhead on most existing systems and for this reason this technique was not widely adopted. A more successful strategy, for the existing hardware architectures, would be to use a GPU-friendly ray-tracing acceleration structure and perform the entire algorithm on the GPU. To this end, the OptiX [232] rendering architecture has been later developed.

12.9. Discussion

In this section we have seen that GPUs and stream processors provide a general processing model that can be used to solve many massively parallel problems. In the context of fractured object reassembly, we pinpoint the following areas where a GPU or a general stream processor can contribute:

- **Uniform sampling of data**: The fixed function rasterization hardware of the GPUs can be used to efficiently compute a uniform sampling of the input meshes, either on two dimensions, as performed by Papaioannou et al. [233], or in three dimensions, using an efficient GPU accelerated voxelization algorithm, as the one presented by Gaitatzes et al. [234].

[231] McGuire, Morgan, and David Luebke. "Hardware-accelerated global illumination by image space photon mapping." Proceedings of the Conference on High Performance Graphics 2009. ACM, 2009.

[232] Parker, Steven G., James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister et al. "Optix: a general purpose ray tracing engine." ACM Transactions on Graphics (TOG) 29, no. 4 (2010): 66.

[233] Papaioannou, Georgios, E-A. Karabassi, and Theoharis Theoharis. "Virtual archaeologist: Assembling the past." Computer Graphics and Applications, IEEE 21.2 (2001): 53-59.

[234] Gaitatzes, Athanasios, Pavlos Mavridis, and Georgios Papaioannou. "Two Simple Single-pass GPU methods for Multi-channel Surface Voxelization of Dynamic Scenes." 19th Pacific Conference on Computer Graphics and Applications-short papers (PG) pp. 2011.

- **Efficient descriptor computation**: The computation of a "*descriptor*" on each input element (point or face) can be seen as a gathering operation, where the final value is computed by reading and performing computations on the local neighborhood around each element. Gathering operations can typically implemented efficiently on GPUs, provided that the neighborhood of each element can be efficiently accessed, without many diverging branching operations. This could be rather challenging if a graph-based representation is used, since each vertex can have an arbitrary (but typically bounded) valence.
- **Efficient linear algebra operations**: Many classes of algorithms, including methods for segmentation, matching and registration, require the efficient solution of linear algebra problems. These problems can be efficiently solved on GPUs using the appropriate libraries, like cuBLAS or cuSparse, that provide a large performance gain over the commonly used CPU-based implementations.
- **General parallel computations**: Any part of an algorithm where the same set of instructions should be performed on a large se of data, with minimal divergent code paths.

13. ACKNOWLEDGEMENTS

We would like to thank Georgios Tsatiris for contributing the survey on the ICP algorithm and its variants.