Towards Parameter-less 3D Mesh Segmentation

by

Sara Farag
M.Sc, B.Sc

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

Deakin University

October, 2013

DEAKIN UNIVERSITY
ACCESS TO THESIS - A

I am the author of the thesis entitled

Towards Parameter-less 3D Mesh Segmentation

submitted for the degree of **Doctor of Philosophy**

This thesis may be made available for consultation, loan and limited copying in accordance with the Copyright Act 1968.

'I certify that I am the student named below and that the information provided in the form is correct'

Full Name: .....**Sara Farag**.....
(Please Print)

Signed: ..... Signature Redacted by Library .....

Date: .....**31/10/2013**.....

DEAKIN UNIVERSITY
CANDIDATE DECLARATION

I certify the following about the thesis entitled (10 word maximum)

Towards Parameter-less 3D Mesh Segmentation

submitted for the degree of **Doctor of Philosophy**

a. I am the creator of all or part of the whole work(s) (including content and layout) and that where reference is made to the work of others, due acknowledgment is given.

b. The work(s) are not in any way a violation or infringement of any copyright, trademark, patent, or other rights whatsoever of any person.

c. That if the work(s) have been commissioned, sponsored or supported by any organisation, I have fulfilled all of the obligations required by such contract or agreement.

I also certify that any material in the thesis which has been accepted for a degree or diploma by any university or institution is identified in the text.

'I certify that I am the student named below and that the information provided in the form is correct'

Full Name: .....**Sara Farag**.....
(Please Print)

Signed: ..... Signature Redacted by Library .....

Date: .....**31/10/2013**.....

To my lovely husband, children and parents.

Acknowledgments

I would like to express my gratitude to my supervisors, Prof. Saeid Nahavandi and Dr. Douglas Creighton, for their unlimited continuous support.

I would also like to thank all the members of the Center for Intelligent Systems Research (CISR) at Deakin University.

Very special thanks to my family for their encouragement and support. My deepest gratitude to my husband, my beloved children, my parents, my sister, and my brother.

List of Publications

1. Abdelrahman, W. and Farag, S. ”Automated 3D mesh segmentation using 2D footprints” ACM SIGGRAPH 2010 Posters, ACM, 2010

2. Abdelrahman, W.; Farag, S.; Nahavandi, S. and Creighton, D. Simon, R. and Akihiko, S. (Eds.) ”Adaptive Automatic Deformation Basis Generation for Haptic Interaction with Physically Deformable Models” Proceedings of Virtual Reality International Conference (VRIC 2010), pp.1-7, 2010

3. Abdelrahman, W.; Farag, S.; Nahavandi, S. and Creighton, D. ”Data-based dynamic haptic interaction model with deformable 3D objects” 8th IEEE International Conference on Industrial Informatics (INDIN) , pp.314 -318, IEEE, 2010

4. Farag, S. and Abdelrahman, W. ”Physical modeling of heterogeneous embed- ded deformable object deformation” ACM SIGGRAPH 2010 Posters, ACM, 2010

5. Abdelrahman, W.; Farag, S.; Nahavandi, S. and Creighton, D. ”A comparative study of supervised learning techniques for data-driven haptic simulation” IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp.2842-2846, IEEE, 2011

6. Farag, S.; Abdelrahman, W.; Nahavandi, S. and Creighton, D. ”Physically based simulation of heterogeneous deformable models using XFEM” 9th IEEE International Conference on Industrial Informatics (INDIN), 233-237, IEEE, 2011

7. Farag, S.; Abdelrahman, W.; Nahavandi, S. and Creighton, D. ”Towards a Parameterless 3D Mesh Segmentation” International Conference on Graphic and Image Processing (ICGIP 2012), vol.8768, pp.87682O-87682O-8, SPIE, 2013

8. Farag, S.; Abdelrahman, W.; Creighton, D. and Nahavandi, S. ”Extracting 3D mesh skeletons using antipodal points locations” 15th International Conference on Computer Modelling and Simulation (UKSim), pp.135-139, 2013

9. Farag, S.; Abdelrahman, W.; Creighton, D. and Nahavandi, S. ”Automatic deformation transfer for data-driven haptic rendering” 11th IEEE International Conference on Industrial Informatics (INDIN), IEEE , 2013

Table of Contents

Table of Contents v

List of Figures x

Abstract xiv

Acronyms xvi

1 Introduction 1

1.1 Challenges in 3D mesh segmentation 3

1.1.1 Parameter dependency 3

1.1.2 Output evaluation 4

1.1.3 Speed and real-time performance 5

1.1.4 Benchmarking literature algorithms 5

1.2 Contributions 6

1.3 Thesis Organization 7

2 Literature review 9

2.1 3D mesh segmentation 10
2.1.1 3D mesh segmentation definitions and applications 10
2.1.2 Types and classifications of 3D mesh segmentation techniques 12
2.2 Related work 15
2.2.1 Interactive approaches 15
2.2.2 Semi-automatic approaches 18
2.2.3 Automatic approaches 25
2.3 Critical review of 3D mesh segmentation techniques literature . . . 27
2.4 Proposed framework 33
2.5 Summary 35

3 Automation of the 3D mesh segmentation process 36

3.1 Transforming the problem from 3D to 2D 36
3.1.1 Footprint calculation algorithms 37
3.1.2 2D footprint for 3D mesh segmentation 42
3.2 Antipodal location 44
3.2.1 Introduction to antipodal location techniques 45
3.2.2 Literature review of antipodal location techniques 48
3.2.3 Antipodal point definition 49
3.2.4 Antipodal point computation methodology 55
3.2.5 Experimental results of the proposed antipodal point location techniques 58

3.3 Connecting two points in a mesh 65
3.3.1 Related work of closed contour creation 65
3.3.2 Proposed closed contour formation algorithm 66
3.4 Output Smoothing of the 3D mesh segmentation process 67
3.4.1 Converting cut lines to 3D parts 68
3.5 Conclusion 69

4 Automated 3D mesh segmentation framework 70

4.1 Input formats and pre-processing 70
4.2 3D mesh segmentation processing 72
4.2.1 3D mesh segmentation framework 74
4.2.2 Complexity analysis of the proposed 3D mesh segmentation algorithms 75
4.3 Output and results of the proposed 3D segmentation algorithms . . . 85
4.4 Benchmarking of the proposed 3D mesh segmentation algorithms . 85
4.5 Conclusion 97

5 Study of deformation transfer between isometric objects 99

5.1 Deformation transfer preliminaries 100
5.2 Related work of deformation transfer techniques 101
5.3 Deformation transfer problem statement 103
5.4 The Proposed deformation transfer algorithm 105
5.5 Antipodal location 108

5.5.1 2D antipodal vertex calculation . . . 109
5.5.2 3D antipodal vertex calculation . . . 111
5.6 Implementation and results of the proposed deformation transfer algorithm . . . 113
5.7 Conclusion . . . 114

6 Study of 3D mesh skeletonisation 118

6.1 Introduction to 3D mesh skeletonisation process . . . 119
6.2 Literature review of 3D mesh skeletonisation techniques . . . 120
6.3 The proposed 3D mesh skeletonisation algorithm . . . 122
6.3.1 Preliminaries . . . 122
6.3.2 Preprocessing . . . 123
6.3.3 Main module of the skeletonisation algorithm . . . 124
6.4 Experimental results of the proposed 3D mesh skeletonisation algorithm . . . 126
6.5 Conclusion . . . 127

7 Conclusion 130

7.1 Thesis Contributions . . . 130
7.2 Potential applications . . . 132
7.2.1 Online generic 3D skeletonisation using depth cameras . . . 132
7.2.2 Shape correspondence for data-driven haptic simulation of deformable models . . . 133

7.3 Future research directions 135

References **137**

List of Figures

1.1 Current challenges of the 3D mesh segmentation process. 4

2.1 3D mesh segmentation flow diagram. 13
2.2 3D mesh segmentation high level classification. 14
2.3 Part 3D mesh segmentation vs. surface 3D mesh segmentation . . . 15
2.4 Example of interactive 3D mesh segmentation applications 16
2.5 The minima rule defines the seeds of cutlines. 19
2.6 Region growing vs. multiple source region growing 21
2.7 Shape diameter function calculation. 26
2.8 A 3D model that does not have obvious cutlines. 33
2.9 A high level flow chart of the proposed 3D segmentation framework. 34

3.1 Dynamic Spatial Approximation Method (DSAM) for estimating spatial footprints 38
3.2 Swinging arm algorithm steps 39
3.3 Regular and irregular footprints from topological viewpoint 40
3.4 Simply connected vs. non simply connected footprints 41
3.5 alphahull package usage, sample code, and sample output for a camel 3D object. 43

3.6 Example of a different resolution CAD 3D mesh and the corresponding 2D footprint. 44
3.7 Example of vertex antipodal. 47
3.8 2D antipodal solutions for different curve pairs. 49
3.9 Visualisation of a 3D object at each processing step for a 2D antipodal search. 51
3.10 A vertex antipodal pair is not necessarily a commutative pair (2D example). 53
3.11 2D antipodal location search problems. 54
3.12 Data flow diagram of computing the 2D and 3D antipodal points. . . 56
3.13 Possible solutions to navies' antipodal selection using point in polygon algorithm. 57
3.14 Examples of 2D and 3D search results for different classes of objects. 59
3.14 Examples of 2D and 3D search results for different classes of objects. 60
3.14 Examples of 2D and 3D search results for different classes of objects. 61
3.15 The shortest distance between two points is performance measure for the 3D antipodal search. 62
3.16 The 2D/3D search performance with respect to the mesh vertices count. 63
3.17 The automatic evaluation of the accuracy using ellipse fit algorithm. 64
3.18 Converting cutlines into sub parts in 3D mesh. 68

4.1 The calculation steps of the 3D mesh cutlines. 73
4.2 The values of angles between the faces of the teddy 3D mesh. 74

4.3 Data flow diagram of the 3D segmentation framework. 75
4.4 Experimental results for the teddy 3D object against other literature methods. 86
4.5 Experimental results for the ant 3D object against other literature methods. 87
4.6 Experimental results for the hand 3D object against other literature methods. 88
4.7 Experimental results for the octopus 3D object against other literature methods. 89
4.8 Experimental results for the chair 3D object against other literature methods. 90
4.9 Experimental results for the cup 3D object against other literature methods. 91
4.10 Experimental results for the glasses 3D object against other literature techniques. 92
4.11 Benchmark results for the CDI metric. 93
4.12 Benchmark results for the HDI metric. 94
4.13 Benchmark results for the GCI metric. 95
4.14 Benchmark results for the LCI metric. 96
4.15 Benchmark results for the RI metric. 97

5.1 The proposed algorithm flowchart. 106
5.2 Identification of high curvature points in 3D meshes. 107
5.3 Example of vertex antipodal. 109
5.4 Visualisation of 3D object at each processing step for 2D antipodal search. 110

5.5 Possible solutions to navies' antipodal selection using point in polygon algorithm. 112
5.6 Ellipse fit example to guide contour completion. 113
5.7 The results of a hand mesh segmentation into convex parts. 116
5.8 Isometric mesh can be matched after being decomposed. 117

6.1 Example of skeletons for various 3D objects 119
6.2 Antipodal point and skeleton points definition 124
6.3 Data flow diagram of the algorithm steps to compute the 2D and 3D antipodal points. 125
6.4 Skeleton results for selected 3D objects classes with a different vertices count 128
6.5 Skeleton results for the hand 3D object when compared with literature methods. 129

7.1 Acquisition system of full 3D point cloud using multiple depth sensors. 133
7.2 Shape correspondence can identify the areas of similarities and differences. The data collection can be only done for the different areas. 134

Abstract

This thesis investigates and proposes algorithms that support the automation of 3D mesh segmentation process. 3D mesh segmentation is an important task employed to decompose 3D mesh into meaningful parts according to application context. The process suffers from being subjective, as the output can differ from one person to another. This has prohibited the automation of the process and made it harder to be parameter-free. The automation of such technology enables so many dependent applications to advance. Examples include 3D mesh databases, 3D animation and deformation transfer. Deformation transfer specifically was one of the motivations for this research as it is an efficient methodology for mapping deformations among several objects.

The current 3D mesh segmentation literature includes several methods that try to optimize the resultant 3D parts quality to be as close as possible to the common minima rule criteria. This has reached plausible quality in recent years and even hybrid approaches have emerged that augment multiple techniques power. However, further investigation has the potential to discover new approaches for some areas in the field. According to the latest surveys and the author's critical review, the missing parts are in the input parameters dependency and the real-time processing ability. These two areas can boost the field of 3D mesh segmentation enormously and make it a handy method for many studies such as autonomous robots.

Algorithms are proposed to tackle the area of parameter dependency by designing a parameter-free framework. If parameters are inevitable, they will be mesh independent and just output controlling parameters that for instance increase or decrease the number of cutlines. The algorithm transforms the problem from the 3D domain into the 2D one. Through the aid of topology algorithms and statistical algorithms, the algorithm can locate the cutlines of the input mesh using an input criteria such as the minima rule. The topology algorithm such as 2D footprint, vertex antipodal location, and graph theory algorithms adds more information to the input mesh. The statistical algorithms manipulate the data sets and enable optimized mining of knowledge. The algorithm main core loops feature independence between the mesh vertex ,which enables the parallelization of them.

The innovation in this approach emerges from the fact that it is generic for any mesh, parallel-processing ready, and independent of the cut criteria. This makes the provided algorithms a base for many applications of the 3D mesh segmentation process. An important part of the algorithm, which is the vertex antipodal location, can also serve other applications as is and can solve many problems in the mesh processing domain.

The proposed algorithms have been tested and benchmarked against the latest works and databases from the literature. The algorithm show competitive accuracy when compared with the literature. Even if the benchmarking values are not the best along all metrics, the algorithm can still be favored as it is an automated one, which works autonomously and in batch modes. The impact of these proposed algorithms is tested in a computer haptics domain application, which shows how deformation can be automatically transferred between objects that share certain similarities.

Acronyms

3D-PRI Three dimensional probabilistic random index

ADB All dots on boundary

AGD Accurate geodesic distance

ANN Artificial neural networks

CAD Computer aided design

CDI Cut discrepancy index

CED Curvature extreme at dots

CISR Center for intelligent systems research

COTS Commercial of the shelf

CPU Central processing unit

CSG Constructive solid geometry

CT Computed tomography

DSAM Dynamic spatial approximation method

EM Expectation maximization algorithm

FC Full coverage

FEM Finite elements methods

FVM Finite volume methods

GCI Global consistency index

GPGPU General Purpose computing on Graphics Processing Units

GPU Graphics Processing Unit

HDI Hamming distance index

JC Jordan components

LCI Local consistency index

MAT Medial axis transform

MGF Medial geodesic function

NDB No dots on boundary

NP Nondeterministic polynomial

NPRI Normalized probabilistic random index

OI Overlap index

RBF Radial basis functions

RI Random index

SCC Simply connected components

SDF Shape diameter function

SNN Shared nearest neighbor

XFEM Extended finite element methods

Chapter 1

Introduction

The processing of objects can be made easier when they can be decomposed into smaller parts. These parts are extremely useful when they can be associated with meta data that makes them identifiable to humans and automated processing algorithms. This argument can be seen in nearly all fields such as medical analysis of the human body or geographical analysis of the planet earth. 2D Images and 3D mesh follow the same rules and are easily identifiable when decomposed into smaller parts. 2D segmentation has attracted a lot of interest from researchers and commercial applications are already in the market such as advanced photo editing tools. 3D mesh segmentation is relatively new compared to the 2D segmentation domain and is now required to catch up to the rapid advances in 3D rendering hardware and software.

A high level definition of 3D mesh segmentation is that it is the process of decomposing an input mesh into parts based on a provided criteria. As will be seen in the next chapter, there are variants of this process and its inputs and outputs. This thesis will focus only on 3D boundary meshes and on part-type decomposition. Many principles can be adopted from this thesis to apply to other types of 3D mesh

segmentation but with certain adaptations. 3D boundary mesh is usually a set of vertices, edges, and faces that represent the boundary of the represented topology.

3D mesh segmentation is not a totally new field and many concepts were considered from related fields such as 2D image segmentation, finite element mesh partitioning and statistical clustering. The first approaches to such problems always start with semi automated techniques that ask the user for more information rather than inferring them from the input object. In 3D segmentation this can be critical and anti-automation such as asking the user for the segments count [1], or it can be just for tuning parameters [2], can be automated later on if there is an automated testing method.

The motivation behind this research is the full automation of the 3D mesh segmentation process. The target is to have a parameter-free framework that can infer cutlines without dependency on objects or users. This target is essential for the evolution of the 3D mesh segmentation domain from theory to be practically used by experts and non-experts of 3D computer graphics. Besides, batch processing of 3D mesh is crucial if 3D databases are to be realized as searchable for automatic engines.

To fully achieve this goal, there are certain questions that need to be investigated:

1. Is there a method that people commonly follow to cut 3D mesh?
2. Is the data of the 3D mesh, which comprises only vertices and faces enough for the processing?
3. Can this method be automated?

1.1 Challenges in 3D mesh segmentation

The 3D mesh segmentation domain is a relatively new one and still requires more development [3]. The main challenges that need further research are: input mesh dependent parameters, automatic evaluation of output, real-time processing, and benchmarking of different algorithms. These different aspects are challenging tasks but having them realized is necessary for the future of the topic. The important notice here is that without the first challenge, which is the full automation of the process, all the validity of other challenges will be in question. Thus, it is important to make sure that existing parameters dependency is relaxed and removed. Figure 1.1 shows the current challenges and the effect of the process automation on its efficiency and utilization.

1.1.1 Parameter dependency

Parameter dependency or the presence of the human in the loop is a drawback of most of the existing 3D mesh segmentation techniques. Critical parameters, such as the number of segments, need to be avoided absolutely. Tuning parameters such as the internal clustering algorithm can be tolerated given that a generic input will produce an acceptable output. The ultimate goal is to eliminate the need for parameters. However this is not an easy task due to two factors. The first is that human beings themselves may segment the same mesh in different ways. The second is that the process of segmentation is not yet deterministic with a single solution. Different solution can still be accepted as the mesh originally represent a continuous domain and discretized vertices do not capture the whole continuum.

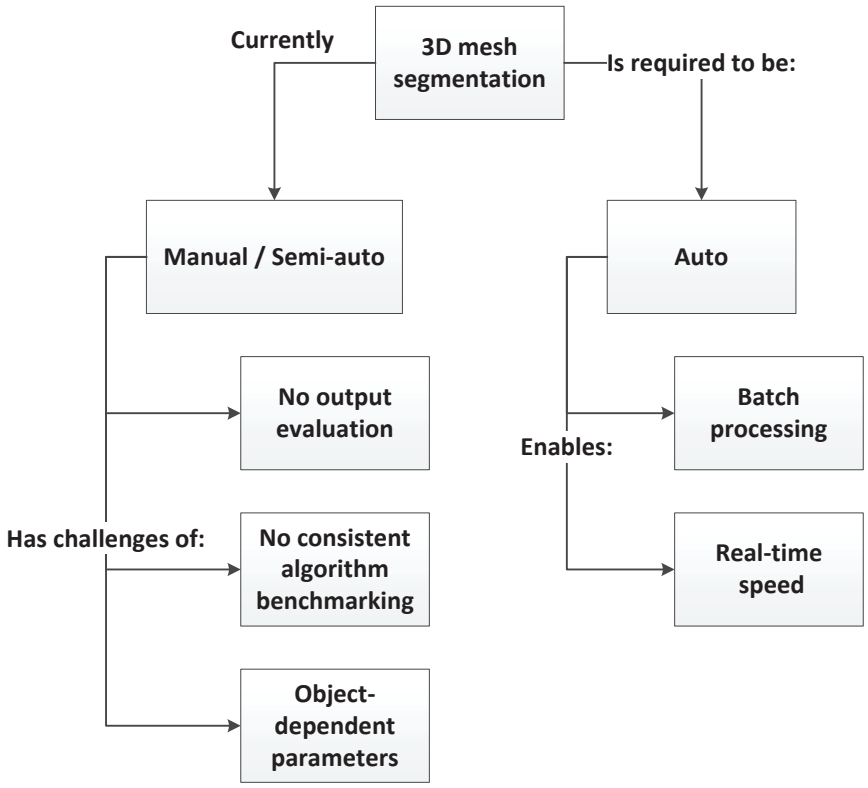


Figure 1.1: Current challenges of the 3D mesh segmentation process.

1.1.2 Output evaluation

The output of the 3D segmentation process is a set of cutlines or a set of sub-meshes. Both of these sets are currently evaluated manually through the human eye. Automating such process of output evaluation is a challenging one. The main problem is that the small output parts are not easily recognized semantically. Without automating the evaluation part, the segmentation process will not be a fully deterministic one.

The key to solve such a challenge is the addition of meta data and characteristics of the whole input mesh and its possible sub parts. This has to follow pattern recognition methodologies to be able to expect the output and judge its validity and

quality. An example would be that a human body is expected to have six main parts: The head, body, hands, and legs. A leg is expected to follow a cylindrical topology and so on. This needs hybrid approaches that can make the best combination of 2D and 3D domains.

1.1.3 Speed and real-time performance

The advances in computer hardware and software created a high demand for real time processing powers. An online algorithm is highly desirable for the interactive nature of immersive computing environments. The 3D mesh segmentation process is still a resource consuming process. This is due to the fact that it needs to process nonlinear inputs and cluster them using nonlinear algorithms [4] from a performance point of view.

Parallelization of the process is one of the main solutions to such a problem. Currently, central processing units (CPUs) and graphical processing units (GPUs) feature multiple cores, which can be really a large number as in the case of modern GPUs [5]. However, the core question is whether the segmentation process is ready to be parallelized or not. To have this feature, the process core time consuming loops needs to have a certain level of independence on the input data to each loop iteration. Besides, the data needs to be easily integrable at the end of the independent processing threads.

1.1.4 Benchmarking literature algorithms

Benchmarking 3D mesh segmentation algorithms output is a major challenge that has attracted high attention in recent years. Chen *et al.* [6] have proposed a set of

metrics and have collected a benchmark database generated from users feedback online. The main focus was to build the metrics and the database to reflect the quality of the literature algorithms when compared to human collected samples. However, there is the main problem that needs to be attended as well. The algorithms do not operate on the same input parameters. They are usually semi-automatic and can have advantages over each other based on the input parameters.

For these benchmarking tools (i.e. reference data and metrics), the algorithms need to be fully automated with the same input and same expected output. New metrics might even arise when the algorithms are ready that test other measures such as speed and convergence.

1.2 Contributions

The mentioned major challenges and others make the 3D mesh segmentation domain a demanding problem for further research and investigation. This thesis contributes to the main major challenge, which is the automation of the process and also enables further improvements towards real-time speed and optimization. As mentioned before, parts of the proposed algorithm can also be used as stand-alone solutions for other problems in computer graphics and other domains. The contributions of this thesis can be listed as folllows:

Generic parameter-free 3D mesh segmentation algorithm : The proposed algorithms are independent of the input 3D mesh altogether and avoid the involvement of human feeds. The scissoring criteria is an input to the algorithm and thus it can be adapted based on the application context. Usual application utilizes the minima rule from the cognition theory but other criteria are allowed, such as diameter-based segmentation [7].

Parallelization-ready algorithms : The algorithm loops operate on each vertex independently, which enables parallel processing of the input mesh using the advances in GPU computing.

Antipodal location algorithm : This algorithm is an important methodology in the problem of 3D mesh segmentation. It also serves other problems in the domains of robotics, topology, and geography.

Automation of related fields : A direct benefit from the proposed algorithms is the algorithm of deformation transfer, which used to be done manually. The algorithm can now operate autonomously and in batch mode as well. A study in computer haptics is shown that makes use of this important methodology to collect data. Another common use of the proposed algorithm is in generating 3D mesh skeletons. The proposed algorithm produce qualitative skeletons autonomously with desired traits such as being centered and hierarchical.

1.3 Thesis Organization

The thesis is divided into seven chapters, the rest of the thesis will be organized as follows:

Chapter 2 presents the literature review and related work. This includes a critical review and shows how the proposed research fits with the literature. The chapter also has high level view of the proposed framework.

Chapter 3 demonstrates the supporting methodologies of the different modules of the proposed algorithm. This covers different phases of the segmentation process, starting with the data preprocessing phase, and ending with the results

postprocessing one. The discussed algorithms include 2D and 3D antipodal location, cutlines connection, and cutlines smoothing.

Chapter 4 shows the main framework of the proposed 3D segmentation algorithm. This is accompanied by results, complexity analysis, and comparative studies against the literature algorithms using the recent benchmarking metrics. The used metrics are: The used metrics are: cut discrepancy index (CDI), Hamming distance index (HDI), global consistency index (GCI), local consistency index (LCI), and rand index (RI). These metrics are proposed in the literature and have been used in several publications.

Chapter 5 discusses the study of deformation transfer between isometric objects using the proposed algorithm. The chapter investigates the problem statement, proposed solution and experimental results.

Chapter 6 investigates the 3D mesh skeletonisation study. The skeletonisation of 3D meshes is a related field to the 3D mesh segmentation. The automation of the process supports many application such as gait analysis and 3D animation.

Chapter 7 is the last chapter. This chapter is dedicated for concluding remarks and possible future research directions. 3D mesh segmentation is a rapidly growing field and continuous development is required to make it close to classic related fields such as 2D image segmentation.

Chapter 2

Literature review

The literature of 3D mesh segmentation is not relatively large. This is due to the fact that 3D mesh segmentation is in its beginning when compared, for instance, with image segmentation [3]. However, in recent years an increasing number of researchers have been attracted to the field. The field has many potential areas for contribution and the 3D application in general are debated topics because of the widespread of recording and displaying tools [8]. The 3D segmentation is an essential tool in the 3D geometrical processing field. This chapter aims to survey the literature, critically define the gap areas, and lay the foundation for the rest of the thesis.

The 3D mesh segmentation problem is a challenging one. Under certain conditions [9] it can be a nondeterministic polynomial time (NP) complete problem [10]. If the sub meshes count is equal to k and the number of mesh elements is equal to n, then the search space is of order k^n. This means that a complete enumeration of all possible solutions is not possible because of the large size of the solution space. An approximate solution is sought instead and thus it needs careful inspection.

2.1 3D mesh segmentation

The field of 3D geometric processing started to grow with the increasing demand of 3D graphics content. The processing of the 3D input utilizes algorithms from mathematics, computer science and other fields to transform it into other formats. These formats enable efficient storage [11], manipulation [12], and analysis [13] that cannot be easily done on the raw 3D formats. 3D mesh segmentation is one of these geometric processing algorithms that targets the simplification of the 3D models into smaller meaningful parts within a context. The output of the segmentation process is useful in many applications such as 3D mesh databases, 3D animation, mesh parametrization and others.

2.1.1 3D mesh segmentation definitions and applications

The 3D mesh segmentation can be defined basically over a 3D mesh according to [4] as:

"3D Mesh segmentation $\sum$: Let M be a 3D boundary-mesh, and S the set of mesh elements, which is either V, E or F. A segmentation $\sum$ of M is the set of sub-meshes $\sum = \{M_0, ..., M_{k-1}\}$ induced by a partition of S into k disjoint sub-sets."

The segmentation is done based on a criteria function J, which needs to be minimised or maximised under set of constraints C. C can be empty for simple cases. The J function works as a classifier that places the mesh elements into separate sets. The criteria function is application dependent and while general ones are usually used, customised ones can be designed.

An example of the process elements can be seen when using a shape diameter function (SDF) criteria function. The function can be defined as the variance of the distance between a mesh vertex and its antipodal vertex. The goal here would be to

minimize the variance and a required constraint would be that a certain number of segments is expected.

The 3D mesh segmentation can be used as a stand-alone process and as a tool within other processes. The segmentation into smaller parts enables semantic labeling of the sub parts. This facilitates sorting, searching, and characterising the small parts. Applications that can make use of this include 3D databases, 3D skeleton extraction, mesh parametrization, morphing and deformation transfer. Many of these are heavily dependent on the accuracy of the segmentation and in most cases cannot work properly without it.

Many other domains from the computer science literature share similar concepts with the 3D mesh segmentation. This includes 2D segmentation and clustering domains. All of these share the properties of being search problems in large spaces where heuristics are required to bound the time and guarantee convergence and results fidelity. These shared properties affected the early development of the 3D mesh segmentation techniques and many used similar concepts from the related domains. Unfortunately, this created a set of limitations, as will be seen in the next section, on the output and utilization of the 3D mesh segmentation.

The input to the process of 3D segmentation is usually a 3D boundary mesh. An example of another input is computed tomography (CT) scans [14, 15, 16], which can be reconstructed or processed individually. The common 3D mesh input format is a challenging one due to the following factors:

1. There is no function to describe the whole mesh or locate its vertices. The mesh is a set of components, which are vertices, faces, and edges.

2. The 3D mesh does not have any meta data associated with its individual components.

3. There are no predefined relations or grouping among the mesh components.

The output of the 3D segmentation process can have different formats. The general classification of the output format is to be disjointed sets of mesh components or set of cutlines that outline the borders of these disjointed sets. Each of these formats can be inducted from the other and thus either of them is usually acceptable. The main challenge with the output of the 3D segmentation is that it is not unique. Segmentation used criteria determines the output and the criteria is usually based on the application context. Another challenge is the validation of the output. So far, the recent benchmarks proposed in the literature [6] use manual segmentation as their reference. This is due to the nature of the segmentation process, which is subjective rather than objective [17].

Segmentation criteria is an important part of the 3D segmentation algorithm. Examples of used criteria include diameter length [7], resemblance to geometrical objects such as ellipsoids [18], and minima rule [17]. The minima rule specifically is one of the common approaches for segmentation and according to the cognitive theory, is the closest to the human approach [19]. Thus, in this thesis, the minima rule will be used. However, the proposed algorithms are open for any other criteria. Figure 2.1 shows the 3D segmentation process flow, possible inputs, and outputs.

2.1.2 Types and classifications of 3D mesh segmentation techniques

Using a high-level classification [3], there are two types of 3D mesh segmentation: surface-based type and part-based type. Both of these types of algorithms can be further classified, as shown in figure 2.2, based on the usage mode to: interactive, semi-automatic, and automatic. The two extremes are being totally interactive or

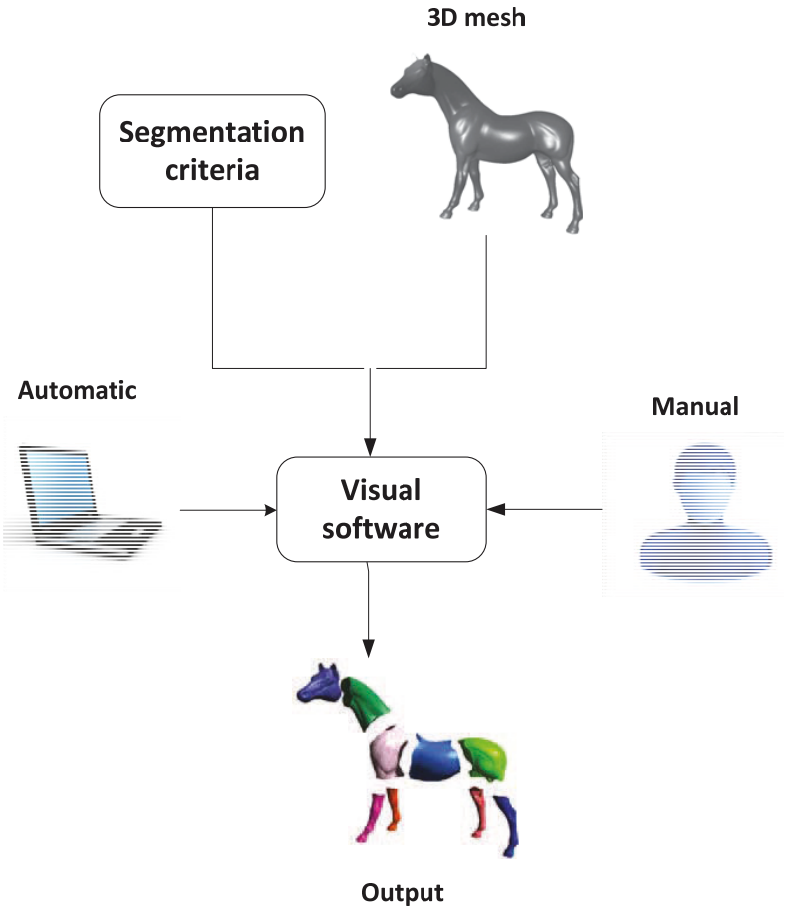


Figure 2.1: 3D mesh segmentation flow diagram.

totally autonomous. Most of the research contributions target the semi-automatic approaches. This is due to their dependency on tuning and object-dependent parameters that make the intervention of the users inevitable.

In the patch-type or surface type 3D mesh segmentation, the objective is to partition the surface mesh into patches under criteria such as, planarity or size of convexity. This type can be used in many applications such as texture mapping [20], remeshing [21], simplification [22], compression [23], and morphing [24].

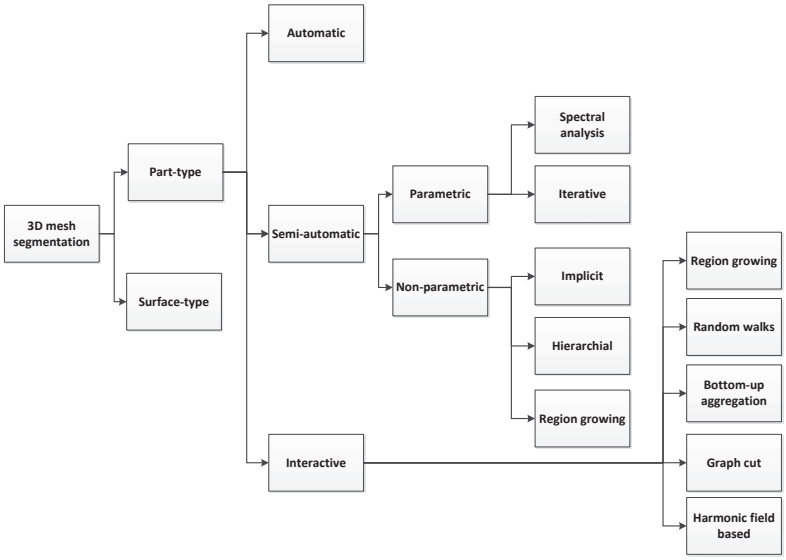


Figure 2.2: 3D mesh segmentation high level classification.

On the other hand, the part-type 3D mesh segmentation aims to segment the object represented by the mesh into meaningful parts or components. Some of the applications that are based on that segmentation type are: shape matching [25], shape reconstruction [26], object skeleton creation [27], collision detection [28], and animation [29]. Figure 2.3 shows examples of patch-type and part-type segmentation techniques.

There is no single technique yet, which is more suitable for part-type or patch-type techniques. The decision on which algorithm is to be used has a significant effect on the segmentation results and is strongly linked to the desired goals of the mesh segmentation process. The next section surveys the literature of the 3D mesh segmentation methodologies.

Figure 2.3: Part 3D mesh segmentation vs. surface 3D mesh segmentation [3].

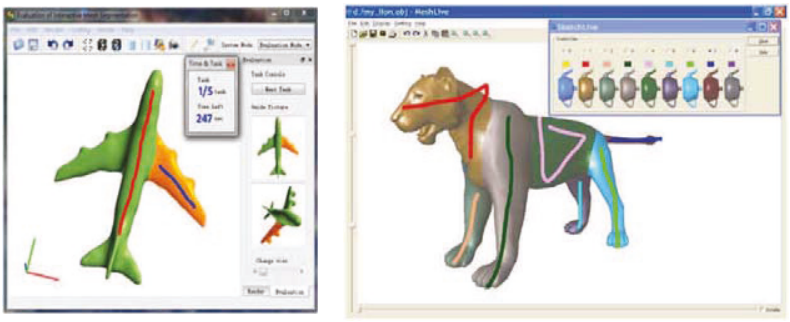
2.2 Related work

The literature of the 3D mesh segmentation problem is relatively small when compared to other domains such as 2D image segmentation. However, the interest in the problem is growing as the 3D media availability is increasing. There are not many commercial applications yet that utilize the advances in the literature but are expected very soon to complement the advances in computer vision. In this section the related works are surveyed and classified according to the level of user intervention in the process. Other surveys that have different classification approaches can be found in [3, 30, 31, 32, 33, 34]. The proposed approach in this thesis aims to compete against the last class of fully automated methodologies.

2.2.1 Interactive approaches

This class of algorithms contains the manual or highly supervised techniques. The challenge in this class is to minimize the user interactions and provide accurate

results that meet the user requirements. As shown in figure 2.4, the interactive 3D segmentation programs expect the user to draw some lines or click in certain areas to detect cutlines and segment the underlying 3D mesh.



(a) (b)

Figure 2.4: Example of interactive 3D mesh segmentation applications [30].

A recent survey by Meng *et al.* [30] inspects the works in this class of algorithms. The survey also, evaluated the literature algorithms and created a ground-truth segmentation data-set. The authors classified the approaches to the following:

- **Region growing:** The algorithms in this class are greedy. The start is a seed where the neighborhood is tested for inclusions and the algorithm stops when there are no more neighbors to include or a metric requirement is met. The seeds are provided by the user. Ji *et al.* [35] used an improved feature-aware isophotic metric while Wu *et al.* [36] used two dimensionless feature sensitive metrics. The advantages of this approach include accuracy, the small number of required seeds and the tolerance to noise. The disadvantages include that it is time consuming and the possibility of over-segmentation.

- **Random Walks:** In this category the computations use the probability value computed by minimizing a Dirichlet energy [37]. The user provides a number of triangles as seeds where the number of seeds is equal to the number of

the desired parts. For each other triangle the probability is calculated for a random walk to arrive to the seed triangle. An example equation is used in [38]

$$P^l(f_k) = \sum_{i=1}^{3} P_{k,i} P^l(f_{k,i}) \tag{2.2.1}$$

where $f_{k,i}$ are the neighboring triangles of f_k and $P^l(f_k)$ is the probability of the random walk.

- **Bottom-up aggregation:** This approach starts with the introduction of multi-scale geometric similarity measure between neighbor mesh elements. The mesh elements are then iteratively aggregated with an adaptive process to reduce the computational cost. The aggregation process is based on the statistics of curvature to recognize consistent geometry. Xiao *et al.* [39] implemented the random walks using curvature statistics while Papaleo and De Floriani [40] a semantic-based approach. The user input here is the selection of features or annotations to aggregate the base on.

- **Graph-cut:** This is one of the common approaches in computer graphics where a minimum graph cut determines the optimal part boundaries. An example cost function $E(A)$ is

$$E(A) = \lambda \sum_{p \in \rho} R_p(A_p) + \sum_{\{p,q\} \in N} B_{p,q}(A_p, A_q) \tag{2.2.2}$$

where $R_p(.)$ is the penalty cost, $B_{p,q}(.,.)$ is the sum of the costs of edges along the partition boundary and $A = (A_1, ..., A_p, ..., A_{|\rho|})$ is the binary partitioning vector. The optimal minimization of this function is obtained by using a min-cut/max-flow algorithm. Related works in this area are the papers done by

Funkhouser *et al.* [41], Brown *et al.* [42], and Fan *et al.* [43]. They all share similar methodlogies but differ in the expected input from the user and in how interactive the developed software is.

- **Harmonic field based:** The user input here is a foreground seed set U and another background one V. The goal is to solve the Poisson equation

$$\Delta\Phi = 0 \tag{2.2.3}$$

with boundary constraints $\Phi(\chi) = 1, \chi \in U, \Phi(\chi) = 0, and \chi \in V$. The harmonic field can be viewed as a smooth interpolation between the constraints. Methods here have more discriminative power for cutting out protruding semantic parts. However, there are limitations in processing non-manifold mesh surfaces [44]. Zheng and Tai [45], Lefohn [46], and Zheng *et al.* [47] are examples of algorithms in this area.

2.2.2 Semi-automatic approaches

The second category of 3D mesh segmentation approaches is the semi-automatic ones. This is the most common class of approaches and contains the highest percentage of the literature research. Some of the used techniques in interactive approaches are used in this class as well. However, the difference here is in the expected input from the user. The user here is expected to provide tuning parameter values and not to touch the 3D object directly or sketch on the 3D mesh. Semi-automatic approaches are sometimes considered to be fully automatic if certain domains are considered such as CAD models or certain quality is satisfactory such as in [48]. The criteria that will be used here to classify the approach to be semi-automatic if:

1. it has parameters other than the 3D mesh;

2. these parameters are not related with the segmentation method. An example of this is the relation between the minima rule and the angle threshold to be used as shown in figure 2.5; and

3. these parameters are related with the 3D mesh. So their values are dependent on the input 3D mesh and hence will definitely need a user intervention.

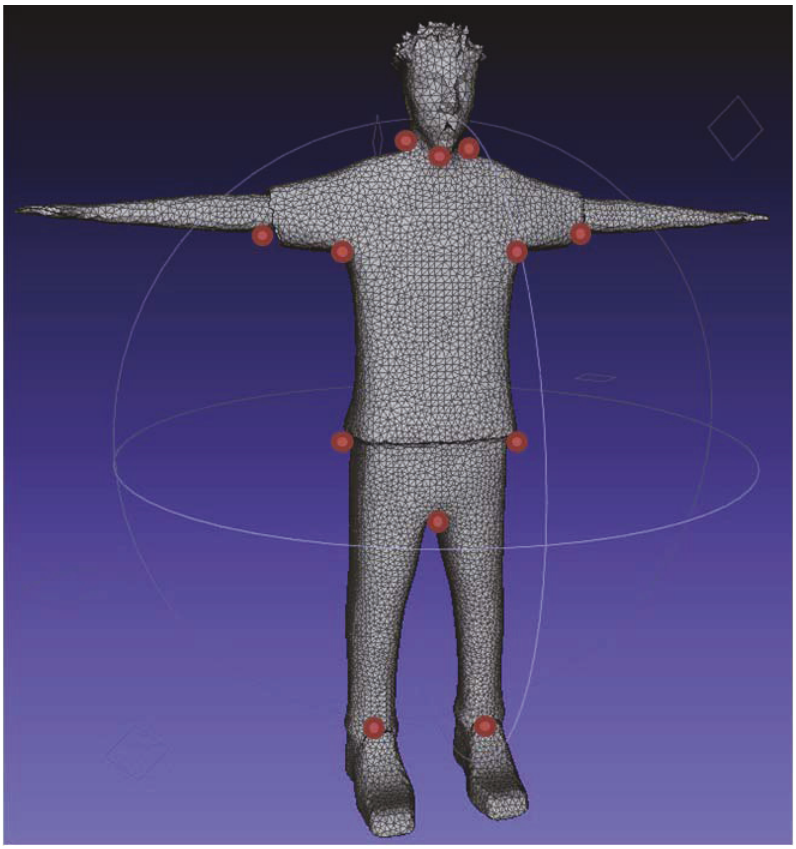


Figure 2.5: The minima rule defines the seeds of cutlines.

The approaches here can be classified from the required parameters point of view, as shown in the figure 2.2. As this category has relatively more literature than others, it can be classified on two basis stages as follows:

- **Non-parametric:** In this technique the algorithm stops based on a quality criteria. The quality criteria is application dependent and can be controlled by the user. The main drawback of this approach is that it over-segments the underlying mesh thus, merging techniques are usually required in the post processing phase. Under this category there are three different techniques:

  - **Region growing:** The greedy region growing algorithm is also used in semi automatic approaches. The number of desired clusters is unknown and the region start seeds are selected randomly. This has the advantage of being simple. However, a disadvantage is its dependency on the initial seeds' locations. Pure random selection of the seeds location can generate bad segmentation results.

    The algorithms that use a region growing approach differ mainly in the judgement criterion that make the regions grow. Kalvin and Taylor [49] approximates a cluster to an ellipsoid. Lavoué *et al.* [50] used an algorithm that is based on the curvature tensor field analysis but applied it mainly on Computer aided design (CAD) models. Sheffer [51] introduced a modeling system called *Shuffler* that uses region growing patches based on convexity and compactness in its initial stage. Zhang *et al.* [52] also employed a region growing scheme in the second phase of their approach, which is based on the vertices curvature.

  - **Multiple source region growing:** Multi source region growing is a common variation of the region growing approach. Here, the seeds start from different locations and grow simultaneously in parallel. The difference between the two can be seen in figure 2.6.

    Example algorithms of this approach include texture atlas generation

Region growing	Multiple source region growing
1. Initialize a priority queue Q of elements Loop 2. until all elements are clustered a. Choose a seed element and insert to Q b. Create a cluster C from seed c. Loop until Q is empty i. Get the next element s from Q ii. If s can be clustered into C 1. Cluster s into C 2. Insert s neighbours to Q 3. Merge small clusters into neighbouring ones	1. Initialize a priority queue Q of pairs Choose **a set** of seed elements $\{s_i\}$ 2. Create a cluster C_i from each seed s_i 3. Insert the pairs $< s_i , C_i >$ to Q 4. Loop until until Q is empty a. Get the next pair $< s_k , C_k >$ from Q b. If s_k is not clustered already and s_k can be clustered into C_k i. Cluster s_k into C_k ii. For all un-clustered neighbours s_i of s_k 1. insert $< s_i , C_k >$ to Q 5. Merge small clusters into neighbouring ones

Figure 2.6: Region growing vs. multiple source region growing [3].

[53]. The texture atlas algorithm first extracts feature contours and use them to define region boundaries. Another example is watershed region growing [54], which has multiple variations. The seeds here are located using a definition of a height function where the algorithm finds all local minima of that function. The variations are due to differences in the definition of feature energy or the height function, where the water goes up inside. Zhou and Huang [55] used Accurate geodesic distance (AGD) that is calculated based on Dijkstra's shortest path [56] method. Wu and Levine [57] used electrical charge distributions over the mesh where the charge density corresponds relatively to the sharp convexities and concavities. Sun *et al.* [58] used the normal variations within a neighborhood of a vertex. Page *et al.* [59] defined a height function between adjacent vertices using Euler's formula:

$$f_{uv} = k_{max} \cos^2 \theta + k_{min} \sin^2 \theta \tag{2.2.4}$$

where k_{max} and k_{min} are the maximum and minimum curvature. θ is the angle between the maximum principal direction and the vector connecting u and v in the u tangent plane.

Multiple region growing shares the drawbacks of the region growing

where the seeds initial locations have a large effect on the results. However, sometimes region growing is mixed with iterative approaches and the seed locations are redefined in every iteration.

– **Hierarchical:** Hierarchical approach for 3D mesh segmentation can be viewed as the bottom up construction of a tree. This is again a greedy approach similar to region growing. However, it can be seen as global greedy because it considers all clusters before taking a merging decision. Different techniques exist within the hierarchical approach and the difference between them is again the merging criteria. A drawback of the hierarchial mesh segmentation approach is that discontinuities may appear in the results.

  Examples of the hierarchical approach in 3D mesh segmentation includes the work of Sheffer [60] where the mean squared distance of a patch to the best fitted plane is used. Gelfand and Guibas [61] used slippage similarity scoring to merge vertices. Attene *et al.* [48] generalized the fitting to more primitives such as planes, spheres, and cylinders. The cost of merging where calculated was based on the fitting errors against all possible primitives.

– **Implicit:** The implicit approach is different than the other non-parametric approaches. It defines the boundaries between the clusters rather than defining the clusters themselves and hence they are implicitly defined. Two main techniques can be found under this category:

  * Top down approach: This is an opposite approach to the hierarchical approach. The start here is with the whole object as the root of the tree and then it is partitioned to smaller parts. The algorithm uses a stopping condition for partitioning such as a certain

tolerance part level that is met or the number of levels of parts is reached. Graph cut can be used here as a post processing step for border smoothing [62].

Katz and Tal [63] used geodesic distances and convexity. They defined a hybrid algorithm that uses iterative clustering and graph cut. Lien *et al.* [64] used the same approach for segmentation and skeletonisation where the tolerance threshold was defined through the quality of the approximated skeleton and a concavity measure of the mesh.

* Inferring: Inferring start by extracting the mesh skeleton, which is a 1-D representation of the mesh. This makes use of the close relation between segmentation and skeletonisation processes. Li *et al.* [65] used a plane perpendicular to the skeleton branches to identify critical points. Raab *et al.* [66] used bead-like primitives by first extracting a voxelized skeleton.

  Other approaches also exist that are based on image segmentation. Gu *et al.* [67, 68] used geometry images to segment 3D meshes.

- **Parametric:** In this technique the user must determine the number of parts beforehand. Otherwise a meta-algorithm should be used to infer that piece of information. This is a strong constraint on the algorithms of this category as such information of the number of parts are not easily deducted without human existence. Two main approaches that exist under this section:

  - **Iterative:** This approach solves the problem by iteratively searching for the best segmentation for a given number of desired clusters. K-means [69] algorithm is considered to be the main technique used here.

It starts with start k representatives, which represent k clusters and they are recalculated on every iteration. The algorithm stops when there is no change in assignment of the representatives. An important issue with the iterative methodology is its convergence. This is dependent on the choice and calculation of the representatives.

Shlafman *et al.* [1] proposed a k-means based face clustering algorithm and used the following equation to define the distance between two faces (f_1, f_2):

$$PhysDist(f_1, f_2) = (1-\delta)\cos^2(\alpha) + \delta \textbf{PhysDist}(f_1, f_2) \qquad (2.2.5)$$

where α is the dihedral angle between the face and **PhysDist** function is the geodesic distance. Wu and Kobbelt [70] used primitives such as planes, spheres and cylinders to create planar shape proxies. Julius *et al.* [71] used quasi developable patches as proxises such as unions of uniaxial conics. Shatz *et al.* [72] also approximated the 3D meshes by a developable surface in order to extract the analytical boundaries between the approximations.

– **Spectral analysis:** This is based on the spectral graph theory [73]. The graph partitioning problem is reduced to the geometric space-partitioning one. Spectral analysis has the advantage of combining segmentation and smoothing in one process. An important tool used in this category is the graph Laplacian, which can be defined as the matrix L:

$$L = D - A \qquad (2.2.6)$$

where A is the adjacency matrix of the graph and D is the diagonal matrix. Examples of research in this category include the work by Liu and Zhang [74, 75] where a sub mesh embedded in 3D is spectrally projected and a contour is then extracted. Another example is the work by Zhang *et al.* [76], which uses the variation within a segment using eigenvectors of a dual Laplacian matrix whose weights are related to the dihedral angle between adjacent triangles and a regularisation term measuring the length of the boundary between segments.

2.2.3 Automatic approaches

Fully automatic approaches operate equally on all types of input meshes. The user of these approaches does not need to tune any parameters based on the input. The allowed parameters are the ones that control the results quality in general and are not mesh dependent. This section will list these approaches, the used parameters, and their relationships with the input meshes.

An example algorithm in this category is the one using a shape diameter function (SDF) by Shapira *et al.* [7]. The SDF-based algorithm uses volume information collected from the mesh to perform the 3D segmentation process. The SDF is a scalar function defined over the mesh surface and measure the diameter of the mesh in local neighborhoods. A point SDF is the weighted average of all ray lengths, which falls within one standard deviation from the median of all lengths. Figure 2.7 shows an example of the SDF method calculation for 3D hand object. Mathematically speaking, the normalized SDF for a face is calculated as follows:

$$nsdf(f) = \log(\frac{sdf(f) - min(sdf)}{max(sdf) - min(sdf)} * \alpha + 1) / \log(\alpha + 1) \tag{2.2.7}$$

where $sdf : F \rightarrow \mathbb{R}$ is the sdf function and α is a normalizing parameter. The segmentation algorithm is composed of two steps: soft-clustering of the mesh faces to k clusters based on their SDF values, and k-way graph-cut to include local mesh geometric properties.

Figure 2.7: Shape diameter function calculation.

The SDF-based algorithm uses the following parameters in its calculations:

- **Cone angle:** The used default opening angle for the cone is 120°.
- **Number of rays:** The used default in the paper is 30.
- **normalized SDF of face α:** The used default value is 4.
- **Partitioning candidates:** The recommended default value by the authors is 5 [6].

The advantage of these parameters is that they can be fixed and that they are not related to the input mesh. However, there are two comments on the above

parameters. The first is that the default values are determined based on the authors experience and not on a mathematical model. The second comment is that the parameters are not orthogonal on each other and it is difficult to define relationships between them.

In general, there are not many 3D mesh segmentation approaches that are qualified to be fully automated. Thus, it is important to develop new algorithms and also improve understanding of the human approach of 3D mesh segmentation to have an efficient simulation.

2.3 Critical review of 3D mesh segmentation techniques literature

The literature review in the previous section and the provided classification leads to some conclusions:

1. the 3D mesh segmentation problem is difficult as it requires the capturing of human perception of objects and their sub parts.

2. the contribution in fully automated 3D mesh segmentation is currently limited and still needs further investigation.

3. other contributions are also required for improvements in areas such as evaluation metrics of 3D mesh segmentation, real-time processing, and new object digital representations that are more supportive for the segmentation process.

The current literature of 3D mesh segmentation has many limitations. The most important ones are being dependent on parameters that vary based on the underlying mesh. Other problems exist such as stability in performance for different classes of

3D objects (e.g. articulated models, CAD models, and others) and real-time speed. Many algorithms tried to improve reliability, such as being pose invariant and local noise resistant. However, some of these improvements still need to match the human perception because this is the ultimate goal of the field.

Another complimentary aspect of the 3D segmentation field is the evaluation metrics and benchmark databases. In recent years, many contributions have been made in this area such as the ground truth databases [6, 33, 77]. The databases were built using large amounts of classified 3D mesh and ground truth cutlines were collected intelligently using human effort internationally through tools such as Amazon's Mechanical Turk [6].

The main existing literature metrics that enable segmentation algorithms to be compared against ground truth segmentations are as follows [33]:

1. **Cut Discrepancy Index (CDI):** This metric was proposed by Chen *et al.* [6]. It aims to measure the distance between cutlines or clustered regions boundaries. This can be calculated as follows:

$$CDI(S_1,S_2) = \frac{DCD(S_1 \Rightarrow S_2) + DCD(S_2 \Rightarrow S_1)}{avgRadius} \tag{2.3.1}$$

where S_1 and S_2 are two segmentations of the mesh. *avgRadius* is the average Euclidean distancefrom a point on the surface to the centroid of the 3D mesh. *DCD* is a directional function defined as

$$DCD(S_1 \Rightarrow S_2) = mean\{d_G(p_1,C_2), \forall p_1 \in C_1\} \tag{2.3.2}$$

and the geodesic distance from a point $p_1 \in C_1$ to a set of cuts C_2 is

$$d_G(p_1,C_2) = \min\{d_G(p_1,p_2), \forall p_2 \in C_2\} \tag{2.3.3}$$

where C_1 and C_2 are the point on the segment boundaries of S_1 and S_2 respectively. The perfect match between segments happen when the CDI value is equal to zero.

2. **Hamming Distance Index (HDI):** HDI is another metric, proposed by Chen *et al.* [6], between segmentations that measure the region difference between their respective segments. The HDI can be calculated using the following equation:

$$HDI(S_1,S_2) = \frac{1}{2}(M_r(S_1,S_2) + F_r(S_1,S_2)) \tag{2.3.4}$$

and

$$M_r(S_1,S_2) = \frac{D_H(S_1 \Rightarrow S_2)}{\|S\|} \tag{2.3.5}$$

$$F_r(S_1,S_2) = \frac{D_H(S_2 \Rightarrow S_1)}{\|S\|} \tag{2.3.6}$$

$$D_H(S_1 \Rightarrow S_2) = \sum_i \|R_2^i R_1^{i_t}\| \tag{2.3.7}$$

$$i_t = argmax_k \|R_2^i \cap R_1^k\| \tag{2.3.8}$$

where S_1 and S_2 are two segmentations of the mesh. The operator represents the set differencing and R_k^i. D_H is the directional Hamming distance. The lower the HDI the better is the segmentation.

3. **Global Consistency Index(GCI):** This metric was proposed by Benhabiles *et al.* [77] and Chen *et al.* [6]. The GCI metric measures the ratio of the

number of vertices that are not common between the segmentations S_1 and S_2. This can be calculated using the following equation:

$$GCI(S_1,S_2) = \frac{1}{N}\min\{\sum_i L_{3D}(S_1,S_2,\upsilon_i), \sum_i L_{3D}(S_2,S_1,\upsilon_i)\} \tag{2.3.9}$$

and

$$L_{3D}(S_1,S_2,\upsilon_i) = \frac{|R(S_1,\upsilon_i)R(S_2,\upsilon_i)|}{|R(S_1,\upsilon_i)|} \tag{2.3.10}$$

where $R(S,\upsilon_i)$ is the region in segmentation S that contains the vertex υ_i and N is the number of vertices.

4. **Local Consistency Index(LCI):** The LCI is similar to the GCI but uses the following slightly different equation instead to account for locality:

$$LCI(S_1,S_2) = \frac{1}{N}\sum_i \min\{L_{3D}(S_1,S_2,\upsilon_i), L_{3D}(S_2,S_1,\upsilon_i)\} \tag{2.3.11}$$

Both GCI and LCI have a range of values between $[0,1]$. The complete similarity between segmentations is indicated by the value of 0. The opposite is the value of 1, which means maximum deviation.

5. **Overlap Index (OI):** The OI metric was proposed by Berretti *et al.* [78]. It measures the extent to, which two regions R_i and R_j from two segmentations overlap. This can be defines as:

$$OI = \max_j \frac{A(R_i \cap R_j)}{A(R_i)} \tag{2.3.12}$$

where $A(.)$ is the function that calculates a region area. The higher the overlapping the better matching between the two segmentations.

6. **Rand Index (RI):** Another metric proposed by Chen *et al.* [6] is RI. This is based on computing pairwise label relationships. The RI is the ratio of the number of pairs of elements having a compatible label relationship in the two segmentations S_1 and S_2. The RI can be defined as:

$$RI(S_1,S_2) = \frac{1}{\binom{n}{r}} \sum_{i,j,i<j} \mathbf{I}(l^i_{S_1} = l^j_{S_1})(l^i_{S_2} = l^j_{S_2}) + \mathbf{I}(l^i_{S_1} \neq l^j_{S_1})(l^i_{S_2} \neq l^j_{S_2}) \quad (2.3.13)$$

where $\mathbf{I}$ is the identity function, N is the number of vertices, and $l^i_{S_k}$ is the corresponding label of all elements contained in region R_i of segmentation S_k. The two segmentations S_1 and S_2 are identical if RI is equal to 1. The value ranges down to 0, which is the total opposite.

7. **3D Probabilistic Rand Index (3D-PRI):** This was proposed by Benhabiles *et al.* [33] and is inspired by similar work in the 2D-image domain by Unnikrishnan *et al.* [79]. The PRI is similar to RI but with the addition of probabilistic interpretation and the option to compare a segmentation algorithm against multiple ground truth segmentations. The probabilistic metric is defined using the following equation:

$$3DPRI(S_a,\{S_k\}) = \frac{1}{\binom{n}{r}} \sum_{i,j,i<j} e_{ij}p_{ij} + (1-e_{ij})(1-p_{ij}) \quad (2.3.14)$$

and

$$e_{ij} = \mathbf{I}(l^i_{S_a} = l^j_{S_a}) \quad (2.3.15)$$

$$p_{ij} = \frac{1}{K}\sum_k \mathbf{I}(l^i_{S_k} = l^j_{S_k}) \quad (2.3.16)$$

where e_{ij} is the event of two vertices i and j that belong to one segment. p_{ij} is the probability of that event in the ground truth segment S_k. $l^i_{S_k}$ has the same definition as in RI. The range of PRI is $[0,1]$ where 0 represents the worst match and 1 is the perfect match between two segmentations.

A more powerful and discriminative variation of the PRI is the normalized PRI (NPRI). Following the index normalization strategy [79] with respect to its baseline:

$$Normalized\ index = \frac{Index - Expected\ index}{Maximum\ index - Expected\ index} \tag{2.3.17}$$

The NPRI is defined as follows:

$$3DNPRI(S_a) = \frac{3DPRI(S_a,\{S_K\}) - E[3DPRI(S_a,\{S_K\})]}{1 - E[3DPRI(S_a,\{S_K\})]} \tag{2.3.18}$$

Many of the above metrics suffer from problems such as [33]: no degenerative cases, tolerance to refinement, cardinality independence, tolerance to cut boundary imprecision, multiple ground-truth, and meaningful comparison. Thus, more research is required in the area of metrics and evaluation of algorithms in order to improve the quality and rectify the research outcomes.

In this thesis, we propose a new 3D segmentation methodology that is fully automated and does not need mesh dependent parameters. The algorithm also is independent from the used cut criteria. Thus, while using the minima rule for the experiments through the thesis, any other cut criteria can be implemented even if it does not have human perception. In the next section we will summarize the proposed framework for 3D mesh segmentation.

2.4 Proposed framework

The main idea behind the proposed algorithm is to transfer the problem from the 3D domain into the relatively easier to process 2D domain. This is actually what the user of an interactive 3D mesh segmentation does, especially when the object to be segmented does not represent any meaning in the user's mind such as a 3D human or a dog model. An example is shown in figure 2.8 where the cutlines are not easy to draw and differ from person to person due to the model asymmetry. The user in this case try to rotate the 3D mesh around all the cartesian axes (x, y, and z) to get 2D snapshots from all view angles. Once this was done, the user applied perception criteria because knowledge was complete about the 3D mesh.

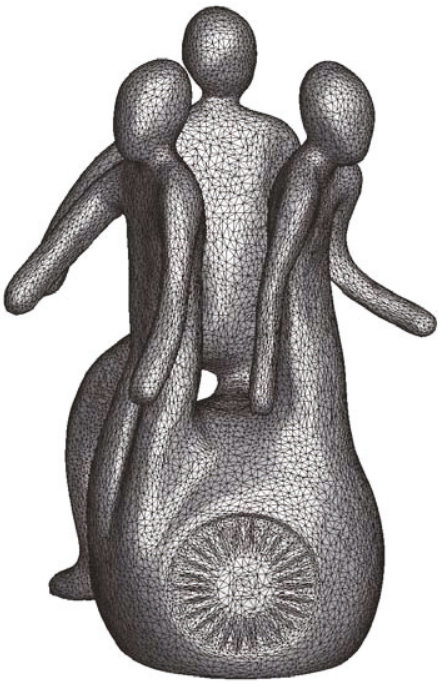


Figure 2.8: A 3D model that does not have obvious cutlines.

The algorithm uses some support tools to assist in producing the correct segmentations. This includes 2D footprint generation, 3D vertex antipodal location

algorithm, graph theory, and other mesh processing operations such as smoothing and re-sampling. The 3D vertex antipodal location module is solving an important problem per se, which has applications in many 3D graphics applications. The problem required novel heuristics to solve the ambiguity in the 3D search. Figure 2.9 shows a high level diagram of the different processes that are involved in the proposed 3D segmentation framework.

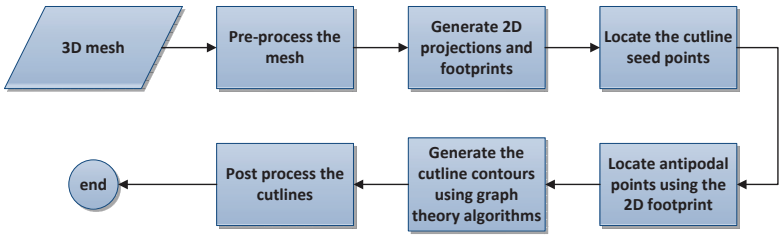


Figure 2.9: A high level flow chart of the proposed 3D segmentation framework.

The modules of the framework such as 2D footprint generation, cutlines seed location, seed antipodal points location, and cutline closed contour generation have the advantage of being ready for parallel processing. Recent advances in general-purpose computing on graphics processing units (GPGPU) [80] and the increase in the number of processing cores in graphic processing units (GPU) [81] comes handy for such processes. The parallelization of these processes is possible as they involve independent smaller sub-routines. They do not even share data and hence guarantee smooth flow of each sub-routine.

3D mesh retrieval [82], which is an important domain in 3D graphics, depends on two important components. They are namely, 3D mesh features and 3D mesh representations. The proposed approach is a good source for both components as the resultant 3D segmentation is easier to express, save, and match.

2.5 Summary

The 3D mesh segmentation is a challenging problem that has many usages in the computer graphics domain. There are many approaches that tackled the problem and they can be classified in terms of amount of user intervention to: manual, semi-automatic, and automatic. The area of automatic 3D mesh segmentation is still in its early stages as the current approaches are still dependent on default parameter values that are deducted using trial and error experiments. New methodologies are required that can relax these dependencies and move further towards fully automatic segmentation. The next chapters will go over the proposed approach and its components.

Chapter 3

Automation of the 3D mesh segmentation process

The proposed approach for the 3D mesh segmentation problem requires a set of supporting tools and algorithms to accomplish its task efficiently. These tools can be used in other problems as well and have generic interfaces for input and output formats. The main tools that are used in this thesis: 2D footprint calculation, 2D and 3D vertex antipodal point location, graph shortest path calculation, and cutlines smoothing for the final steps. This chapter will go over each tool and show how it is incorporated in the 3D mesh segmentation framework.

3.1 Transforming the problem from 3D to 2D

The problem of 3D mesh segmentation can be related to 2D shape segmentation. In fact, what the user can see on a screen is only the 2D projection of the 3D mesh on a certain plane in the space. Every transformation of the mesh, in the form of translation or rotation, produces a new 2D projection. The problem of 3D mesh

segmentation, as will be seen later, is easier when it is transferred to the 2D domain and more data is collected there. This uses the same approach that a user would normally select.

The 2D projection $P = \{p_1, p_2, ..., p_n\}$ of a 3D mesh $M = \{V, F\}$ that has vertices $V = \{v_1, v_2, ..., v_n\}$ can be calculated from V where one of the x, y, or z components is equal to 0. This is of interest for the display of the 3D object. However, a different structure is more useful for the 3D mesh segmentation problem. This is what is defined as the 2D footprint. Several other terminologies are also used in the literature such as outline, shape, hull, or region. The 2D footprint is favored in the literature and is widely acceptable [83].

2D footprint calculation can be defined as the process of assigning a region-like entity to a collection of point-like entities in space [83]. There are many calculation methods in the literature. In this thesis, some examples will be listed that represent different algorithms and calculation strategies.

3.1.1 Footprint calculation algorithms

The surveyed algorithms are: α-shape [84], Dynamic Spatial Approximation Method (DSAM) [85], swinging arm [86], and concave hull [87]. The differences among them exist in three main areas: nature of expected input, the process, and the output.

Edelsbrunner *et al.* [84] demonstrated a generalization of the convex hull algorithm on a set of finite 2D vertices. The proposed algorithm has a computational time of $O(n \log n)$ using $O(n)$ space. They identified an α-hull notion to be:

"The α-hull $(\alpha < 0)$ of S is the complement of the union of all open discs of radius not less than $-\frac{1}{\alpha}$, which contain no point of S"

where the set S contains the n vertices of the 2D plane. The algorithm is dependent on the α values.

Alani *et al.* [85] introduced a method for estimating spatial footprints. The estimation is from the locations of points that lie inside a region and those that do not lie in the region. They called it DSAM and it was based on Voronoi diagrams [88], as shown in figure 3.1. DSAM is also capable of generating fuzzy and historical boundaries to account for changes.

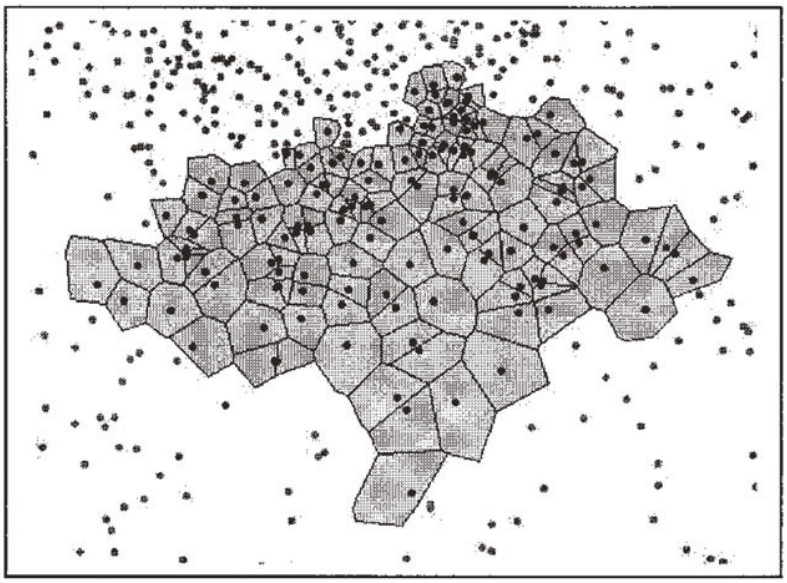


Figure 3.1: Dynamic Spatial Approximation Method (DSAM) for estimating spatial footprints[85].

Galton and Duckham [86] proposed a swinging arm procedure to calculate the 2D footprints. Figure 3.2 shows the steps of constructing a 2D footprint using the swinging arm algorithm. The algorithm works using a sequence of swings of a line that is anchored at an external point to the input points set. The line length r is an input to the algorithm. The algorithm is iterative where the anchor point is changed to guarantee the coverage of all input points.

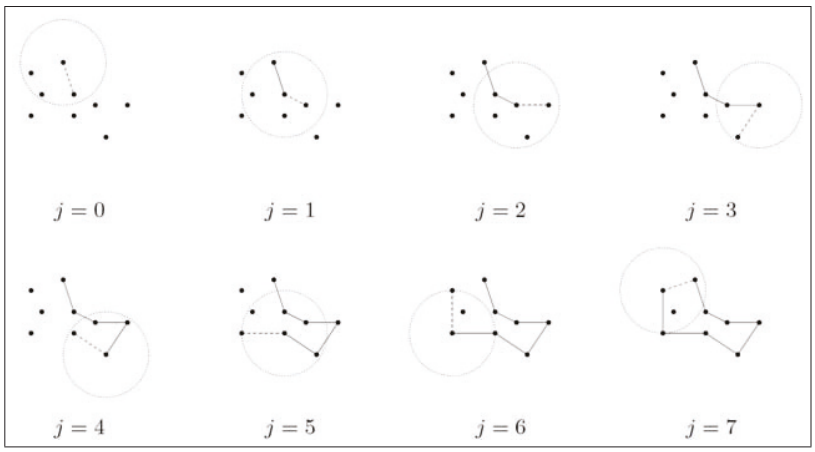


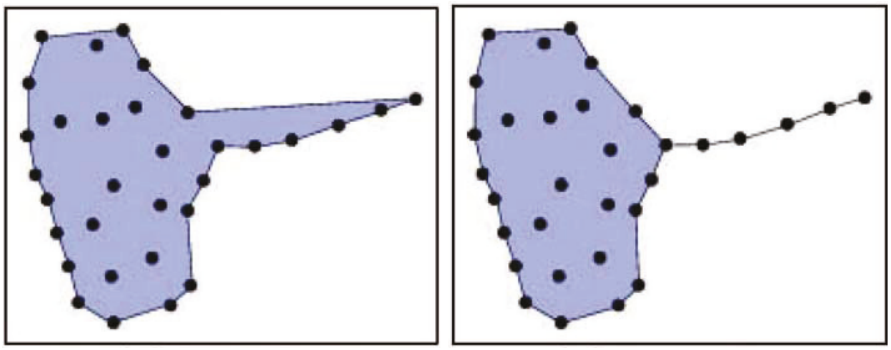
Figure 3.2: Swinging arm algorithm steps [86].

Moreira and Santos [87] introduce an algorithm that is called concave hull. The main idea in this algorithm is that it always selects the next vertex in each iteration from the k nearest neighbors of the current vertex. The algorithm is defined on the value of k, which can be defaulted to 3. This default value can be increased if it is not efficient. The concave hull algorithm requires pre-processing of the input using the Shared Nearest Neighbor (SNN) algorithm [89].

Dupenois and Galton [83] used certain criteria to judge the above algorithms and others. These criteria are classified by the authors into intrinsic criteria that deals with the footprint properties and relational criteria, which is concerned with the relationship between the output footprint and the input dots.

The intrinsic 2D footprint criteria are:

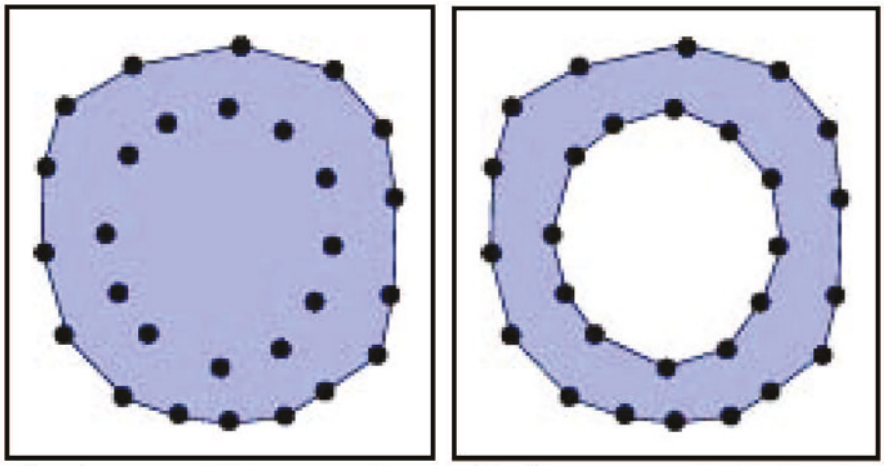
- Connected (C): The output footprint has one single connected component.
- Regular (R): This refers to being topologically regular. Figure 3.3 shows examples of irregular topology and regular ones.



(a) Regular. (b) Irregular.

Figure 3.3: Regular and irregular footprints from topological viewpoint [83].

- Polygonal (P): The footprint is composed of only straight lines and not curves.
- Jordan Components (JC): A Jordan boundary is defined as homeomorphic to a circle, which means that it does not intersect with itself. The footprint is desired to have Jordan components.
- Simply Connected Components (SCC): The one or more components of the footprints need to be simply connected. Figure 3.4 shows an example of non simply connected component, which contains a hole inside the component.



(a) Simply connected. (b) With cavity.

Figure 3.4: Simply connected vs. non simply connected footprints [83].

The relational 2D footprint criteria are:

- Curvature Extrema at Dots (CED): The curvature extrema of all the boundaries is required to coincide with the input points
- All Dots on Boundary (ADB): This is satisfied, as the name indicates, if all the input points lie on the boundary of the footprint.
- No Dots on Boundary (NDB): This is the opposite to ADB where no points lie on the boundary. Both ADB and NDB are extreme cases.
- Full Coverage (FC): This requires that all the points are included within the area covered by the footprint even if they are considered as outliers.

All of these criteria have three values, which are $-1, 0$, and 1. -1 is given where the criteria is never met. 0 is given when some of the outputs meet the criteria while 1 indicates that the criteria is always met. Table 3.1 shows an example of testing some of the literature algorithms against these criteria.

Footprint Examples	C	R	P	JC	SCC	CED	ADB	NDB	FC
α – **shape**	0	-1	1	-1	0	1	0	-1	0
DSAM	1	1	1	0	0	-1	-1	1	1
Swingingarm	0	-1	1	-1	1	1	0	-1	1
Concavehull	1	-1	1	-1	1	1	0	-1	1

Table 3.1: 2D footprint algorithms against classification criteria

3.1.2 2D footprint for 3D mesh segmentation

For the special purpose of the support of the 3D mesh segmentation process, α-shape footprint algorithm has been selected. The selection considered available commercial off the shelf (COTS) software and the fact that most of the available algorithms has a dependency on an input parameter. What helped in making the selection easier is that a high resolution point input is easier for the algorithms to work on and can have fixed input parameters. The high resolution mesh can be achieved by resampling the input mesh. Thus, the value of α can be fixed in the α-shape algorithm.

A package called alphahull [90] that was designed for the R project [91] is used for this purpose. Figure 3.5 shows the package interface and sample script code.

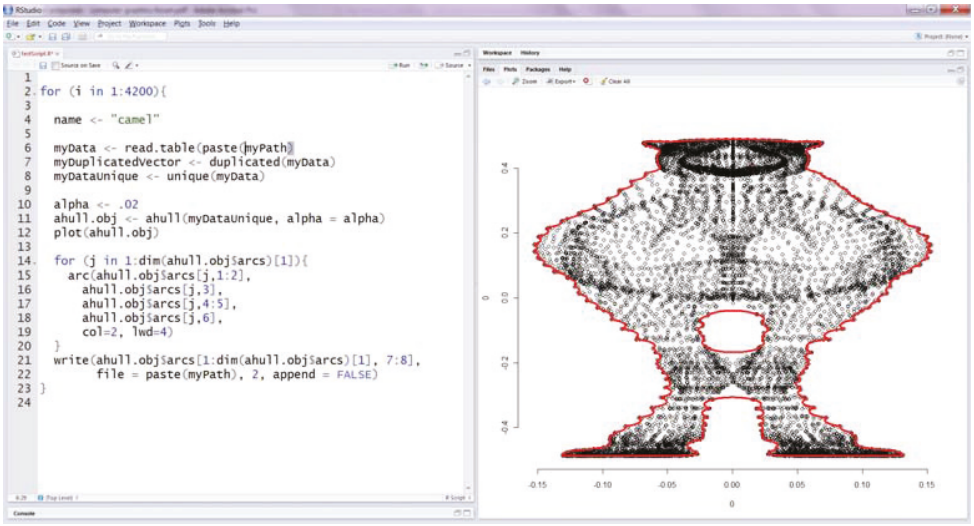
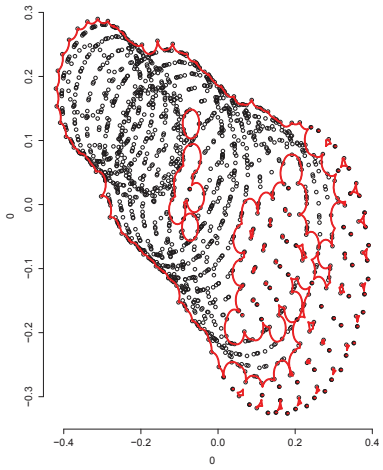


Figure 3.5: alphahull package usage, sample code, and sample output for a camel 3D object.

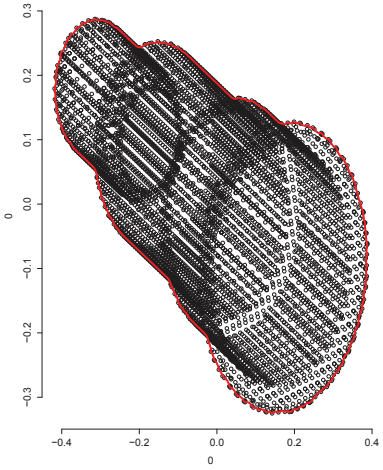
An example of sparse projections such as computer aided design (CAD) models with low resolution is shown in figure 3.6, The remeshing process is utilized to increase the mesh density by splitting triangles into smaller ones with an exit condition of a certain minimum triangle area threshold.

(a) Low resolution 3D mesh.

(b) High resolution 3D mesh.



(c) Corresponding low resolution 2D footprint.



(d) Corresponding high resolution 2D footprint.

Figure 3.6: Example of a different resolution CAD 3D mesh and the corresponding 2D footprint.

3.2 Antipodal location

Determining the location of vertex antipodal pairs in a 3D mesh is a challenging proces. A modified antipodal point definition and calculation methodology is needed for a wide range of graphic rendering applications. A calculation methodology for antipodal points of the 3D mesh vertices, based on the generic definition of being diametrically opposite, is introduced in this thesis. Locating antipodal points is

important for many interactive and automatic computer graphics applications, including mesh scissoring and segmentation. In the case of 3D mesh segmentation, the mesh cutlines are often diametric and require the location of cut points and their antipodal points to form the cutline contour. The proposed methodology completes the search in two phases: 2D search and 3D search to improve coverage and accuracy. To evaluate the proposed algorithm, the distance between the calculated vertex and that observed by the human eye is used as the performance measure. The calculation methods are tested for multiple classes of 3D benchmark objects and an average deviation error value of 0.05%, with respect to the mesh size is achieved.

3.2.1 Introduction to antipodal location techniques

3D meshes are important for many graphics applications due to the vector format's appropriateness for storage and transformations. Applications include topology matching [41], animation [92, 93], and shape morphing [1, 55, 94, 95]. The processing is efficient when more control and understanding is available of the underlying 3D mesh. Understanding here refers to the ability to tag individual elements semantically. One of the areas that improves this understanding is the ability to determine the mesh vertices antipodal points.

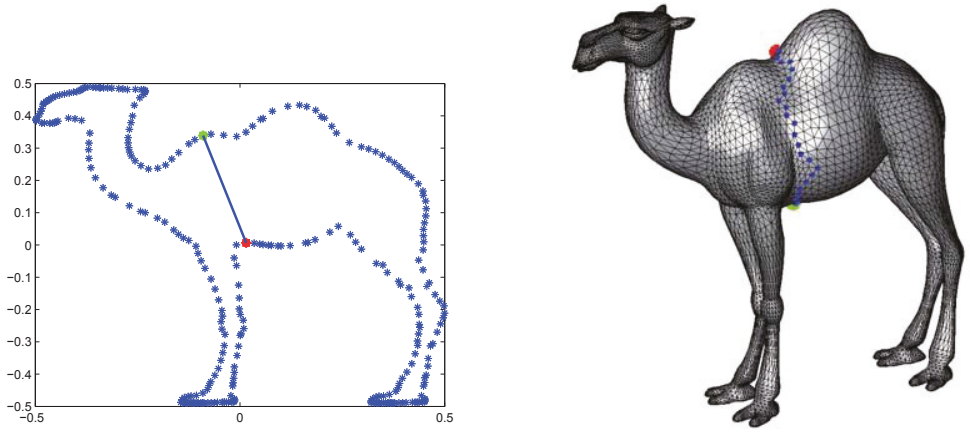
A vertex antipodal point on a sphere is defined as 'diametrically opposite' [96], but alternative variations of this definition exist in specialized domains. In geography [97], an antipodal of a place on Earth is the point on the Earth's surface that is diametrically opposite to it. In mathematics [98], x and y are called antipodal points when they are points on the $n-$ dimensions sphere S^n and $x = -y$. The antipodal definition is also related to the term "antipodal graphs" in [99]. Garry and Karen defined a graph as being an antipodal to another graph G when it:

1. has the same vertex set as G; and

2. is with an edge joining vertices u and v, if the distance between u and v is equal to the diameter of G.

For a simple and closed 3D mesh, which is one of the common mesh representations, the antipodal definition needs to be modified. This is due to the unstructured nature of the mesh as it is difficult to represent the 3D mesh in a function of a few parameters. Cases other than this class of simple and closed 3D mesh are beyond the scope of this thesis because they are not commonly used and may require certain variations of the proposed algorithm to cover discontinuities. Figure 3.7 illustrates the term in 2D and 3D cases according to the Merriam Webster dictionary definition. The antipodal point is intuitive to locate by eye, but automatic techniques find difficulty in doing so [7], especially in the 3D case.

In a boundary 3D mesh $M = \{F, V\}$ composed of a set of faces F and vertices V, a vertex v_i is formed as a shared point between a set of triangular faces $F_v = \{f_1, f_2, f_3, ..., f_n\}$ where every f_i belongs to a plane p_i that might be shared with other faces. This makes it difficult to specify a tangent plane for the vertex v_i and hence hard to locate its opposite vertex or antipodal. Thus, the problem of acquiring a vertex antipodal needs to be transferred from the 3D space to another more suitable space.



(a) 2D case.

(b) 3D case.

Figure 3.7: Example of vertex antipodal.

Jia [100] provided an algorithm to compute all pairs of antipodal points on a simple, closed, and twice continuously differentiable plane curve. To calculate the antipodal points, the author dissected the curve into segments based on being a convex curve or a concave one and then matched them simultaneously on pairs of these segments. This is sufficient for the author's area of application to 2D curves. However, 3D mesh antipodal points location requires several modifications to be addressed in terms of the problem definition, the algorithm design and the pre and post processing of the inputs.

In this thesis, the proposed algorithm is applied using a novel method to compute an antipodal for an input vertex that belongs to a 3D mesh. The main idea is to convert the domain into 2D by acquiring the projections of the 3D mesh in multiple orientations along the three Cartesian axes. A 2D antipodal location algorithm is then applied to compute the output based on a set of search criteria. The advantages of the proposed algorithm are that it:

1. operates on 3D points;

2. does not assume a curve with specific features but deals with a set of 2D points to locate the antipodal points; and

3. is suitable for the computer graphics domain.

3.2.2 Literature review of antipodal location techniques

Vertex antipodal pair mapping is essential when processing 2D and 3D objects. The pair represent the best points from which to grasp an object [101]. They can provide information for the object thickness as demonstrated by Shapira *et al.* [7] using a shape-diameter function (SDF). Furthermore, the mid-point of the line joining the pair belongs to the object skeleton [102]. The object skeleton is important in applications such as mesh animation [103].

As discussed in the introduction, the antipodes definitions and calculation methods are used in many fields including topology, robotics, and mathematical structures. The most frequent approach is to locate the antipodal points for spherical objects. However, in a broader sense they can be generalized to other objects where two points need to be rendered as opposite to each other within a context of some reference [104, 105, 106, 107]. The antipodal location is of great importance in order to use, analyse, and semantically label these objects in a more efficient way.

An example in robotics is where a robot arm must grasp an object with two points. The best positions to grasp, in order to form a force closure, is actually the two ends of a sphere diameter (i.e. antipodal grasp) [101].

In the computer graphics domain, existing research in the literature makes use of the applications of the antipodal locating such as 3D mesh segmentation, 3D mesh skeletonisation, and robot grasping. However, the abstract term of ”vertex

antipodal" definition or calculation method were not thoroughly investigated. In 2D cases, the work of Jia [100] is a cornerstone given that the input is a set of plane curves. However the same concepts are not easy to generalize for higher dimensional objects such as 3D mesh.

3.2.3 Antipodal point definition

It is not as easy to determine an antipodal point of a 3D point in space as it is in the 2D case [7]. Figure 3.8 shows possible 2D cases where the antipodal is the line connecting the points' tangents. The difficulty in the 2D case is differentiating between inward and outward normals to the tangents and choosing in corners, such as in the case of Figure 3.8f. In the 3D domain, the main problem is due to the difficulty of identifying a tangent plane and that in the computer graphics domain where 3D objects are not always symmetric and antipodal point location can be approximate. If an observer is asked to locate the antipodal manually, they will rotate the 3D object before taking a decision on the antipodal point location.

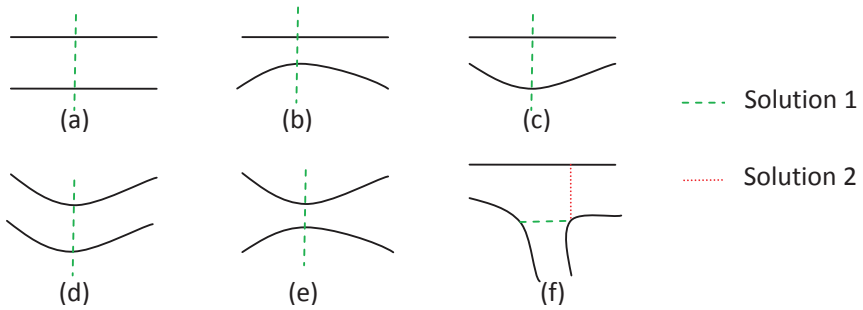


Figure 3.8: 2D antipodal solutions for different curve pairs.

In Jia's paper [108], the author studied the suitable definition of an antipodal

point in the robotic domain and defined over $\alpha(u)$ simple, closed, and twice continuously differentiable curve that it has to satisfy the following conditions

$$N(a)+N(b) = 0 \quad (3.2.1)$$

$$N(a)\times(\alpha(b)-\alpha(a)) = 0 \quad (3.2.2)$$

$$N(a)\cdot(\alpha(b)-\alpha(a)) = 0 \quad (3.2.3)$$

where a and b are two points on the curve α. Thus, the main concern was that their normals are opposite and collinear. However, this might not be the sufficient condition for the 3D mesh case. In this thesis, we proposed a more practical and suitable definition for an antipodal point in the 3D mesh case.

Given a 3D boundary mesh $M = \{F,V\}$ composed of a set of faces F and vertices V, a 2D projection in a certain position is defined as $P_{\theta,\rho,\phi}$ where θ, ρ, and ϕ are angles around x, y, and z axes respectively. The 2D projection $P_{\theta,\rho,\phi}$ is a subset of the mesh vertices V or $P_{\theta,\rho,\phi} \subset V$ where occluded vertices that share same x, y coordinates with others but have lesser z values are not included.

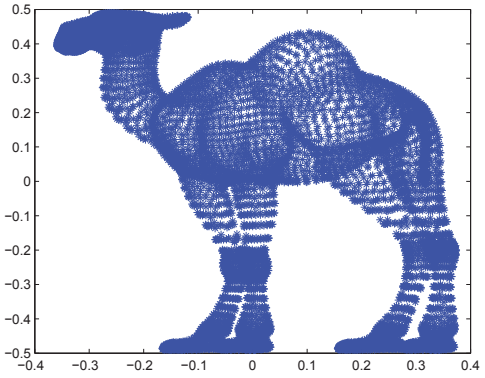
For the sake of antipodal points location, another subset of $P_{\theta,\rho,\phi}$ is defined, which specify only the silhouette or 2D footprint of $P_{\theta,\rho,\phi}$. Thus, $FP_{\theta,\rho,\phi} \subset P_{\theta,\rho,\phi}$ is a set of points that lay only on the borders $P_{\theta,\rho,\phi}$. A recent survey by Dupenois and Galton [83] discuss the recent techniques of identifying $FP_{\theta,\rho,\phi}$. Although the authors specify that a 2D footprint might not be always defined or accurate, it is accurate for 3D mesh specific cases as:

1. the mesh forms a closed and simple curve, which is easy to calculate 2D footprints; and

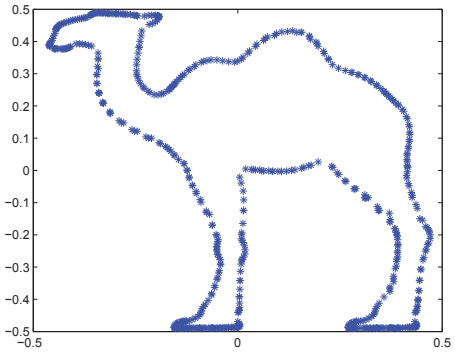
2. if the mesh has a low resolution (i.e. large edge lengths or triangle areas), divisional vertices can be added to increase the mesh resolution.

The 2D footprint $FP_{\theta,\rho,\phi}$ is formed through a subset of points of V. Thus, multiple $FP_{\theta,\rho,\phi}$s for different angle values are required to cover all the points of V. In light of these data structures and formats, the problem of locating a 3D point antipodal is divided into two search problems: a 2D search and a 3D one. Figure 3.9 shows a visualized description of the different terminologies mentioned in this section.

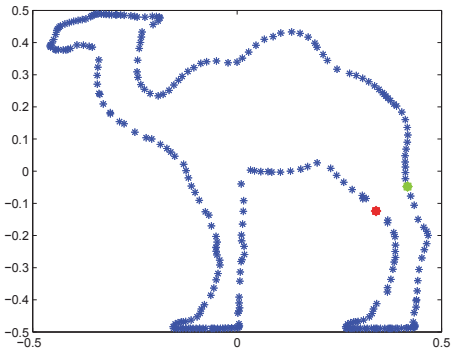
(a) Input for camel 3D mesh.



(b) 2D projection of camel mesh.



(c) 2D footprint of camel mesh.



(d) Camel 2D antipodal pair. Source is red and destination is green.

Figure 3.9: Visualisation of a 3D object at each processing step for a 2D antipodal search.

Computer graphics 2D antipodal

As the antipodal location problem is transformed to a 2D domain, a search for an antipodal point in 2D is performed first. The search is done on the set $FP_{\theta,\rho,\phi}$ that includes the source and destination. It is to be noted that a vertex antipodal pair is not necessarily a commutative pair. Figure 3.10 shows an example in the 2D domain where the normals $\overrightarrow{N_v}$ of the vertex and $\overrightarrow{N_A}$ of the antipodal have an intersection angle greater than 0. The best destination match should satisfy the following

$$max\{N_{in}(a) \times N_{in}(b)\} \tag{3.2.4}$$

and

$$min\{\sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}\} \tag{3.2.5}$$

where $N_{in}(a)$ and $N_{in}(b)$ are inward normals of the vertices a and b, that are defined as normal towards the inside part of the curve formed by the footprint $FP_{\theta,\rho,\phi}$. These are not easy to find unless we apply the point in polygon check [109] that make sure of the normal direction through the odd-even rule, which will be discussed in the next section. Another method could be to neglect the normal direction but avoid neighbors through the minimization of the following

$$min\{\overrightarrow{ab} \times \overrightarrow{ax}\} \tag{3.2.6}$$

where x is a neighbor vertex that belongs to $FP_{\theta,\rho,\phi}$ that is not equal to b, as shown in figure 3.11a. The above search criteria should be satisfied in order to achieve fine results. Figure 3.11 shows the problems in a 2D vertex antipodal search. Mainly, there is the neighboring problem when neglecting the normal directions. The other one is an out-of-boundary problem, which happens due to multiple intersections of the normals.

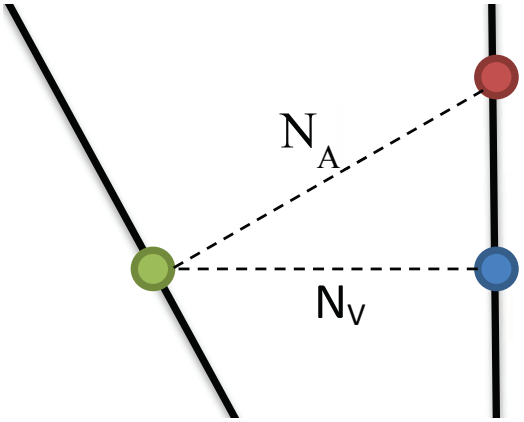
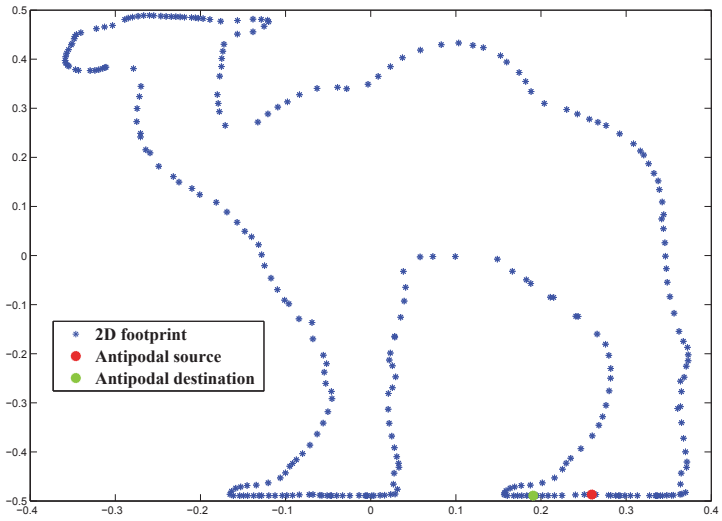
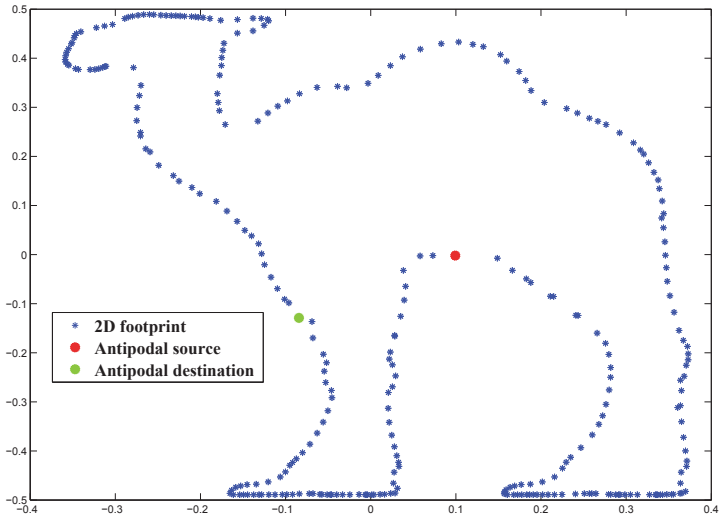


Figure 3.10: A vertex antipodal pair is not necessarily a commutative pair (2D example).



(a) A false positive case of locating a neighbor vertex as an antipodal.



(b) Out of the boundary problem.

Figure 3.11: 2D antipodal location search problems.

Computer graphics 3D antipodal

The 3D domain antipodal point formulation is dependent on the 2D one. However, a 2D antipodal point is not always directly a 3D one. Thus, an antipodal search in the 3D case is a function of all the vertex antipodal points in different projections and its neighbor vertices antipodal points as well. This function could be the minimum, the maximum, the average or any combination of these points. Thus all vertices of antipodal points are required to be calculated before starting the search process.

A proposed function here can be expressed mathematically, defining a vertex a and a set of neighbors $NE = \{v_1, v_2, ..., v_n\}$ where n is the set count. The neighbors can be immediate or up to a certain level of indirect neighborhood. An antipodal point b to vertex a should satisfy the following

$$min\{\sqrt{(Avg_x - b_x)^2 + (Avg_y - b_y)^2 + (Avg_z - b_z)^2}\} \tag{3.2.7}$$

where Avg is the average antipodal point for the set of neighbors NE. This function utilizes the neighborhood information to avoid local errors.

3.2.4 Antipodal point computation methodology

This section is dedicated to the details of calculating the above formulations. The 2D case will be handled first and then the 3D one. A data flow diagram for the whole 2D/3D process is shown in figure 3.12 where boxes represent processes and arrows hold inputs/outputs.

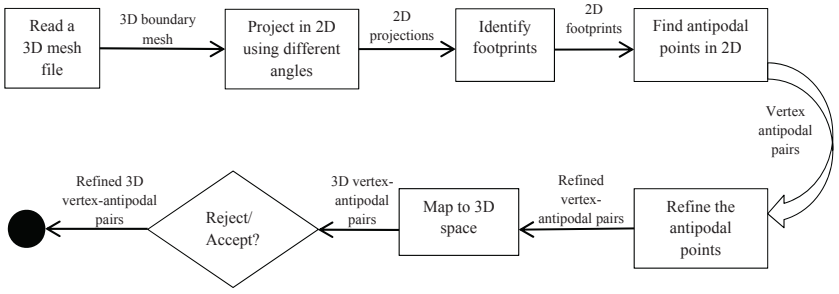


Figure 3.12: Data flow diagram of computing the 2D and 3D antipodal points.

Antipodal 2D point computation

As shown in figure 3.12, the input to the program is a boundary 3D mesh file. The first processing to be done on that mesh is to calculate 2D projections in different poses. The poses are generated by rotating the mesh with different angles around the three Cartesian axes. This produces a set of 2D points but we need to keep the link to the 3D space. Thus, if two points share the same x and y coordinates we only consider the one with higher z value. Secondly, these projections are input to an algorithm that calculates a 2D footprint out of the projected set of points, as stated earlier. Thirdly, the search for 2D antipodal points start. The search is done over the footprint set and the best match should satisfy the objective functions in equations (3.2.4), (3.2.5), and (3.2.6) above. This happens in an exhaustive manner by giving each point of the footprint set a score as follows

$$V_{score} = w_1F_1 + w_2F_2 + w_3F_3 \tag{3.2.8}$$

where F_1, F_2, and F_3 are the objective functions, while w_1, w_2, and w_3 are their weights. The weights are assigned to show priority and the weights polarities define whether it is a minimization or a maximization function.

The search for 2D antipodal cannot always produce all correct antipodal points

at one pass. Thus, here comes the fourth and final step in the 2D search to validate and rectify the search anomalies. The goal of having similar normals is given higher weight. This leaves two main problems, as shown in figure 3.11a and figure 3.11b. The first problem is detecting neighbors as antipodal points because they are close in distance and share the absolute normal values. The problem is solved by re-ranking based on equation (3.2.6) given higher priority. The other problem is having antipodal points in an out of the mesh manner. The problem is solved using point in polygon check [109]. To apply this check, the footprint is transformed into a set of lines and the normal of the vertex that we desire to get an antipodal for is extended from both sides. Then starting from any side we detect pairs. figure 3.13 shows this process for a camel footprint.

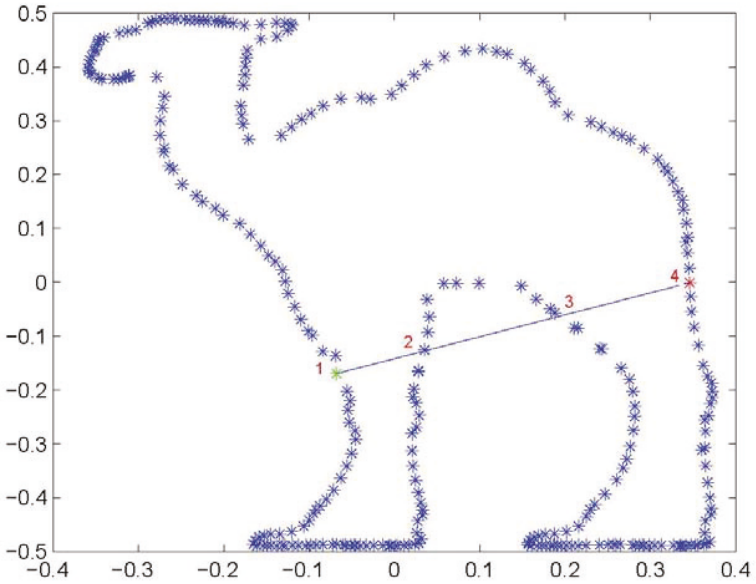


Figure 3.13: Possible solutions to navies' antipodal selection using point in polygon algorithm.

Antipodal 3D point computation

A link to the 3D space is kept through all the steps of the 2D search for the antipodal points to ensure the conversion to 3D space is straightforward. However, the calculated antipodal point is not always accurate, as discussed earlier. A two step mechanism have been proposed to correct this problem. First, an antipodal point acceptance/rejection test. Then a method to calculate new antipodal points for the rejected ones using local information from neighbors is applied.

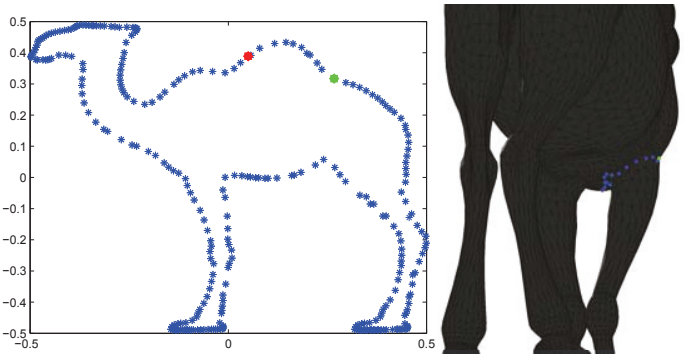
The acceptance/rejection test is repeated in a similar manner to the point in polygon test through having a ray between the vertex and its antipodal. The ray is tested against all the mesh faces (e.g. triangles). If it intersects with more than two faces, then it is rejected. In case of rejection, the algorithm suggests a replacement, which is the matching pair from the point in polygon test. The suggestion is tested against local neighbors' results. If the difference is larger than a certain threshold then an adjustment is added to the point.

3.2.5 Experimental results of the proposed antipodal point location techniques

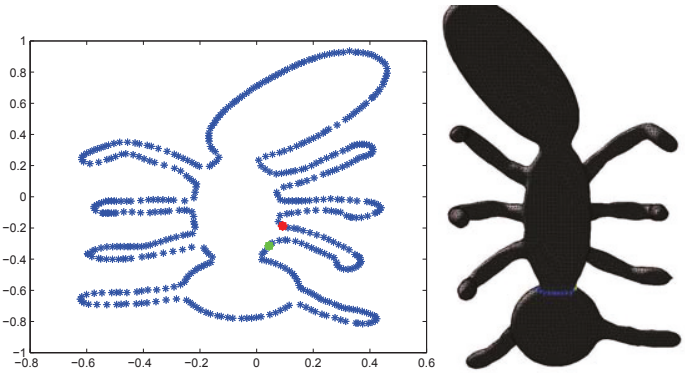
In this, the proposed algorithm performance is assessed in the detection of antipodal points for various classes of 3D mesh. The classes are chosen from the publicly available database by Chen *et al.* [6]. The object categories according to Chen's classification are four-leg, ant, fish, glasses, hand, octopus, and teddy. For testing purposes, we generated 30 random indices of each 3D mesh vertices and checked the performance of the algorithms versus the manual computation by the user.

In Figure 3.14 illustrated selected the 2D and corresponding 3D figures from different classes. The 2D performance measure is the average number of points

that form the shortest path between the manual observed vertex and automatic one determined by the proposed 2D search algorithm. The average is an appropriate measure, as we have more than one projection. It is to be noted that the projections are chosen randomly. A similar measure is applied in the 3D case as well, but without averaging as we only get one 3D point.

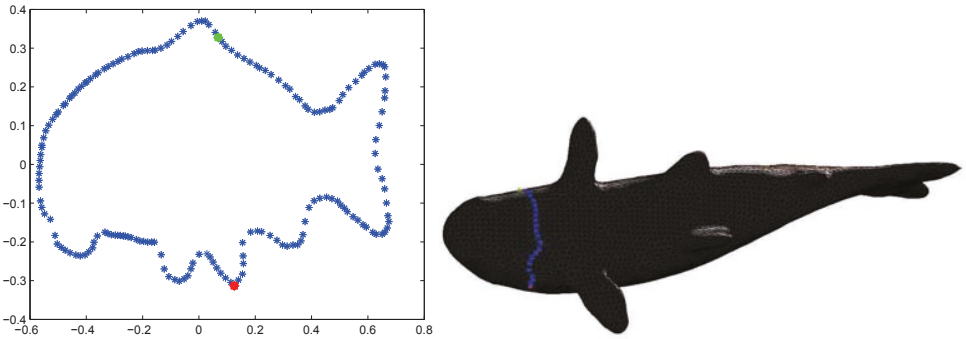


(a) Camel 2D and 3D search.

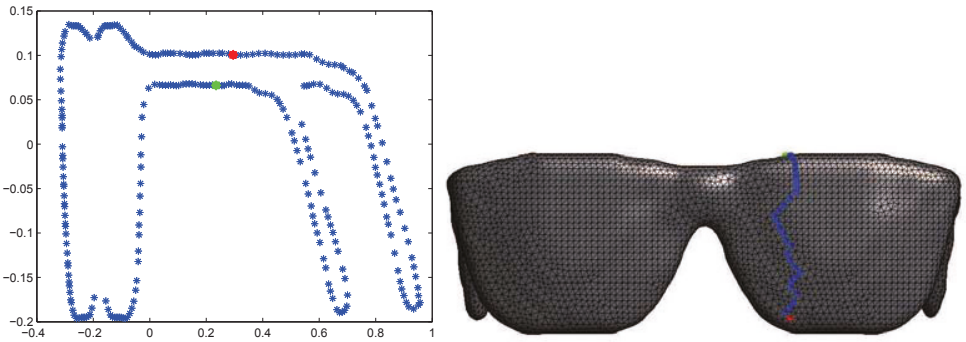


(b) Ant 2D and 3D search.

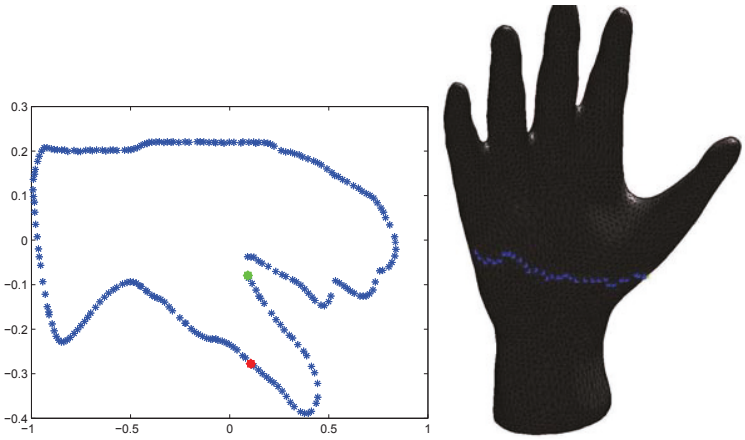
Figure 3.14: Examples of 2D and 3D search results for different classes of objects.



(c) Fish 2D and 3D search.

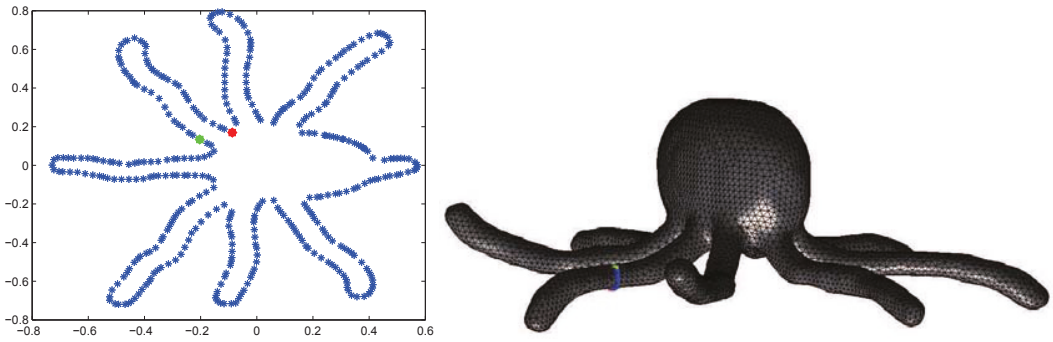


(d) Glasses 2D and 3D search.

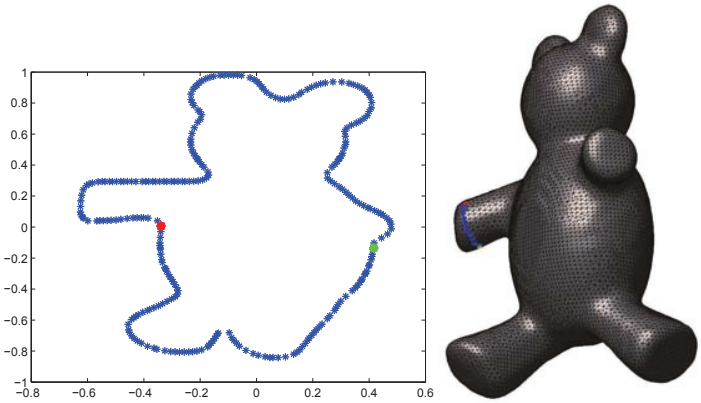


(e) Hand 2D and 3D search.

Figure 3.14: Examples of 2D and 3D search results for different classes of objects.



(f) Octopus 2D and 3D search.



(g) Teddy 2D and 3D search.

Figure 3.14: Examples of 2D and 3D search results for different classes of objects.

Object Name	Vertices count	Avg. 2D perf. dist.	Avg. 3D perf. dist.
Camel	9,770	2.56	4.05
Ant	8,388	1.56	2.45
Fish	6,264	2.04	2.82
Glasses	7,407	1.28	3.83
Hand	6,607	1.32	2.07
Octopus	7,251	1.28	4.69
Teddy	11,090	2.72	6.90
Average	**8,111**	**1.82**	**3.83**

Table 3.2: Results of different classes in 2D and 3D cases

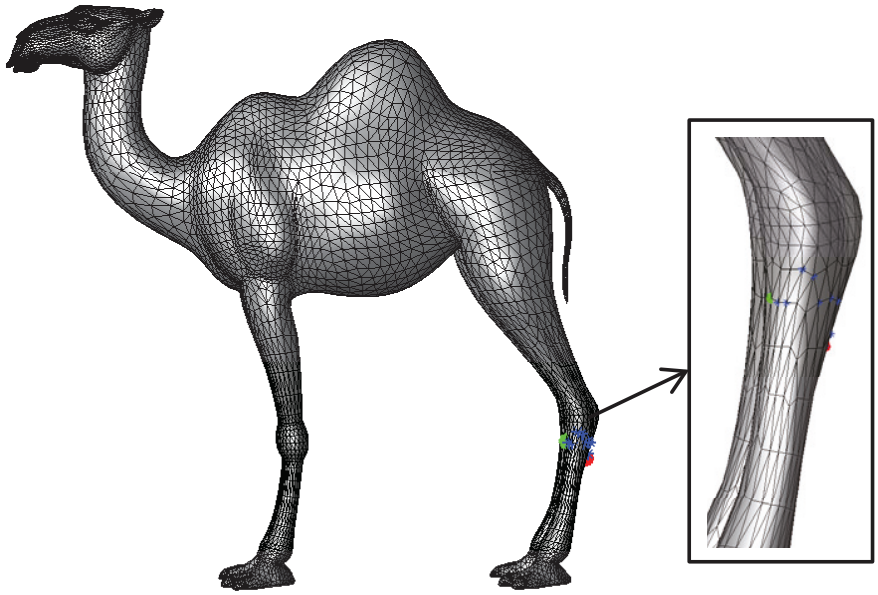


Figure 3.15: The shortest distance between two points is performance measure for the 3D antipodal search.

The shortest distance between two points is used to measure the performance of the search algorithms, as shown in figure 3.15. The results, as shown in table 3.2 and figure 3.16, demonstrate that there is a relation between the mesh size and the performance of the 2D and 3D searches. The 3D search accuracy is dependent on the 2D search accuracy. The numbers are calculated based on 100 points in the

2D case for each 3D object. These 100 points are formed from 10 vertices for 10 different poses. In the 3D case, the numbers are the average of 10 points per object.

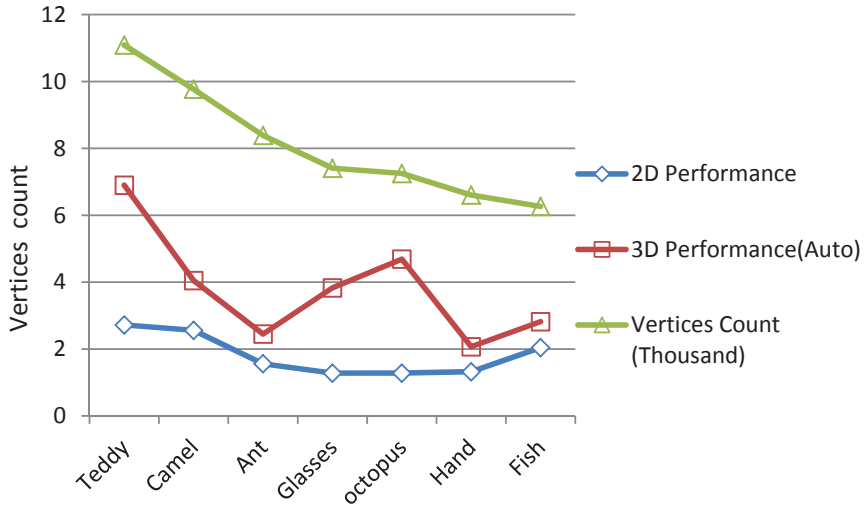


Figure 3.16: The 2D/3D search performance with respect to the mesh vertices count.

We also propose an alternative automatic evaluation approach that utilizes an ellipse fitting algorithm [110]. The algorithm first calculates the shortest path between the vertex and its antipodal vertex. Then, a plane is defined through three points, which are the vertex, its antipodal and the midpoint of the shortest path defined in the previous step. After post processing the intersection of this plane with the mesh vertices, we define the other shortest path between the vertex and its antipodal. This closed contour between the vertex and its antipodal is then fitted to an ellipse. The performance measure is then calculated based on the fitting error of the contour to the ellipse. The advantages of this approach are that it is automated but the disadvantages lie in the fact that a mesh is not always formed of elliptic tubes.

Figure 3.17 shows the performance function that is used in the automatic evaluation. The fitting error is divided on the mesh resolution to form a relative measure.

The results show that the random samples error is small and within the acceptable margin.

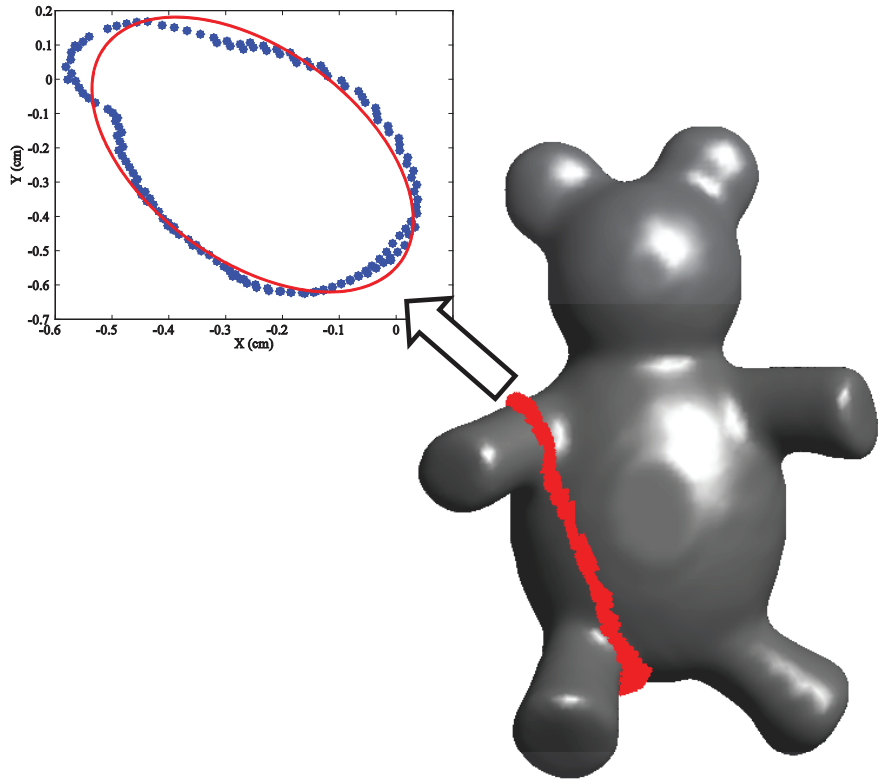


Figure 3.17: The automatic evaluation of the accuracy using ellipse fit algorithm.

The next section will discuss graph theory topics and algorithms, which are necessary and effective for connecting the vertices with their antipodal ones. This is an important step in the proposed mesh segmentation algorithm and its output is the mesh cutlines.

3.3 Connecting two points in a mesh

Once a vertex and its antipodal vertex is identified, the next challenge is to find a contour that is closed, connects the two vertices, and is minimal. The problem here is that the path between the two points has to be directed to go to the other side of the mesh rather than than the natural shortest path. Thus, an algorithm is required to aid in the formation of the closed contour.

3.3.1 Related work of closed contour creation

Several approaches exist in the literature that targeted similar problems. Interactive 3D mesh segmentation methods required the user to input a line and then they completed the line into a closed contour. The computation of the cutting contour used harmonic fields isolines and selected the best isoline based on centerness and curvature [45, 47, 111]. In semi automatic methods, contours are also not generated at one stage. They are usually partially created and then closed in another stage. An example is the work of Lee *et al.* [112] where they suggested a method to close contours that are based on a combination of four functions: Distance function

$$\eta_d(\upsilon) = \sum_{\upsilon_i \in \gamma} \frac{1}{d(\upsilon, \upsilon_i)}, \tag{3.3.1}$$

Normal function

$$\eta_n(\upsilon) = \begin{cases} 1 & \text{if } n_\gamma \cdot n_\upsilon \geqslant \cos(\alpha), \\ \frac{n_\gamma \cdot n_\upsilon + 1}{\cos(\alpha)+1} & \text{otherwise}, \end{cases} \tag{3.3.2}$$

Centricity function

$$\eta_c(\upsilon) = \sum_{\upsilon_i \in \gamma} |c(\upsilon_i) - c(\upsilon)|, \tag{3.3.3}$$

$$c(\upsilon) = \frac{\sum_k w_k \cdot c(\upsilon_k)}{\sum_k w_k}, \tag{3.3.4}$$

and a normalized feature function $\eta_f(\upsilon)$ that was not stated mathematically. γ is the contour and υ is a mesh vertex. n_γ is the center vector of the normal cone of all vertex normals and n_υ is the normal vector of a vertex υ. $w_k = 1/d_k$ is the inverse from the distances d_k from υ to the corner vertices υ_k.

These four functions combined with the length of the edge $l(e)$ and weighting variables w_d, w_n, w_f, and w_c the authors used the cost function:

$$f(e) = l(e) \cdot \eta_d(e)^{w_d} \cdot \eta_n(e)^{w_n} \cdot \eta_f(e)^{w_f} \cdot \eta_c(e)^{w_c} \tag{3.3.5}$$

This method suffers from dependability on the values of parameters w_d, w_n, w_f, and w_c. These parameter values are hard to determine without enough experiment. In this thesis, a simpler automatic method that is not dependent on any parameters, has been provided.

3.3.2 Proposed closed contour formation algorithm

A 3D mesh can be transformed into a graph where vertices will be the nodes and edges will be basically bi-directional and with an equal traversal cost. These can be represented in computers using a sparse matrix to be efficient for processing. Graph theory has many algorithms that help in the processing of the graph data structures. A handy algorithm here is the famous shortest path proposed by Dijkstra [56] and named after him.

For two vertices v_i and v_j that belong to the mesh $\mathscr{M}$, the shortest path between them is defined as $p = \{v_1, v_2, ..., v_n\}$ where n is the count of the vertices in the shortest path. Using the above antipodal algorithm, each vertex can be utilized to

generate an antipodal vertex. Thus we acquire the opposite path to p, which when we connect its ends to p end vertices we can get the closed contour. However, the problem of this algorithm is that it may be affected by errors in the antipodal location method.

An easier and quicker approach is to use curve fitting approaches and specifically ellipse fitting. The generated ellipse out of these algorithms can define the opposite path to p and hence the full closed contour. A Matlab ellipse fitting open source code was utilized here to generate the results. Figure 3.17 shows an example of the method usage.

3.4 Output Smoothing of the 3D mesh segmentation process

The last step in the 3D mesh segmentation processing requires fine tuning of the output cutlines. This can make use of available smoothing algorithms such as an active contour model or snakes [113]. A snake moves by minimizing a function E_{snake}, which can be defined as follows [112]:

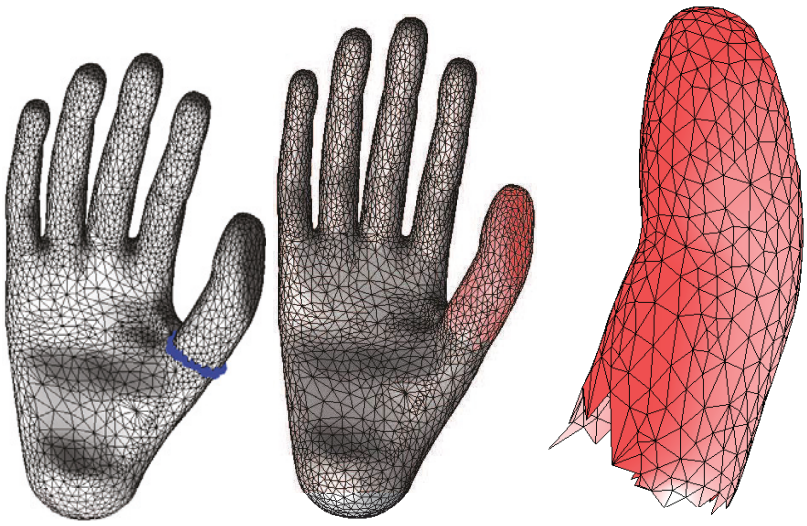
$$E_{snake}(s) = \int (E_{spline}(s) + E_{mesh}(s))dt \tag{3.4.1}$$

Where s is the set of points and the snake is initialized by a sample subset of it. The snake's final output is smoother and can be used as a final cutline. A customized open source Matlab toolbox [114] was used here to automate the smoothing process.

3.4.1 Converting cut lines to 3D parts

A complimentary step is to convert the cutlines to separable individual mesh. This is done using region growing algorithm where neighbor elements such as triangles are added as long as they do not cross the cutline [115]. Although this algorithm is straightforward in application, it suffers from being a memory consuming algorithm. This is due to its recursive nature and that its filling process is done one by one and not in batch mode.

A direct fix for this problem is to keep a global counter of the number of called functions, which stops the recursion once it reaches a certain limit. The process can then iteratively be controlled until there are no more triangles to include. Another approach is to examine neighbors from only one source by including neighbors at a distant of 1 and then 2 and so on. The seed is then moved to the border and the algorithm continued. Figure 3.18 shows an example of a region growing algorithm on a 3D object.



(a) Cutline. (b) Region growing (c) Sub part

Figure 3.18: Converting cutlines into sub parts in 3D mesh.

3.5 Conclusion

In this chapter, the 3D mesh segmentation process supporting methodologies were discussed. These methodologies are the main components of the proposed framework. The input 3D mesh is transformed and processed in each phase to be able to extract the required segmentation data. All the proposed methods are automatic and independent from parameters for the proposed application. Vertex antipodal location was one of the methodologies that is considered as a cornerstone, as it is a powerful methodology for extracting and locating cutlines. Also, refinement methods were analyzed to ensure that the output fitted the user requirements with plausible quality. This chapter showed the methodologies individually. The next chapter will show how these methodologies can be augmented in one framework. It will also show the results of applying them on different 3D object classes and benchmark results against literature databases.

Chapter 4

Automated 3D mesh segmentation framework

The proposed automatic 3D mesh segmentation framework is discussed in this chapter. The framework has several components that an input goes through to produce the desired output. The proposed approach is applied on different classes of 3D mesh. This is then benchmarked against the literature metrics using 3D objects from public databases. The results are competitive against other algorithms, which use object-dependent or tuning parameters. This plus the autonomy and generality features, provides an efficient and usable approach for segmenting 3D mesh.

4.1 Input formats and pre-processing

The expected input to the algorithm is a boundary triangular mesh. This is one of the most common formats for storage and transfer of 3D mesh [116]. Other formats such as voxels, point cloud, or Constructive solid geometry (CSG) can be transformed to the required format. Not all inputs are always ready for processing

and hence some pre-processing is required. For the sake of the proposed algorithm, the input is required to have sufficient dense resolution and to be closed.

The plain format of the input provides limited available data to the algorithm. Basically, only vertices, faces, and (in some file formats) faces normals are available. Thus, more data structures are required to be inferred and calculated. These are mostly regarding the relations between the different components of the mesh. Examples of these data structures are faces neighborhood maps and distance graphs that uses the Dijkstra shortest path algorithm [56]. These structures are supportive to the search process as they provide local and global awareness.

As discussed in the previous chapter, the proposed algorithm uses 2D footprints in one of its stages. Therefore, the denser the mesh vertices, the more quality footprints are gained. Some mesh have a low resolution and hence needs to have it increased to an acceptable level. This is done by re-sampling the mesh. This feature is available in many mesh processing types of software such as Meshlab [117]. The only problem is in relating the resampled mesh vertices with the original ones. Thus, specific routines are written that iteratively divide triangles into smaller ones until a threshold of the triangle area is met. The new vertices are linked to old ones in order to produce an output that is compatible with the original input.

For high noisy mesh, a smoothing preprocess is important. The minima rule criteria locates local curvature discontinuities and hence can over segment for noisy meshes. If no preprocessing is done, then a post processing might be required to augment small parts into a large meaningful one. However, this might need additional information for grouping. The preprocessing smoothing can be done using the Laplacian method [118], which is basically a local averaging process to remove noise. This can be expressed mathematically as

$$\bar{x}_i = \frac{1}{N}\sum_{j=1}^{N} x_j \tag{4.1.1}$$

where N is the number of adjacent vertices to vertex v_i. The new position of v_i is defined by $\bar{x}_i$.

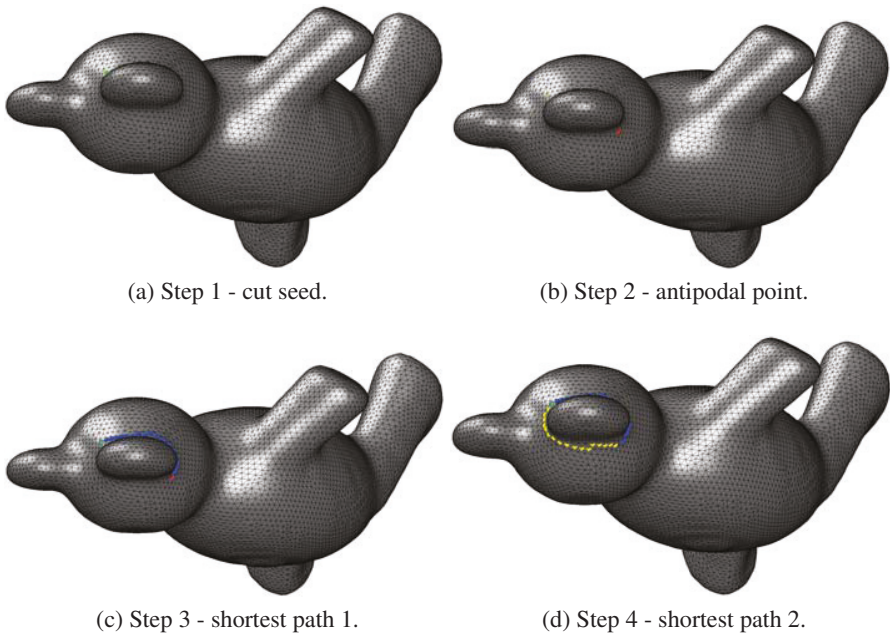
Other optional input to the framework can be the curvature threshold. This controls the definition of cut areas or where the seeds are put for initiating a cutline. The reason that these are optional is that a fixed or default value can always be used irrespective of the input mesh. Such parameter help users in customizing the 3D mesh segmentation process. As discussed in the literature review, different users can have different perspectives when it comes to 3D mesh segmentation and hence the output is not fully deterministic.

4.2 3D mesh segmentation processing

The 3D mesh segmentation can be defined over a boundary mesh $\mathscr{M}$ of $\{\mathscr{V},\mathscr{E}\}$ as the process of splitting it into a set $\mathscr{S}$ of n segments $\{s_1, s_2, ..., s_n\}$ by a criteria $\mathscr{C}$. $\mathscr{V}$ is the mesh vertices set and $\mathscr{E}$ is the edges one. Another output could be a set of cut lines $\mathscr{L}$ of $n-1$ lines $\{l_1, l_2, ..., l_{n-1}\}$ and in that case it is called an implicit approach [3]. This is the category that the proposed algorithm belongs to.

As the output is a set of cut lines or more specifically closed contours we define the components of each line as $l_i = \{P_{start}, P_{end}, Path_1, Path_2\}$. P_{start} is a point of nomination to start a cut such as a point of concavity discontinuation if following the cognitive theory. P_{end} is the antipodal point of P_{start} on the 3D mesh as shown in Figure 4.1b. $Path_1$ is a set of vertices that define the shortest path between P_{start} and P_{start} using a graph theory algorithm such as the popular Dijkstra one [56]. $Path_2$ is also the shortest path between them but after increasing the cost of the edges

included in $Path_1$ and running the shortest path algorithm for the second time as will be shown in implementation section. Figure 4.1 shows the components of the cutline as calculation steps go on.



(a) Step 1 - cut seed. (b) Step 2 - antipodal point.

(c) Step 3 - shortest path 1. (d) Step 4 - shortest path 2.

Figure 4.1: The calculation steps of the 3D mesh cutlines.

The search for these four components of each cut line is required to be as deterministic as possible. While three components, namely the $P_{start}, Path_1$, and $Path_2$ can be deterministic as will be seen later on, the search for P_{end} is harder and requires a set of carefully designed heuristics. P_{end} represents the antipodal point of a vertex. The definition of an antipodal is not always clear in all cases. In 2D space it usually refers to the diametrically opposite vertex. This is hard to calculate in 3D space as the tangent plane to the vertex is not unique. Due to this ambiguity, more information is required to be collected before deciding on, which vertex qualify as an antipodal one in the 3D space. In the next sections a novel algorithm is followed that makes the definition of antipodal means clearer.

4.2.1 3D mesh segmentation framework

Using the previously described algorithms and methodologies, the antipodal point of the seed vertex of the cutline can be located. However, the algorithm needs first to determine this seed vertex. In this implementation we adopt the minima rule in locating the seed point because it is the closest to a human approach [17]. The calculation uses the values of the angles or curvature between the mesh neighbor faces. The diagram, as shown in Figure 4.2, has a dense part and a sparse one. The cutlines seeds lie in the sparse part and the user can have more or less cut lines by considering more or less points respectively.

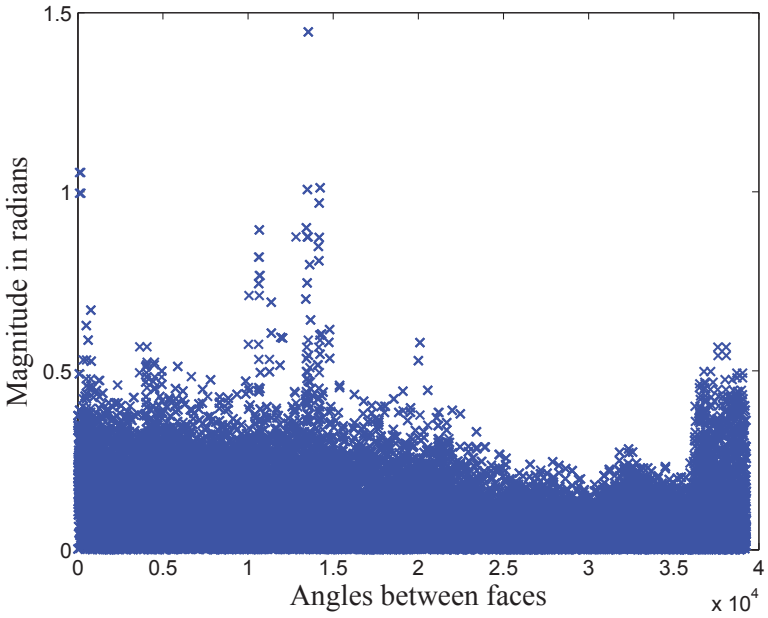


Figure 4.2: The values of angles between the faces of the teddy 3D mesh.

Having the two ends of the cut line is not enough. The contour of the cut line needs to be complete. That is why the next step of the algorithm is to transform the 3D mesh into a graph and apply the shortest path algorithm on the two points. This

produces a half contour. To complete the contour an iterative algorithm can expand one end of the half contour by checking the neighbors of the end points. A point is added to the half contour if it passes a simple test. The test is how close it is to the plane formed by the half contour vertices. Figure 4.3 shows a data flow diagram of the complete proposed 3D mesh segmentation algorithm. The preprocessing refers to changing the resolution of the mesh while post-processing aims to smooth the cutlines and remove the incorrect ones.

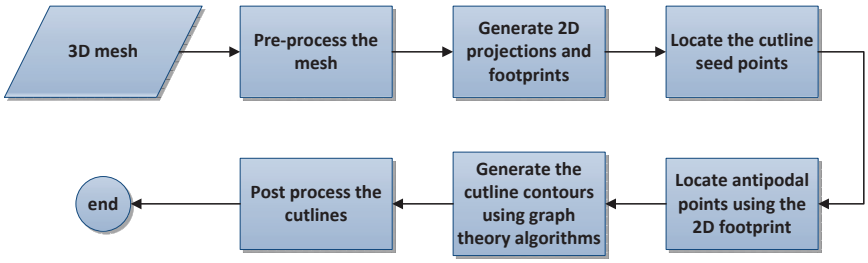


Figure 4.3: Data flow diagram of the 3D segmentation framework.

4.2.2 Complexity analysis of the proposed 3D mesh segmentation algorithms

The algorithm is usually used offline. However, an online use case can be facilitated through parallelization using advances in General Purpose computing on Graphics Processing Units (GPGPU). The complexity analysis of the algorithm gives an estimation of the processing time of its different stages. A commonly used notation for complexity analysis is the big O notation. This notation provides the upper bound on the algorithm growth rate [119]. Other notations also exist such as little o, Ω, ω, and Θ. The big O notation can be mathematically described as:

$$f(x) = O(g(x)) \; as \; x \to \infty \tag{4.2.1}$$

if and only if

$$|f(x)| \leq M|g(x)| \; for \; all \; x > x_0 \tag{4.2.2}$$

where $f(x)$, $g(x)$ are two functions and M is a positive constant. x_0 is a real number.

In order to analyze every stage in the 3D segmentation framework, the pseudocode will be listed first. The first stage is an optional one as the input mesh might not require pre-processing. The pre-processing is mainly to fix resolution and increase density. This is done via the resampling of the vertices to meet a fixed threshold. The threshold is fixed and does not to be very tight as the aim is just to fit the 2D footprint algorithm and not to produce better quality mesh. The resampling is implemented using recursive subdivision of mesh faces. Algorithms 1 and 2 show the involved pseudocode and the main loops and recursive structures. The inputs of the algorithm are the mesh M and threshold t_A while the output is the higher resolution mesh M'. The complexity of the algorithm is $O(n \log m)$ where n is the number of input mesh faces and m is the difference between a triangle mesh area and the defined area threshold.

Algorithm 1 Mesh resampling

Input: $\mathbf{M}, t_A$

Output: $\mathbf{M}'$

1: **for** $i = 1$ to n **do**
2: f = $\mathbf{M}$.triangles[i]
3: A_f = compute_area(f)
4: $\mathbf{M}'$.triangles.add(recursive_split(f, A_f, t_A))
5: **end for**
6: **return** $\mathbf{M}'$

Algorithm 2 Recursive Split

Input: f, A_f, t_A

Output: $\mathbf{f'}$

1: **if** $A_f < t_A$ **then**
2: $[f_1, f_2]$ = half_split(f)
3: A_{f_1} = compute_area(f_1)
4: A_{f_2} = compute_area(f_2)
5: $[\mathbf{f'_1}]$ = recursive_split(f_1, A_{f_1}, t_A)
6: $[\mathbf{f'_2}]$ = recursive_split(f_2, A_{f_2}, t_A)
7: $\mathbf{f'} = [\mathbf{f'_1}, \mathbf{f'_2}]$
8: **else**
9: $\mathbf{f'} = \mathbf{f}$
10: **end if**
11: **return** $\mathbf{f'}$

The second stage in the 3D segmentation framework is 2D footprint and projections generation. 2D projections generation is done on all the 3 x, y, and, z axes using 3 embedded loops with a predefined fixed number of iterations. The complexity of the 2D generations is $O(n^3)$ where n is the number of iterations per axis. The second part is the 2D footprints generation, which uses α-hull algorithm for each projection. As mentioned in chapter 3, the complexity of α-hull algorithm is $O(n \log n)$ [84], where n is the number of vertices in the input 2D projection. Thus, the complexity of the 2D footprints generation is $O(mn \log n)$, where m is the number of 2D projections and n is the number of vertices in each projection. Algorithms 3 and 4 show the code involved in this stage.

Algorithm 3 2D projection generation

Input: $\mathbf{M}, n$

Output: **P**

```
for i = 1 to n do
    for j = 1 to n do
        for k = 1 to n do
            calculate R_x, R_y, and R_z
            M.vertices = M.vertices * R_x * R_y * R_z
            Project to x − y plane
            add to P
        end for
    end for
end for
return P
```

Algorithm 4 2D footprint generation

Input: **P**

Output: **F**

```
for i = 1 to P.length do
    α−hull ( P[i] )
end for
return F
```

After the footprints are calculated, the third stage is to locate seed points that start the definition of cutlines. The minima rule is used in this stage, which follows the human approach and depends on the curvature. Although the minima rule is used, any other segmentation criteria can be deployed without any change in the other stages. An example is segmenting 3D mesh based on fitting results against

3D geometry solids such as ellipsoids [120]. In order to operate effectively, the algorithms in this stage requires faces neighborhood matrix. Using this structure, the algorithm calculates the angles between pairs of faces. As mentioned before, the seeds can be identified based on a user threshold or by locating outliers from curvature plots, as shown in Figure 4.2. The complexity of the algorithm is $O(n)$ as the inner loop has a fixed number. Algorithm 5 shows the code details, which assumes that faces in this case are triangles.

Algorithm 5 Minima rule seeds

Input: **M**

Output: **S**

```
1:  Ne = calculate_face_neighborhood(M)
2:  for i = 1 to Ne.length do
3:      for j = 1 to 3 do
4:          N_f1 = get_normal(M.faces[i])
5:          N_f2 = get_normal(Ne[i][j])
6:          θ = calc_angle(N_f1, N_f2)
7:          if θ satisfies the conditions then
8:              [v1, v2] = calc_common_vertices(M.faces[i], Ne[i][j])
9:              S.add(v1, v2)
10:         end if
11:     end for
12: end for
13: return S
```

Antipodal points location for the calculated seeds is done in the fourth stage. Chapter 3 discussed the proposed antipodal location algorithm and how 2D footprints are manipulated to determine the best match for the seeds. For each seed

vertex, the algorithm works only on the footprints that they are part of. The algorithm inputs are the 2D footprints and the calculated seed points. Pairs of seeds and their corresponding antipodal points are the expected output. The algorithm sorts the antipodal points per vertex according to Euclidean distance. The complexity of the algorithm is $O(mn)$ where m is the length of the seeds array and n is the length of the footprints' array. Algorithm 6 shows the antipodal location pseudocode.

Algorithm 6 Antipodal vertices location

Input: **S**, **F**

Output: **P**

```
1:  for i = 1 to S.length do
2:      for j = 1 to F.length do
3:          if S[i] belongs to F[j] then
4:              F_effective.add(F[j]
5:          end if
6:      end for
7:      for j = 1 to F_effective.length do
8:          v = get_2D_Antipodal(S[i], F_effective[j])
9:          A.add(v)
10:     end for
11:     closest = ∞
12:     chosen = null
13:     for j = 1 to A.length do
14:         if dist(A[j], S[i]) ¡ closest then
15:             closest = dist(A[j], S[i])
16:             chosen = A[j]
17:         end if
18:     end for
          P.add(S[i], chosen)
19: end for
20: return P
```

The fifth stage is the generation of the cutlines. This can be considered the last step in the main processing steps as the one after is for post processing. The

aim of this stage is to convert the pairs of points and antipodal points into closed contours. This requires the calculation of all the points that connect the pair from the two sides. The Dijkstra shortest path algorithm is deployed to calculate the first side. Then, a plane is formed from the points in the first side and connected points are added iteratively if they are the closest to the plane. The inputs are the mesh M and the pairs array P while the output is the contours list. The complexity of this algorithms stage is $O(mn\log n)$ where m is the number of pairs and n is the number of vertices per mesh where an optimized Dijkstra algorithm is implemented. Algorithm 7 shows how to complete the pairs of vertices into a closed contour.

Algorithm 7 Cutlines calculations

Input: **M**, **P**

Output: **C**

```
1:  for i = 1 to P.length do
2:      S1 = Dijkstra(P[i].first, P[i].second)
3:      p = fit_to_plane(S1)
4:      e1 = P[i].first
5:      e2 = P[i].second
6:      while e1 is not direct neighbor to e2 do
7:          closest = ∞
8:          chosen = null;
9:          for j = 1 to e1.neighbors.length do
10:             if dist(e1.neighbors[j], p) ¡ closest then
11:                 closest = dist(e1.neighbors[j], p)
12:                 chosen = e1.neighbors[j]
13:             end if
14:         end for
            e1 = chosen S2.add(chosen)
15:     end while
            C.add( [S1, S2] )
16: end for
17: return C
```

The final stage, which can be optional, is the post processing stage. Many enhancements can be done here such as subpart extraction, contour smoothing, semantic attachment, and global readjustment of subparts if more information is available in priori or by induction. An example of post processing is subparts extraction,

which transforms the cutlines contours into 3D independent solid parts. The extraction starts randomly at any cutline. A seed is selected to be a neighbor to the cutline. A flood fill algorithm is used on the seed to expand everywhere until the boundaries or cutlines are met. Flood fill is a recursive algorithm and its complexity is $O(mn)$ where m is the number of seeds and n is the number of vertices per subpart. This is calculated as any vertex is only visited once by keeping a global visited array. Algorithm 8 shows the details of the flood fill algorithm that calculate subparts out of cutlines.

Algorithm 8 Subparts extraction

Input: **M**, **C**

Output: **Parts**

1: **Seeds** = estimate_seeds(**M**, **C**)
2: **for** $i = 1$ to **Seeds**.length **do**
3: P = flood_fill(**Seeds**[i])
4: **Parts**.add(P)
5: **end for**
6: **return Parts**

Algorithms 1 to 8 suggests that the algorithm running time can be time consuming and hence not suitable for real-time processing. However, many of these algorithms were designed to be ready for parallelization to utilize the availability of multi cores found commonly in Graphics Processing Unit (GPU)s in desktop and laptop hardware. Algorithms such as 3, 4, 5, 6, and 7 all exhibit data independence between the algorithm's main loop iterations. Thus, the algorithm can be implemented in a parallel manner and with sufficient processing power can achieve the required speed for high demand applications.

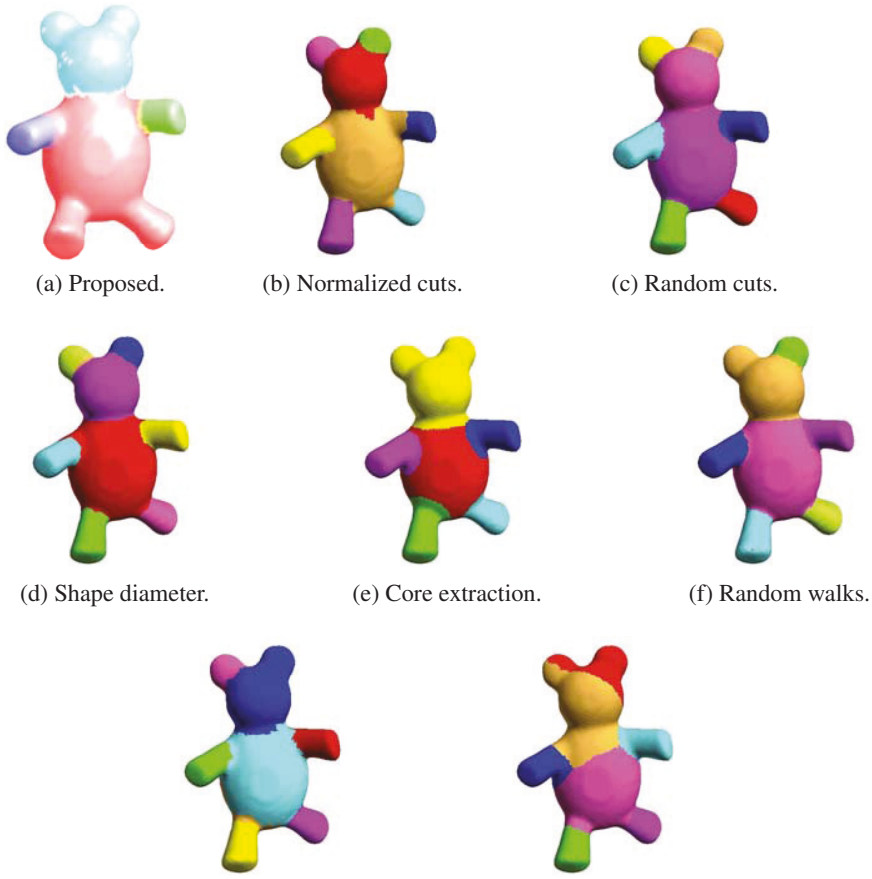
4.3 Output and results of the proposed 3D segmentation algorithms

For the sake of comparison, the proposed 3D mesh segmentation framework is tested against 3D objects acquired from a benchmark database by Chen *et al.* [6]. Different classes of 3D objects were used to compare the proposed framework against seven algorithms from the literature. The algorithms are namely, randomized cuts by Golovinskiy and Funkhouser [13], normalized cuts by Golovinskiy and Funkhouser [13], random walks by Lai *et al.* [38], fitting primitives by Attene *et al.* [48], *K* means by Shlafman *et al.* [1], shape diameter by Shapira *et al.* [7], and core extraction by Katz *et al.* [121]. Chen *et al.* benchmark database also includes manual segmentation acquired from users but as interactive segmentation is not covered, they will not be included.

This section shows the results for visual comparison while the next section will show comparisons through literature metrics. The visual results show that the proposed framework is competitive while having the advantage of being parameter free and generic for different cut criteria. Figures 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, show the results for all algorithms side by side with the proposed one. For all the objects, the minima rule cut criteria were deployed.

4.4 Benchmarking of the proposed 3D mesh segmentation algorithms

The 3D mesh segmentation process is not totally defined between human beings. Although cut criteria exists, different human beings can cut in different ways. Thus, it is a subjective process and there is no absolute answer or result. The previous



(a) Proposed. (b) Normalized cuts. (c) Random cuts.

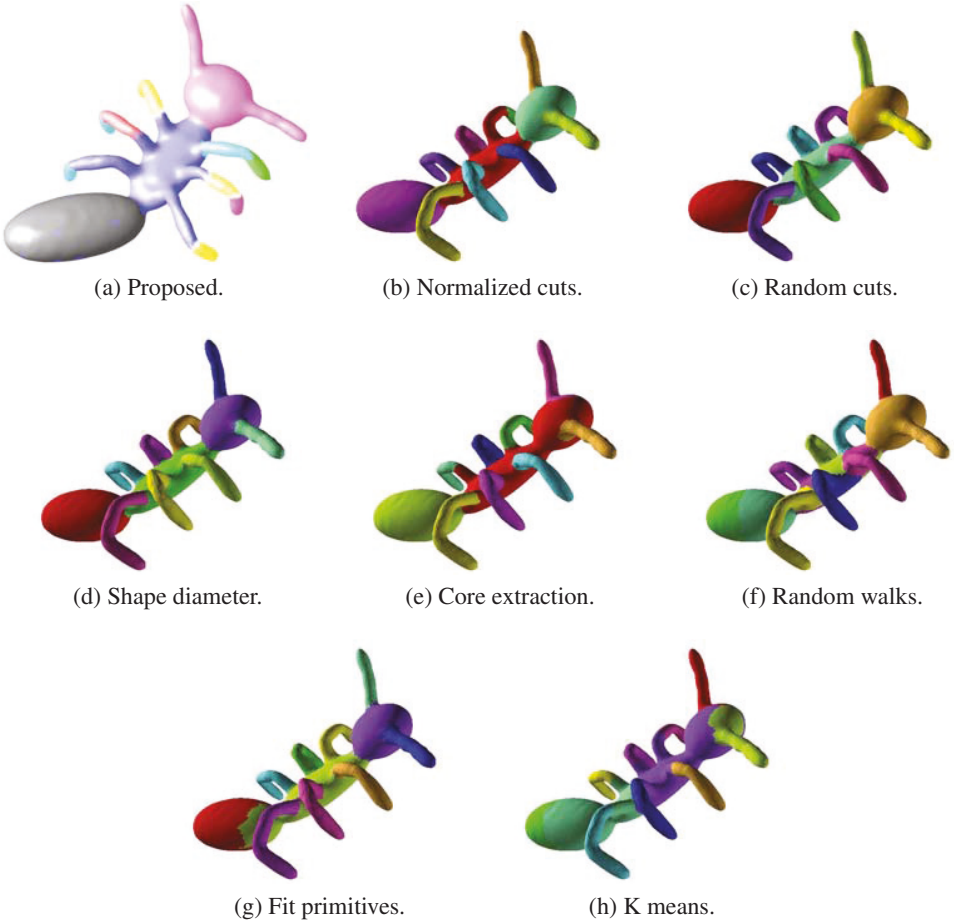
(d) Shape diameter. (e) Core extraction. (f) Random walks.

(g) Fit primitives. (h) K means.

Figure 4.4: Experimental results for the teddy 3D object against other literature methods.

section shows an example of qualitative evaluation where images are shown side by side. This is common in visual applications to show that the algorithm satisfies the requirements and how it is distinguished from other peers' algorithms. Other examples of qualitative characteristics are about segmentation type (e.g. part, boundary), boundary smoothness, pose sensitivity, computational complexity, and input and control parameters.

Another evaluation mechanism is the quantitative one. This quantitative evaluation uses proposed benchmarks that compare the performance of an algorithm



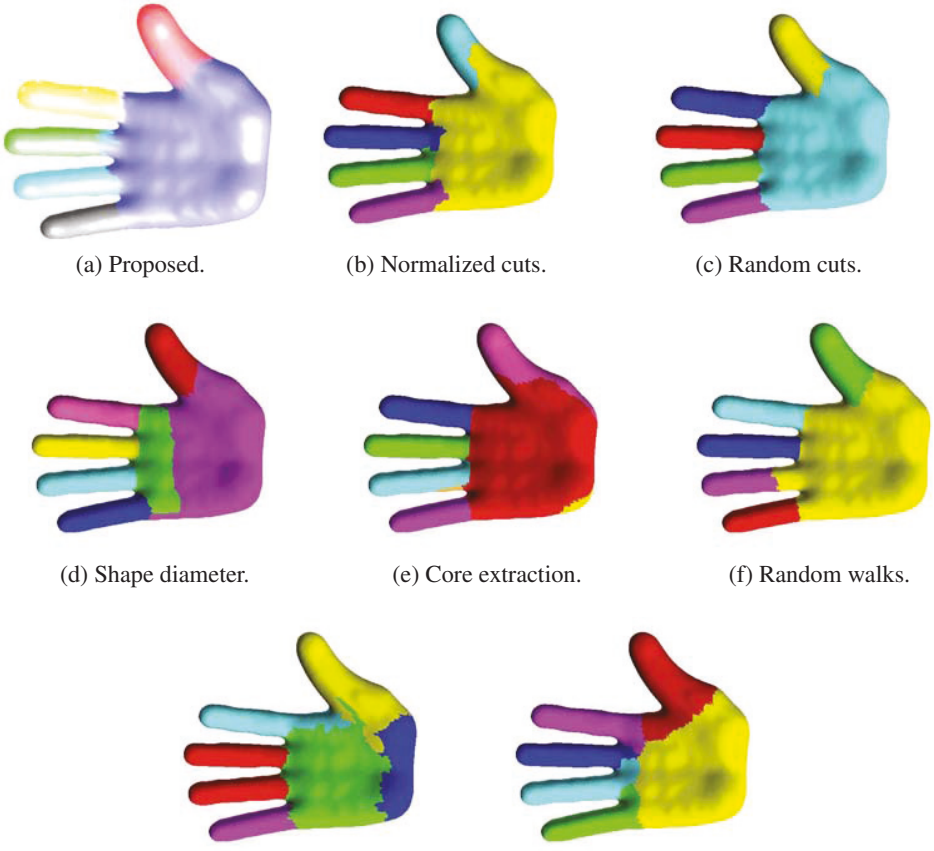
(a) Proposed. (b) Normalized cuts. (c) Random cuts.

(d) Shape diameter. (e) Core extraction. (f) Random walks.

(g) Fit primitives. (h) K means.

Figure 4.5: Experimental results for the ant 3D object against other literature methods.

with others and with a baseline segmentation. There are several proposed algorithms in the literature that aim to benchmark the segmentation algorithms. In this thesis, the benchmarking process will follow the literature approaches. The comparison will be with the major 3D segmentation algorithms in the literature. They are namely, K-means [1], random walks [38], fitting primitives [48], normalized cuts [13], randomized cuts [13], core extraction [121], and the shape diameter function [7]. The baseline is the average human 3D segmentations collected by Chen *et al.* [6] through an online data collection facility from diverse samples all over the globe.



(a) Proposed. (b) Normalized cuts. (c) Random cuts.

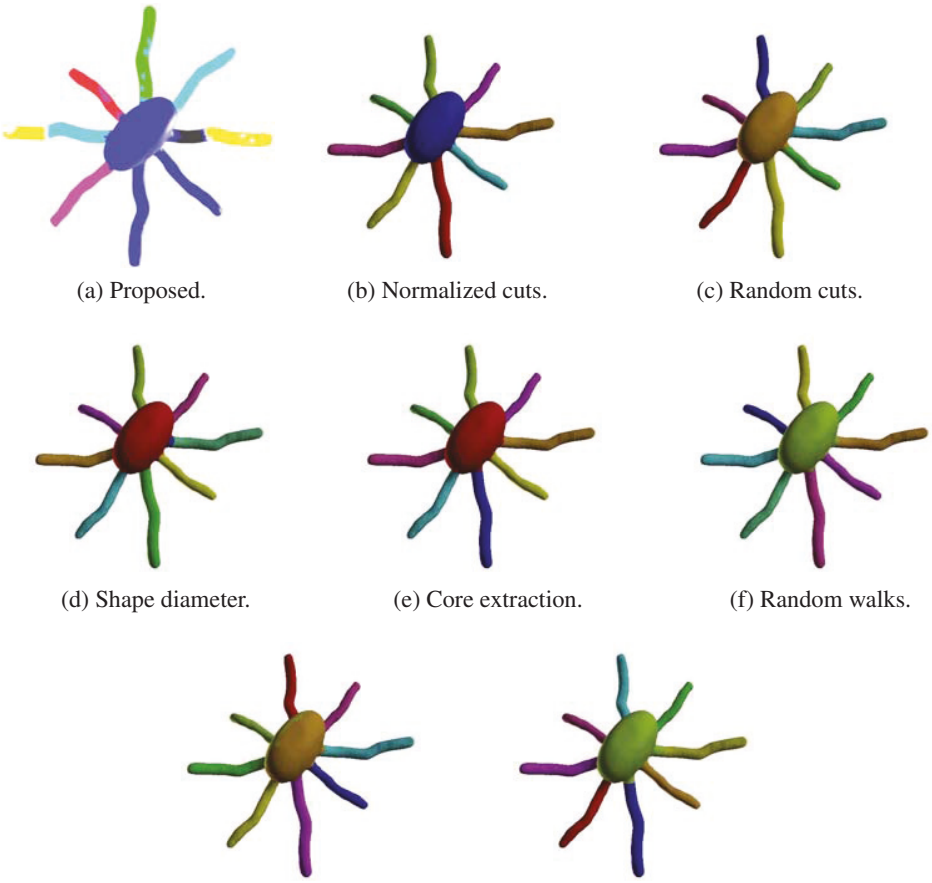
(d) Shape diameter. (e) Core extraction. (f) Random walks.

(g) Fit primitives. (h) K means.

Figure 4.6: Experimental results for the hand 3D object against other literature methods.

The performance of other peer algorithms is also extracted from the corresponding benchmarking papers [6].

In this section, the proposed algorithm will be compared against the literature metrics. The used metrics are: the cut discrepancy index (CDI), the Hamming distance index (HDI), the global consistency index (GCI), the local consistency index (LCI), and the rand index (RI). These metrics are proposed in the literature and have been used in several publications. The goal is usually to measure how close are the cutlines in compared to the baseline and how consistent are the segment



(a) Proposed. (b) Normalized cuts. (c) Random cuts.

(d) Shape diameter. (e) Core extraction. (f) Random walks.

(g) Fit primitives. (h) K means.

Figure 4.7: Experimental results for the octopus 3D object against other literature methods.

interiors are respect to the baseline. More details of these metrics or comparison indices can be found in [33]. Thus, only the main equation and the comparison results will be listed here. The calculated numbers here are the average number for all the experimented models, shown in the results section.

The **first** metric is the CDI. This metric is for the quality of the cutlines with respect to the base segmentation. The following equations are used to calculate it

(a) Proposed.

(b) Normalized cuts.

(c) Random cuts.

(d) Shape diameter.

(e) Core extraction.

(f) Random walks.

(g) Fit primitives.

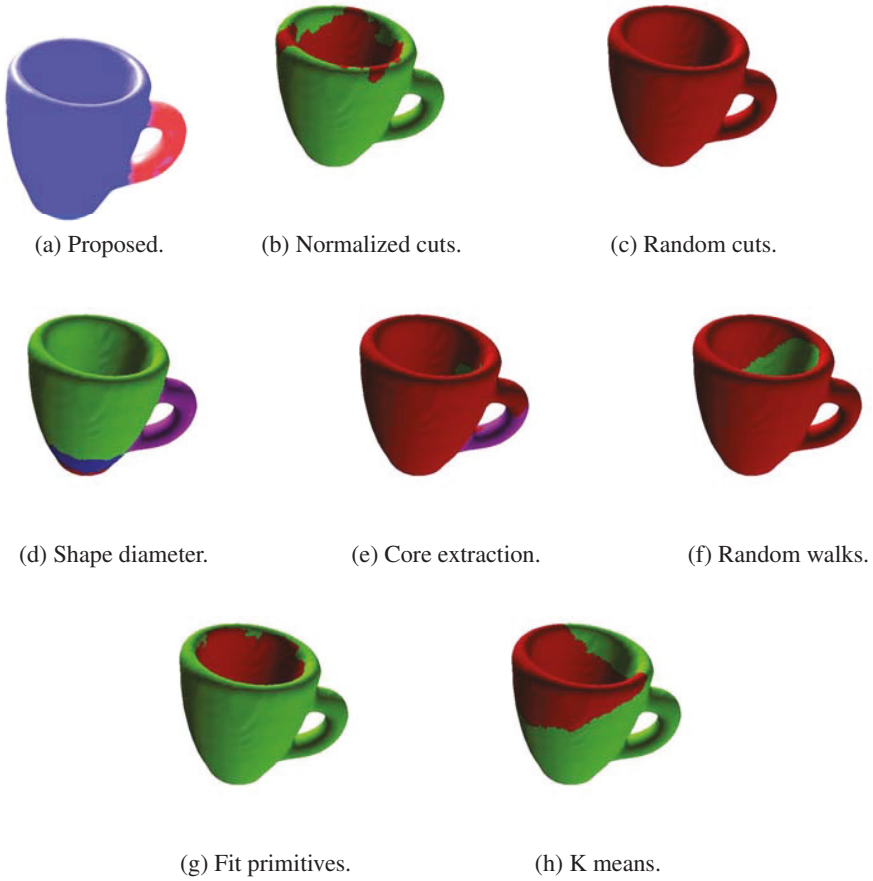
(h) K means.

Figure 4.8: Experimental results for the chair 3D object against other literature methods.

$$CDI(S_1,S_2) = \frac{DCD(S_1 \Rightarrow S_2) + DCD(S_2 \Rightarrow S_1)}{avgRadius} \tag{4.4.1}$$

where S_1 and S_2 are two segmentations of the mesh. *avgRadius* is the average Euclidean distance from a point on the surface to the centroid of the 3D mesh. *DCD* is a directional function defined as:

$$DCD(S_1 \Rightarrow S_2) = mean\{d_G(p_1,C_2), \forall p_1 \in C_1\} \tag{4.4.2}$$



(a) Proposed. (b) Normalized cuts. (c) Random cuts.

(d) Shape diameter. (e) Core extraction. (f) Random walks.

(g) Fit primitives. (h) K means.

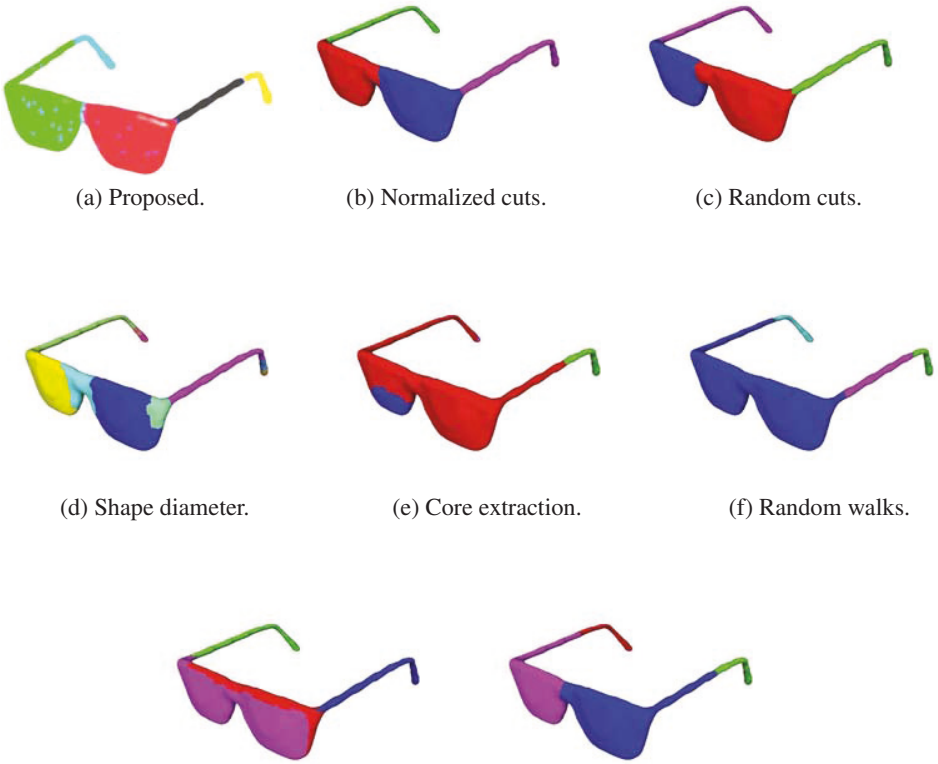
Figure 4.9: Experimental results for the cup 3D object against other literature methods.

and the geodesic distance from a point $p_1 \in C_1$ to a set of cuts C_2 is

$$d_G(p_1, C_2) = \min\{d_G(p_1, p_2), \forall p_2 \in C_2\} \tag{4.4.3}$$

where C_1 and C_2 are the point on the segment boundaries of S_1 and S_2 respectively. The average result of the proposed algorithm is 0.29. Figure 4.11 shows the results plot compared to the literature algorithms. The lower the value of the CDI, the better the match between the algorithm performance and the baseline.

The hamming distance index (HDI) is the **second** metric, which is focused on



(a) Proposed. (b) Normalized cuts. (c) Random cuts.

(d) Shape diameter. (e) Core extraction. (f) Random walks.

(g) Fit primitives. (h) K means.

Figure 4.10: Experimental results for the glasses 3D object against other literature techniques.

the region difference between the proposed algorithm and the base segmentation model. The calculation method for HDI uses the following equations

$$HDI(S_1,S_2) = \frac{1}{2}(M_r(S_1,S_2) + F_r(S_1,S_2)) \tag{4.4.4}$$

and

$$M_r(S_1,S_2) = \frac{D_H(S_1 \Rightarrow S_2)}{\|S\|} \tag{4.4.5}$$

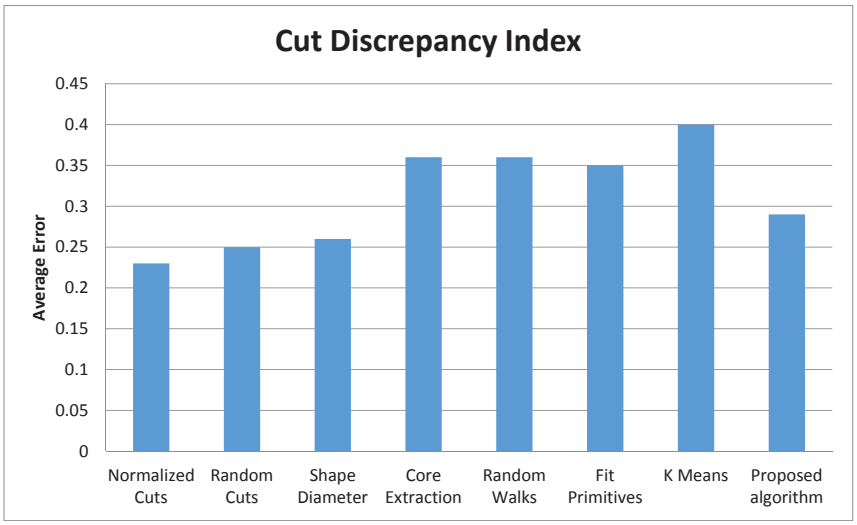


Figure 4.11: Benchmark results for the CDI metric.

$$F_r(S_1,S_2) = \frac{D_H(S_2 \Rightarrow S_1)}{\|S\|} \tag{4.4.6}$$

$$D_H(S_1 \Rightarrow S_2) = \sum_i \|R_2^i R_1^{i_t}\| \tag{4.4.7}$$

$$i_t = argmax_k \|R_2^i \cap R_1^k\| \tag{4.4.8}$$

where S_1 and S_2 are two segmentations of the mesh. The operator represents the set differencing and R_k^i. D_H is the directional Hamming distance [122]. The average Hamming distance result of the proposed algorithm is 0.13. Figure 4.12 shows the result plot compared to the literature algorithms. The best performing algorithms are the ones with low HDI.

The **third** metric that measures the ratio of the number of vertices that are not shared between the segmentation algorithm and the base segmentation is the GCI metric. This is a global metric and can be calculated through the following

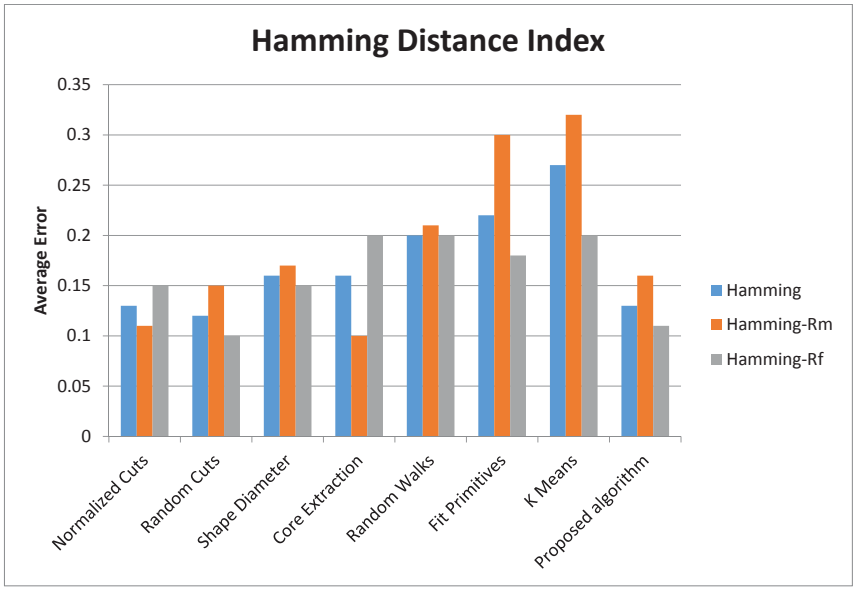


Figure 4.12: Benchmark results for the HDI metric.

$$GCI(S_1,S_2) = \frac{1}{N}\min\{\sum_i L_{3D}(S_1,S_2,\upsilon_i), \sum_i L_{3D}(S_2,S_1,\upsilon_i)\} \tag{4.4.9}$$

and

$$L_{3D}(S_1,S_2,\upsilon_i) = \frac{|R(S_1,\upsilon_i)R(S_2,\upsilon_i)|}{|R(S_1,\upsilon_i)|} \tag{4.4.10}$$

where $R(S,\upsilon_i)$ is the region in segmentation S that contains the vertex υ_i and N is the number of vertices. The average result of the proposed algorithm is 0.13. Figure 4.13 shows the results plot compared to the literature algorithms. GCI is a probability between the range of $[0,1]$ and is preferred to be of low value. This means higher global consistency.

LCI, the **fourth** metric, is another probability metric, which is similar to the GCI but operates on the local level in order to complement the GCI. Thus, LCI low

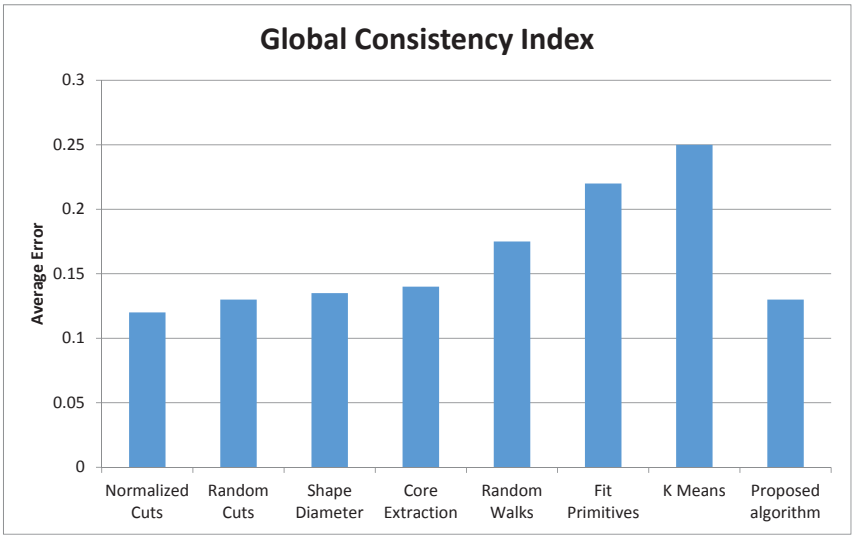


Figure 4.13: Benchmark results for the GCI metric.

values are better as is the case for GCI and the range of values is also between $[0,1]$. The equation that calculates LCI is

$$LCI(S_1,S_2) = \frac{1}{N}\sum_i \min\{L_{3D}(S_1,S_2,v_i), L_{3D}(S_2,S_1,v_i)\} \tag{4.4.11}$$

where the same functions are borrowed from the LCI equation. The average result of the proposed algorithm is 0.06. Figure 4.14 shows the results plot compared to the literature algorithms.

RI is the **last** metric and focuses on a different aspect of comparison than the previous metrics. It computes the pairwise label relationships. This is defined as

$$RI(S_1,S_2) = \frac{1}{\binom{n}{r}} \sum_{i,j,i<j} \mathbf{I}(l^i_{S_1} = l^j_{S_1})(l^i_{S_2} = l^j_{S_2}) + \mathbf{I}(l^i_{S_1} \neq l^j_{S_1})(l^i_{S_2} \neq l^j_{S_2}) \tag{4.4.12}$$

where $\mathbf{I}$ is the identity function, N is the number of vertices, and $l^i_{S_k}$ is the corresponding label of all elements contained in region R_i of segmentation S_k. The range of values is $[0,1]$ with 1 to be the best match. The average result of the proposed

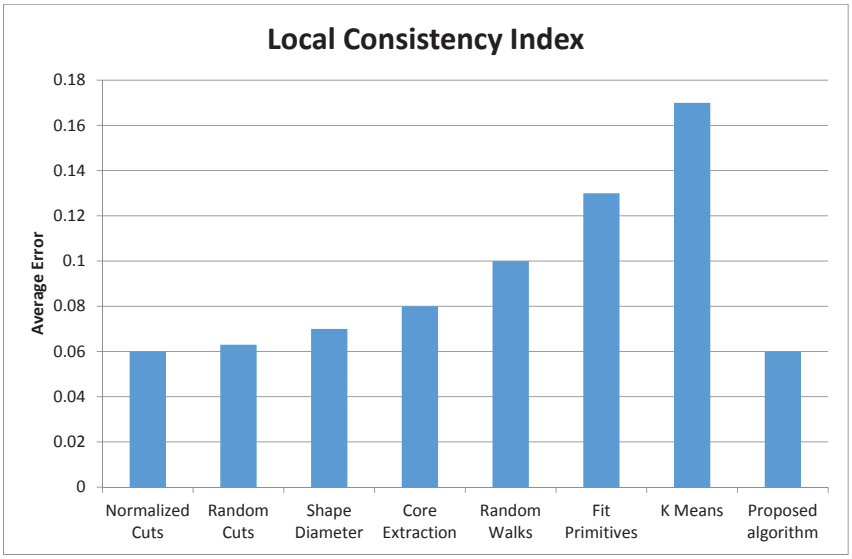


Figure 4.14: Benchmark results for the LCI metric.

algorithm is 0.173. Figure 4.15 shows the results plot compared to the literature algorithms. It is to be noted that the reported values here are $1 - RI(S_1, S_2)$ for consistency purposes.

Many of the used metrics have some criticism in the literature. Examples of these weaknesses are mentioned in chapter 2. An important problem in the field also emerges from the algorithms themselves. Many algorithms require parameters and they are not the same in type or count. In future, this is expected to disappear, as the trend will be towards fully automatic methods to support autonomy in intelligent machines. Also, new metrics can be proposed such as ones that make use of different 2D projections for comparison. Each 2D projection has a number of 2D segments that can be compared with the 3D segmentation.

The benchmark results show that the proposed algorithm is competitive against the literature methods. It might not always take the lead but it has a an important advantage, which is being parameter free. On the other hand, some features are

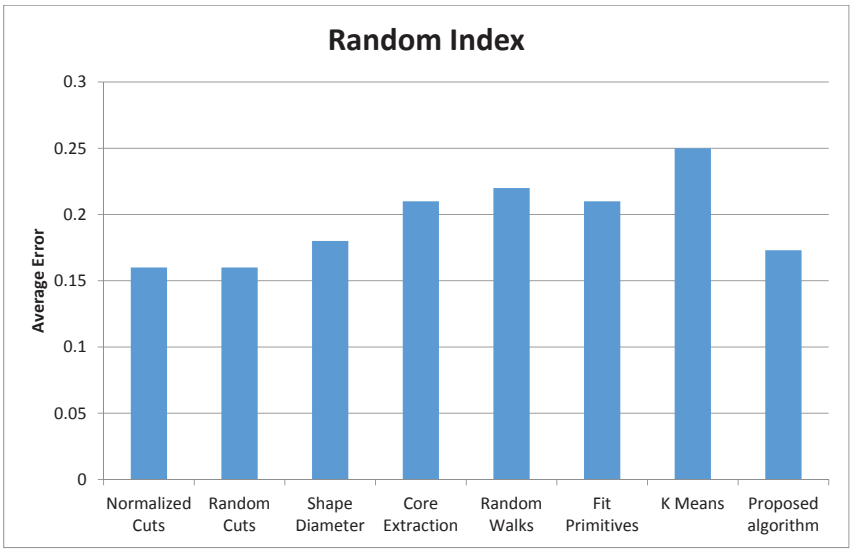


Figure 4.15: Benchmark results for the RI metric.

missing in the proposed algorithm such as being pose invariant. However, this is not always desired in all applications. Two example applications can show how the pose invariant feature is controversial. The first is the deformation transfer application where a pose invariant is not good as it will make it hard to capture deformation changes. The other example is 3D mesh retrieval where an object is desired to be recognized irrespective of its pose. However, even in these types of applications, parts of the proposed algorithm such as the antipodal location algorithm can be utilized.

4.5 Conclusion

This study presents a novel step towards the automation of the 3D mesh segmentation process. Literature current literature suffer from the dependency on the input objects or tuning parameters. The proposed algorithms are generic for any input

mesh or segmentation theory. Using fused 2D search results to refine the 3D results provides robust coverage of the input mesh. A drawback of the algorithm is that it is not pose invariant as it is based on the topology changes. Future directions can be towards the parallelization of the algorithm to improve the speed. Segmentation theories other than cognition theory can also be tested against the proposed algorithm.

Chapter 5

Study of deformation transfer between isometric objects

This study addresses a major challenge in data-driven haptic modeling of deformable objects. Data-driven modeling is done for specific objects and is difficult to generalize for nearly isometric objects that have similarities in semantics or topology. This limitation prevents the wide use of the data-driven modeling techniques when compared with parametric methods such as finite element methods. The proposed solution is to incorporate deformation transfer methods when processing similar instances. The contribution of this research focused on the novel automatic shape correspondence method that overcomes the problems of symmetry and a semantics presence requirement. The results show that the proposed algorithm can efficiently calculate the correspondence and transfer deformations for a range of similar 3D objects.

5.1 Deformation transfer preliminaries

Haptic modeling of deformable models is an important field of computer haptics. Deformable models rendering is a challenging task [123] as the material behavior of the object is required to be modeled along the object topology and structure. Haptic rendering is a resource demanding process when compared to computer graphics applications. A common requirement for a smooth haptic rendering is 1,000 Hz refresh rate while it is only between 30 to 60 for a graphics application [124].

Several methods exist in this field [125]. However, not all of them are physically loyal or at least try to follow the constitutive physics laws of material behavior [126]. The common accurate main categories can be classified into two large groups: parametric modeling techniques and data-driven modeling ones [127]. The parametric methods examples includes finite element methods (FEM), finite volume methods (FVM), and extended finite element methods (XFEM). They are known for accuracy and their independence of the object topology as they use generic estimated material parameters. However, they sometimes fail the real-time requirement without approximations [128] and cannot easily model complex or heterogeneous objects. On the other hand, data-driven modeling techniques exist, which do not require an explicit model. The model is calculated through empirical data [129]. Thus, it can model complex objects and material behavior. Besides, in the run time it is fast as the model is already created offline and no complex calculations are required. However, the model is fixed to one object instance and cannot be easily generalized to related instances that are slightly different [130].

In order to fully utilize data-driven haptic rendering methods, the generalization limitation needs to be relaxed. This is a required feature when aligned with the fact that the data collection for rendering is a long process and can take up to several

days in some cases [131]. In this study, we investigate how deformation transfer techniques developed in the computer graphics domain can be handy in haptics application. A novel automatic shape correspondence algorithm is proposed to relate any objects that are isometric or nearly isometric. The correspondence algorithm overcomes challenges in the literature such as symmetry [132] and the requirement of semantics existence.

The rest of the chapter is organized as follows: section 2 analyzes the related work in the literature. Section 3 discusses the problem statement. Section 4 illustrates the proposed algorithm while section 5 shows more of the problem of antipodal point location. Section 6 is dedicated for the implementation details and experimental results. Section 7 has the concluding remarks and possible future directions.

5.2 Related work of deformation transfer techniques

Data-driven haptic modeling is a modeling method that has the power of modeling complex material behavior in a real-time manner. This make use of machine learning techniques such as artificial neural networks (ANN) to build a model empirically. This can be considered as one of two main approaches to have a physically loyal haptic rendering. The other method is parametric rendering where a model is characterized by a set of parameters and governing behavior equations. An example is finite element method (FEM), which is considered to have plausible accuracy. However, it suffers from being unable to meet the real-time requirement and the high refresh rate of approximately 1000 Hz for stable haptic user experience. An excellent survey about physical rendering methods of deformable objects can be found in [126].

Data-driven methodology for haptic rendering has attracted many researchers recently. Pai *et al.* [133] demonstrated early work on estimating the stiffness matrix K in the equation

$$f = Ku \tag{5.2.1}$$

where u is the displacement vector and f is the external forces vector. Cretu and Petriu [134] and Morooka *et al.* [135] demonstrated the usage of (ANN) and how to reduce the number of used vertices. An alternative approach of radial bases functions (RBF) were used in [131, 136]. Research was also done in the area of data acquisition and processing [130, 137].

Deformation transfer is a relatively new topic in computer graphics. It started with applications for facial expressions transfer [138]. The input is a source reference pose, a deformed source pose and a target reference pose while the output is a deformed target pose [139]. Several approaches exist in the literature. Deformation transfer to a subset of the mesh vertices and estimating the rest via interpolation was proposed by Zayer *et al.* [140]. Baran *et al.* [141] focused on the semantics and how to preserve them rather than arbitrary deformation. Multi-component mesh processing, using spatial deformation transfer were demonstrated by Ben-Chen *et al.* [139]. This was lately extended and generalized by Zhou *et al.* [142]. The general note on most of the literature is that the user exists in the loop to specify key similar vertices.

A main methodology for automating deformation transfer is shape correspondence as the selection of point needs to be automatic to be efficient. This is a common methodology for many applications such as mesh morphing [143], mesh parameterizations [144], shape matching [145], and shape registration [146]. A recent survey by van Kaick *et al.* [147] demonstrated the latest approaches and

tried to provide classifications such as full versus partial correspondence and being dense or sparse from a correspondence point of view. Since the survey came out there have been several enhancements. Examples include the work of Sahillioglu and Yemez [132] that used expectation-maximization (EM) algorithm to establish isometric shape correspondence. Others also used different methods such as fuzzy correspondence [148], functional maps [149], and MorseSmale complex of the auto diffusion function [150].

5.3 Deformation transfer problem statement

The simulated deformable 3D object $\mathbf{M}$ is assumed to be a boundary mesh discretized into finite vertices $\mathbf{V}$ and faces $\mathbf{F}$. Thus $\mathbf{M} = \{\mathbf{V}, \mathbf{F}\}$. The data-driven modeling of the global behavior of the object material results in a model where every external force $\mathbf{f}$, of certain magnitude and direction, has a correspondent visual deformation and force feedback. The deformation is in terms of displacement in x, y, and z axes, while the force feedback is in terms of magnitude and force vector direction.

Using the same literature symbols, we assume two objects $\mathbf{S}$ and $\mathbf{T}$, where $\mathbf{S}$ is the source and $\mathbf{T}$ is the target. All vertices of $\mathbf{S}$ need to be related to vertices of $\mathbf{T}$. To meet this requirement, the number of vertices of $|\mathbf{S}|$, need to be equal to the number of vertices of $|\mathbf{T}|$. As this condition cannot be met the correspondence needs to be of one to many types.

Due to the nature of the study, which is data-driven haptic rendering, the problem inputs, outputs and methodology have to be different. The deformation transfer problem in the literature is about calculating the shape of $\mathbf{T}$ if it undergoes the same

deformation function that were applied to **S**. The challenge, as shown in the previous section, is in how to produce the output through a few sparsely selected points by the user. The case in the data-driven haptics domain is quite different. The goal is to match every point in the source with one or more in the target and hence transfer the visual and force feedback model to the new object.

Assuming that the generated data-driven model for **S** is

$$D(f, v_{\mathbf{S}}, \mathbf{S}) = \mathbf{S}' \tag{5.3.1}$$

where f is the external force, v_S is the vertex that belongs to **S** where the force will be applied, and $\mathbf{S}'$ is the deformed version of **S**. Then, we need to find the data driven model for **T**

$$D(f, v_{\mathbf{T}}, \mathbf{T}) = \mathbf{T}' \tag{5.3.2}$$

where v_T is the vertex that belongs to **T** where the force will be applied. This needs to be under the following condition

$$\min |D_{iso}(\S) - D_{iso}(\S')| \tag{5.3.3}$$

where

$$D_{iso}(\S) = \frac{1}{|\S|} \sum_{(v_{\mathbf{S}_i}, v_{\mathbf{T}_j}) \in \S} d_{iso}(v_{\mathbf{S}_i}, v_{\mathbf{T}_j}) \tag{5.3.4}$$

and

$$d_{iso}(v_{\mathbf{S}_i}, v_{\mathbf{T}_j}) = \frac{1}{|\S| - 1} \sum_{\substack{(v_{\mathbf{S}_l}, v_{\mathbf{T}_m}) \in \S \\ (\S v_{\mathbf{S}_l}, v_{\mathbf{T}_m}) \neq (v_{\mathbf{S}_i}, v_{\mathbf{T}_j})}} |g(v_{\mathbf{S}_i}, v_{\mathbf{S}_l}) - g(v_{\mathbf{T}_j}, v_{\mathbf{T}_m})| \tag{5.3.5}$$

where § is the set of correspondence pairs between the source and the target and §′ is of the same type of sets but between the deformed versions. $g(.,.)$ is the geodesic distance between two vertices. The geodesic distance is often used in the literature as it is a distance-preserving mapping and hence a good method for comparison between the mesh pair and their deformed version pair.

The correspondence algorithm needs to minimize the above function to maintain isometry. This needs efficient data structures, descriptive features, and search strategies. The data structures need to enable easy features extractions. The features need to be unique, comparable, and easy to calculate. Also, the search strategies are required to be efficient and exhaustive. The next sections will show the proposed algorithms and walk through it step by step.

Once the correspondence is identified, the deformation function is transferred. In the case of one to many this needs an averaging function. Unlike other deformation transfer research there is no need to estimate deformation as the deformation function is transferred for all possible external interactions based on the material behavior.

5.4 The Proposed deformation transfer algorithm

The proposed algorithm main idea is to transform the input mesh into other formats that enable easier matching between the source and the destination. Figure 5.1 shows the flowchart of the algorithm. As stated in the problem statement, the algorithm inputs are the two boundary mesh in simple format and the data-driven deformation model of the source. The output is the data-driven deformation model of the destination after the correspondence between the two mesh are found out. The algorithm has five steps that will be stated in this section. More details in supporting

methodologies that are used within the algorithm to follow in the next sections.

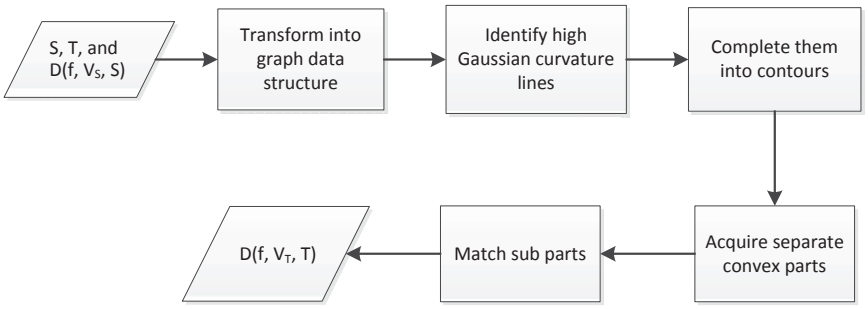


Figure 5.1: The proposed algorithm flowchart.

In order to process the inputs properly, they are firstly transformed into an appropriate data structure. This is required because the used formats for mesh representation are focused on the listing of absolute locations of the vertices and faces. The chosen data structure is a bidirectional graph because it is simple and shows the basic adjacency relationship between the vertices. Moreover extra data can be extracted efficiently, such as the shortest path between two vertices [56]. The graph is built using faces information and edge traversal cost is defaulted to flat value of 1.

The second step is to identify edges and vertices with a high Gaussian curvature [151]. These serve as efficient candidates for mesh segmentation according to several human perception theories such as the minima rule [17]. The curvature angle θ is determined using the following equation

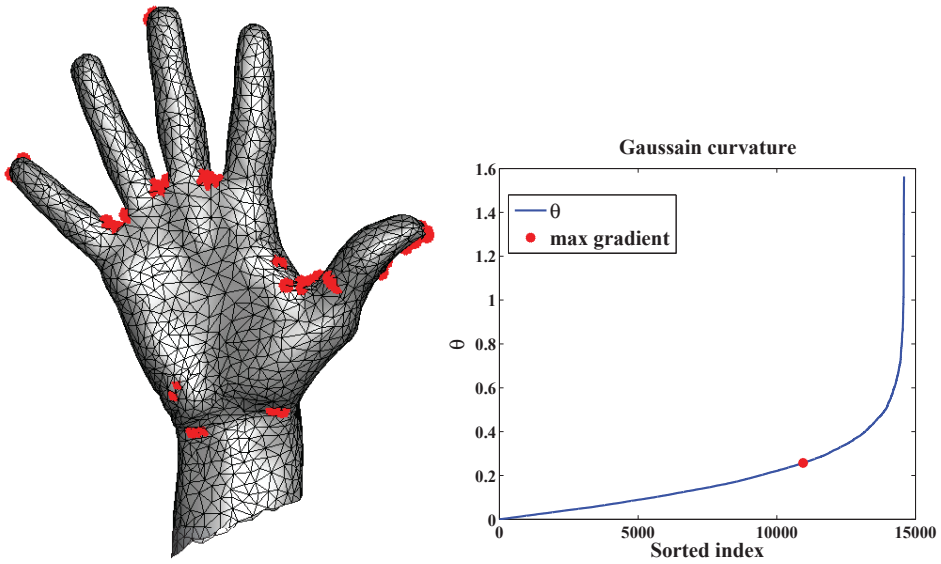
$$\theta = \arcsin \overrightarrow{\|n_{f1}\|} \times \overrightarrow{\|n_{f2}\|} \tag{5.4.1}$$

where $\overrightarrow{\|n_f\|}$ is a face normal. The θ value can then be compared to a predefined threshold or the highest 10% vertices in curvature for example can be selected. Figure 5.2 shows examples of selected high curvature vertices in 3D mesh. The

curve $f(\theta)$ in Figure 5.2b is the result of sorting θ values while the red dot is acquired using

$$\max \frac{d^2}{dx^2} f(\theta) \tag{5.4.2}$$

For the sake of segmentation, isolated vertices are not considered and only adjacent groups that form a semi contour are selected.



(a) High curvature points (red) in a hand mesh.(b) Sorted plot of the gaussian curvature of the hand mesh.

Figure 5.2: Identification of high curvature points in 3D meshes.

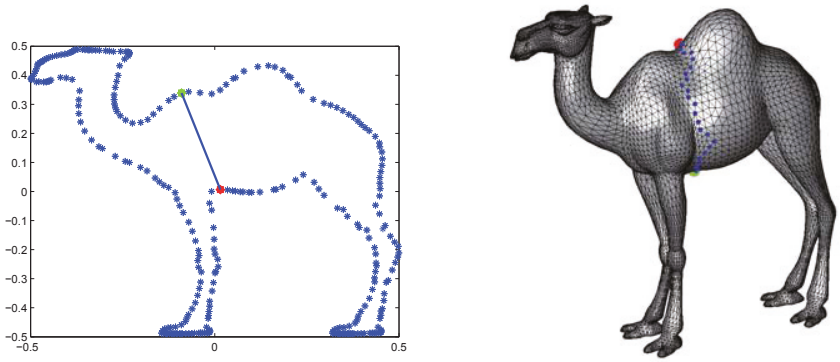
The 3D segmentation of the mesh requires the formation of closed cutlines. The previous step usually only generates semi-contours that need to be completed. The contour completion process, which is the third step, can utilize advances in 3D mesh segmentation and geometry processing. In this research, vertex antipodal location algorithm is utilized and then followed by ellipse fitting to produce a complete contour. The detailed illustration of the used method will be stated in the next section.

The fourth step is dedicated to acquiring separate convex parts that have a hierarchical form. This is easily extracted from the cutlines using flood fill like [152] algorithm. The starting seeds are selected close to the cutlines edges by selecting an edge and choosing the third point in the two faces that are connected via the edge. Flood fill is a recursive algorithm that continues until there are no possible vertices to visit. After this step, the problem of matching is greatly reduced to small sub problems where simpler sub parts are matched.

The final step is to register and match the sub parts. Due to the preservation of the hierarchial relations and the relatively simple structure of the parts as they are nearly convex, matching can use many methods from the literature. An efficient method is tree matching [153], which matches the similar core parts and then the descendants are automatically matched as well.

5.5 Antipodal location

This section is dedicated for antipodal location algorithm as it is a core part of the proposed framework. The antipodal point of a vertex that belongs to a 3D mesh can be linguistically defined as being 'diametrically opposite', according to Merriam-Webster dictionary. This is not far from scientific definitions in other domains such as geography [97] and mathematics [98] where the earth is used or geometric spheres to define the term. Figure 5.3 shows an example of 2D and 3D antipodal vertices in 2D shapes and 3D mesh respectively. The reader can notice that the case n 2D is much simpler than the 3D boundary mesh [7]. The antipodal location problem is important for many applications such as in robotics and computer graphics.



(a) 2D case. (b) 3D case.

Figure 5.3: Example of vertex antipodal.

5.5.1 2D antipodal vertex calculation

As the location of 3D antipodal vertices is a difficult task in the 3D domain [7], the problem is transferred into the 2D domain first to get candidates and then one is chosen to be the 3D vertex antipodal. Multiple 2D projections of the 3D mesh can be generated through rotation and in each projection where the vertex υ is visible the 2D antipodal point υ_a is calculated. The result is a set $S = \{\upsilon_{a_1}, \upsilon_{a_2}, ..., \upsilon_{a_n}\}$ where n is the number of successfully located antipodal points. This leads to three cases:

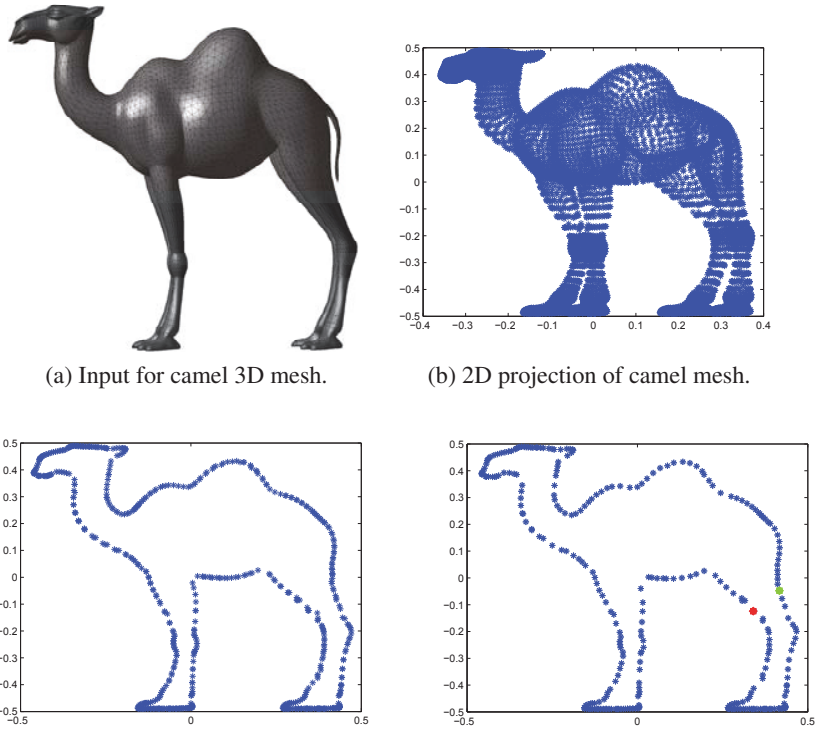
1. $n = 0$, The 3D antipodal cannot be located. This usually happens when the number of projections is not sufficient.

2. $n = 1$, The 3D antipodal is located and is the same as the suggested 2D antipodal. No options are available.

3. $n > 1$, This is the common case where multiple 2D antipodal points are located for multiple 2D projections. However, there is a requirement to choose only one member of the set. In this study, the selected 3D antipodal point

is the one with maximum Euclidean distance from the source vertex υ and where the line segment between υ and υ_a of the 3D vector $\overrightarrow{\upsilon\upsilon_a}$ does not intersect with any face of the 3D mesh.

The 2D antipodal location process is visualized in Figure 5.4, which is following the algorithm used in [154]. There are three steps to acquire a 2D antipodal and can be summarized as follows:



(a) Input for camel 3D mesh.

(b) 2D projection of camel mesh.

(c) 2D footprint of camel mesh.

(d) Camel 2D antipodal pair. Source is red and destination is green.

Figure 5.4: Visualisation of 3D object at each processing step for 2D antipodal search.

- For every projection the 3D mesh is rotated with certain angles θ ρ, and ϕ around x, y, and z axes respectively. This is usually done via nested iterations

with small step to produce as many permutations as possible. The projection is then acquired by eliminating the z dimension, as shown in Figure 5.4b.

- The 2D projection by itself cannot be easily processed. Thus, a better format is to extract the 2D footprint of the formed set of 2D points. The 2D footprint algorithm α-shape [84] is used. Although the algorithm is dependent on the α parameter, the value of α can be fixed as the mesh projection is usually dense. Even in cases of fairly low mesh resolution, restamping process can be considered for the sake of the antipodal point calculation. Figure 5.4c shows a calculated 2D footprint of 2D mesh projection. The acquired 2D footprint is then linearized to be ready for the next step.

- The last step is to locat the 2D antipodal point of a certain source vertex. After the previous step, the vertex is a member of a line and a normal $\overrightarrow{n}$ can be defined on that line. The normal intersections with other lines are calculated. This might create an out of mesh problem where more than one intersection happens, as shown in Figure 5.5. This can be solved using point in the polygon approach [109].

5.5.2 3D antipodal vertex calculation

The 3D antipodal point is calculated based on the results of the 2D search through the generated 2D projections. As discussed, the selection is required when there are multiple candidates. The used heuristics are based on the facts that the antipodal point and its source are diametrically opposite. For the purpose of our study of shape correspondence, the final results can be rectified based on local neighborhood. For a line of source vertices, the antipodal points should form a relatively similar connected line and hence any obvious outliers will be rejected.

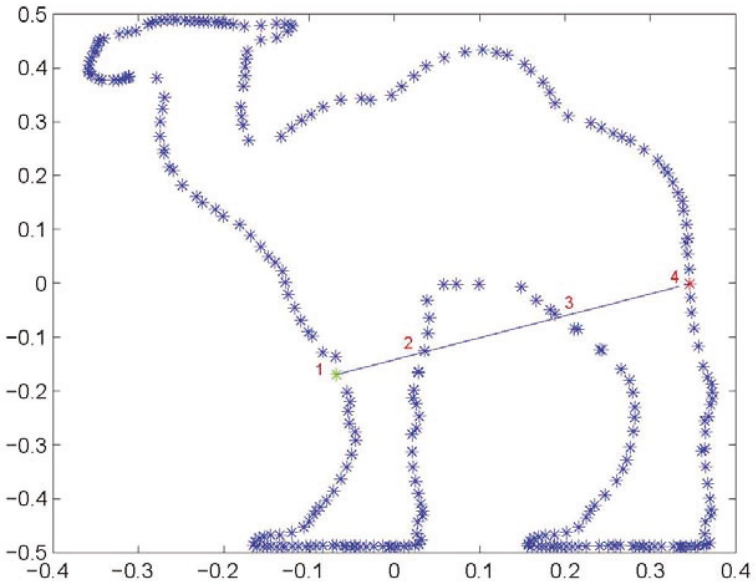


Figure 5.5: Possible solutions to navies' antipodal selection using point in polygon algorithm.

After locating the 3D antipodal points of each semi contour, the contour is completed using ellipse fitting algorithm. Figure 5.6 shows an example of the method, which fits the provided set of projected 2D points to an elliptic curve. The elliptic curve then guides the process of contour completion by iteratively checking neighbor vertices. The implementation of the antipodal points search is done using Matlab and R software. The next section will go through the details and the obtained experimental results.

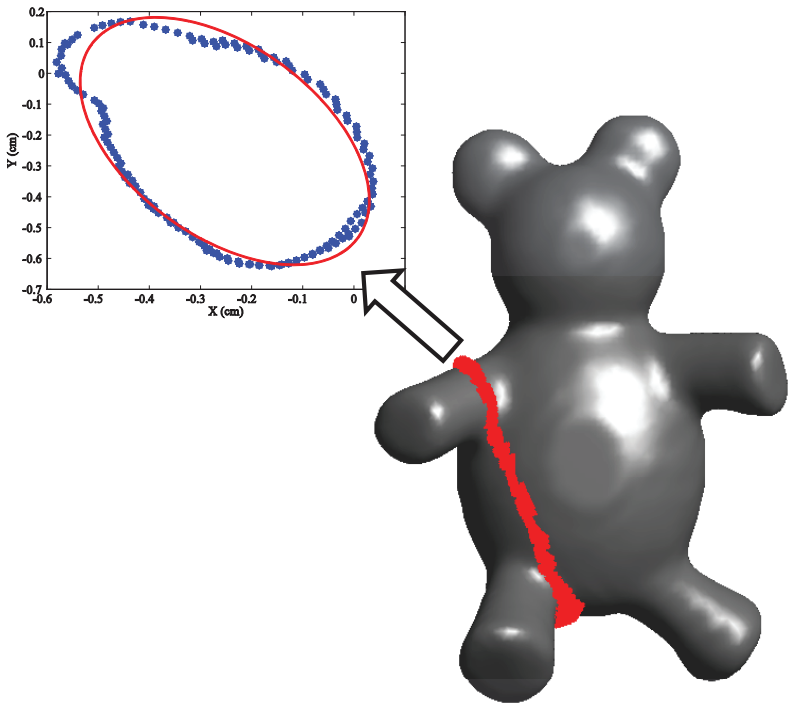


Figure 5.6: Ellipse fit example to guide contour completion.

5.6 Implementation and results of the proposed deformation transfer algorithm

The proposed algorithm is implemented to obtain experimental results. Most of the modules were implemented in Matlab using open source toolboxes for mesh editing. For the generation of 2D footprints, an open source package called Alphahull [90] was used inside the R environment.

The complexity analysis of the algorithm in comparison with related literature approaches is shown in table 5.1. The source of other approaches data is from Van Kaick *et al.* survey [147]. Analysis is done in terms of time and space. The proposed algorithm is competitive in both criteria. An advantage is that mapping first identifies a set of cutline points p which are much less than the number of mesh

Approach	Time	Space
Naive algorithm	$O(m^4n^3 \log n)$	-
Randomized	$O(mn^3 \log n)$	-
Randomized verification	$\approx O(n^3 \log n)$	-
Sets of 4 coplanar points	$O(n^2 + k)$	$O(n)$
Proposed approach	$O(pn \log n)$	$O(n)$

Table 5.1: Complexity analysis of deformation transfer methods for two sets with m and n points. k is the size of the output while p is the number of cutline points.

vertices.

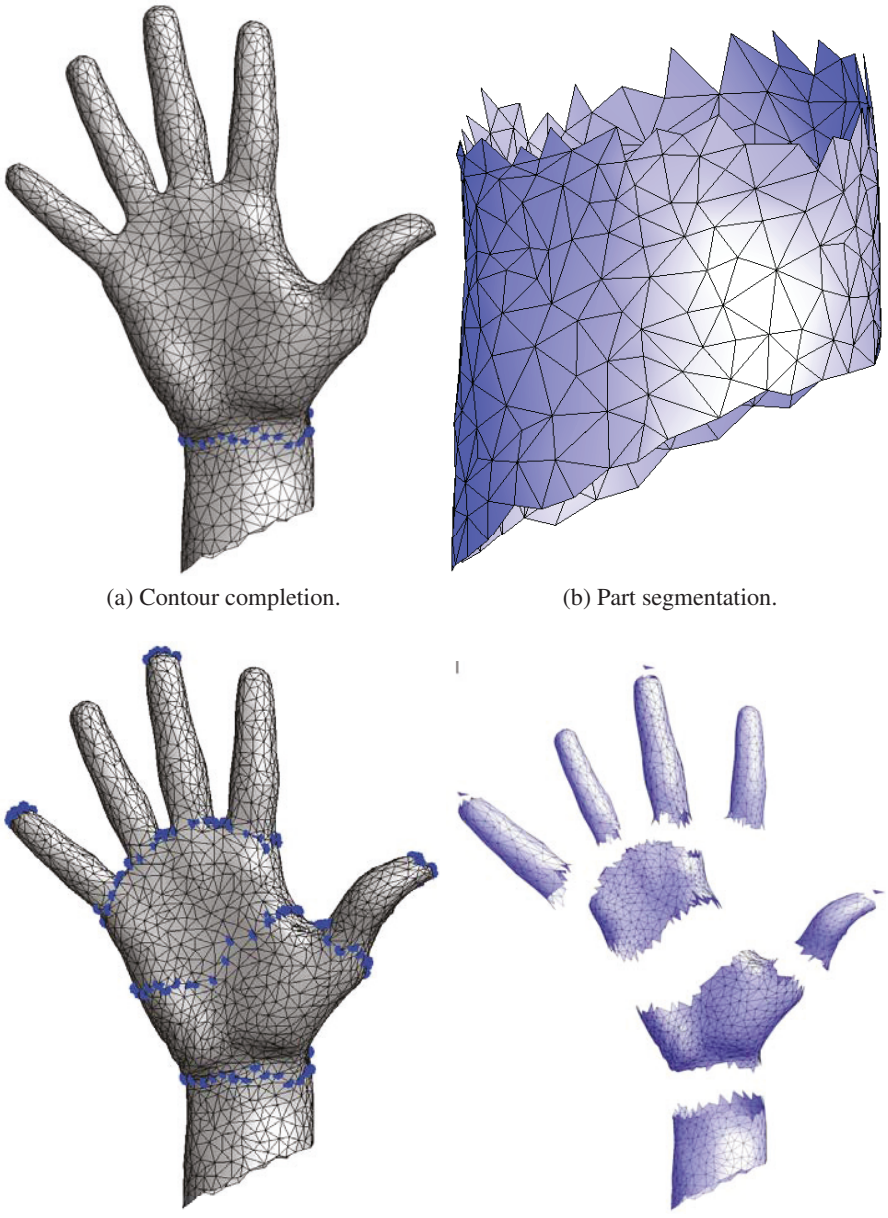
Figure 5.7 shows the 3D segmentation of the models into parts. The advantage of the method is that any form of segmentations are acceptable, given that the hierarchy of the segments is preserved. The resultant 3D sub parts are in simpler convex format.

Figure 5.8 shows the results of correspondence between similar isometric 3D objects. The matching is done through tree matching and 3D registration. This has the advantage of being able to relate sub parts even if similarity exists because the hierarchical relations are preserved. Besides, the sub parts do not need to contain any semantics to be matched.

5.7 Conclusion

A novel algorithm is presented to tackle the problem of data-driven haptic rendering re-use. The data-driven haptic rendering is a resource consuming process and being able to use the resource for a whole class of objects such as livers or lungs is useful for the practicality of data-driven haptic rendering. The algorithm uses concepts from the literature such as deformation transfer, shape correspondence, and 3D mesh segmentation to efficiently transfer the deformation function from one mesh to another where the two exhibit a level of isometry.

The results demonstrated the algorithm ability to process various mesh with various topology and vertices counts. Future directions can be towards the interesting area of deducting the material parameters given a set of similar objects that have the same material. Also concepts from this research can be used generally in other problems in computer graphics such as mesh morphing.



(a) Contour completion.

(b) Part segmentation.

(c) All cutlines.

(d) All parts.

Figure 5.7: The results of a hand mesh segmentation into convex parts.

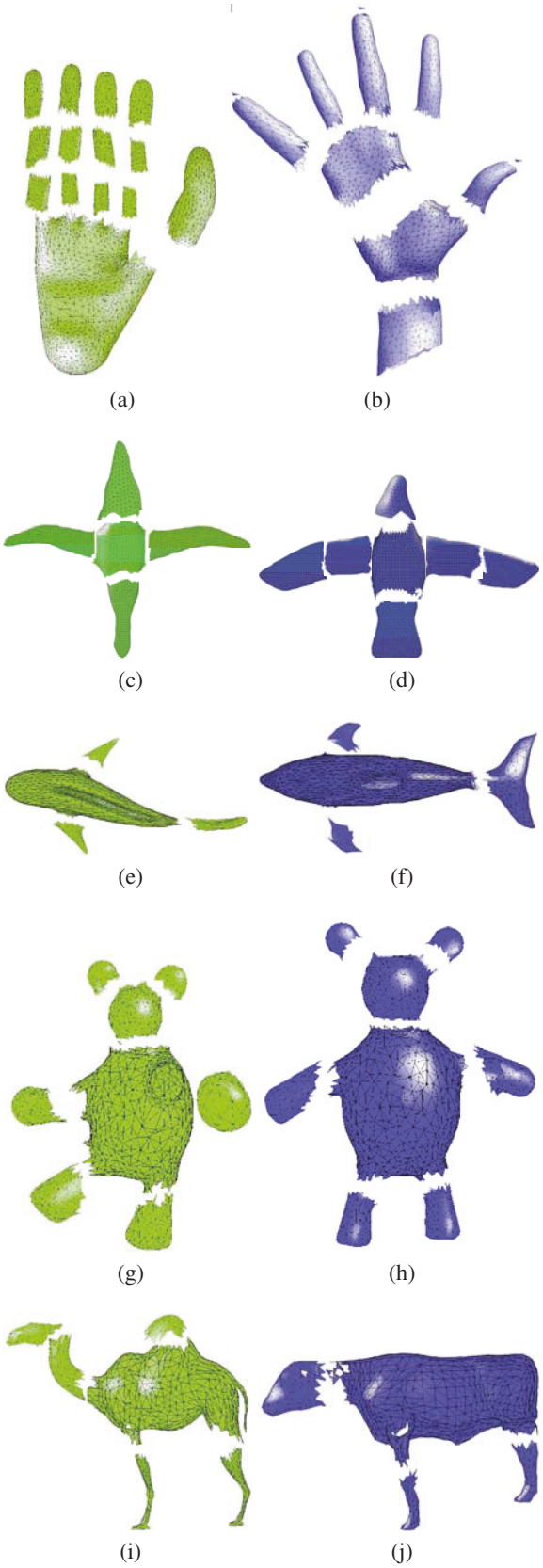


Figure 5.8: Isometric mesh can be matched after being decomposed.

Chapter 6

Study of 3D mesh skeletonisation

Finding the skeleton of a 3D mesh is an essential task for many applications such as mesh animation, tracking, and 3D registration. In recent years, new technologies in computer vision such as Microsoft Kinect have proven that a mesh skeleton can be useful such as in the case of human machine interactions. To calculate the 3D mesh skeleton, the mesh properties such as topology and its components relations are utilized. In this chapter, the usage of a novel algorithm is proposed that can efficiently calculate a vertex antipodal point.

A vertex antipodal point is the diametrically opposite point that belongs to the same mesh. The set of centers of the connecting lines between each vertex and its antipodal point represents the 3D mesh desired skeleton. Post processing is completed for smoothing and fitting centers into optimized skeleton parts. The algorithm is tested on different classes of 3D objects and produced efficient results that are comparable with the literature. The algorithm has the advantages of producing high quality skeletons as it preserves details. This is suitable for applications where the mesh skeleton mapping is required to be kept as much as possible.

6.1 Introduction to 3D mesh skeletonisation process

3D mesh skeleton is best described as a compact representation of the 3D mesh. A formal definition was proposed by Dey and Sun [155] starting with medial axis and using medial geodesic function (MGF). However, later publications [156] proposed that the definition stay open and vary based on the proposed usage and application. Also, the required properties of the skeleton can help in shaping its definition. Properties are listed in [102] and examples of them are robustness, smoothness and centeredness. The 3D mesh skeleton had many applications such as animation, tracking and 3D registration. Figure 6.1 shows an example of a 3D mesh skeleton.

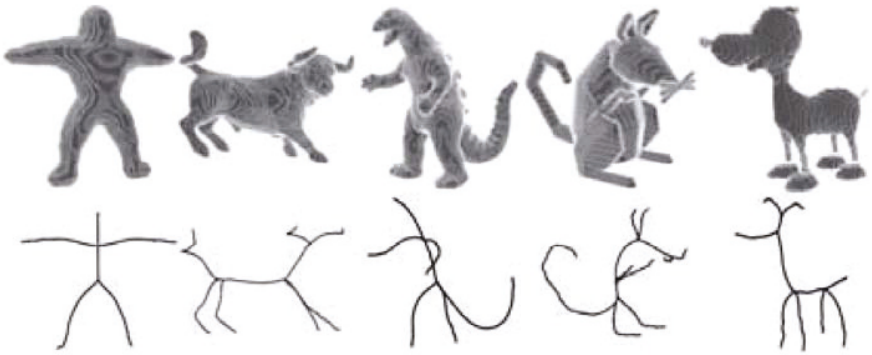


Figure 6.1: Example of skeletons for various 3D objects [102]. The 3D skeleton is a 1D representation of 3D mesh.

The extraction of the 3D mesh skeleton is a challenging process especially when it is required to be autonomous and without supporting inputs from the user. There are several algorithms in the literature that approach the skeleton extraction problem, as will be shown in the related work section. The main drawbacks of the current algorithms are that they are sophisticated in terms of parameters and implementation [157][158] and inaccurate in mesh description [159]. This, when combined with the lack of an ultimate output definition makes it clear that more methodologies are required.

In this study, a novel algorithm that locates antipodal points for mesh vertices is introduced. This algorithm can directly be deployed to locate the set of points that form a 3D mesh skeleton. The algorithm, as shown in the implementation section, transfer the problem into the 2D domain and make use of multiple 2D projections of the 3D mesh in different poses. Once a vertex and its antipodal pair are located the center of their connecting line is added to the skeleton set of points. Post processing is then applied to ensure smoothness and connectivity. The algorithm is efficient due to its autonomy (i.e. no parameters are required from the user side), ability to describe the input 3D mesh, ease of implementation, and possibility of parallelization where faster processing is required.

The rest of this chapter is organized as follows: Section 2 analyzes the related work. Section 3 introduces the proposed algorithm and its related definitions. Section 4 shows the experimental results while Section 5 is dedicated for the concluding remarks and future directions.

6.2 Literature review of 3D mesh skeletonisation techniques

The skeletonisation of the 3D mesh process is an active area of research. The advances in the computer graphics and computer vision domains made the skeleton even more important as an excellent representation for processing and transmission purposes. Thus, there are many algorithms that targeted the skeletonisation problem and they can be classified in different ways.

A relatively recent survey is by Cornea and Min [102] focused on listing the mesh skeleton desired properties and available algorithms. These properties can be summarized as follows:

1. Homotopic: The skeleton needs to preserve the mesh topology [160].
2. Connected: The skeleton needs to be connected as long as the mesh is connected.
3. Invariant under isometric transformations: $T(Sk(O)) = Sk(T(O))$ [102], where $T(O)$ is a transformation function and $Sk(O)$ is the object O skeleton.
4. Reconstruction: The mesh can be reconstructed from the skeleton [161].
5. Thinness: an opposite feature to reconstruction, which requires that the object be as thin as possible.
6. Centeredness: The skeleton needs to be central with respect to the mesh [162].
7. Reliability: All the mesh points need to be visible from at least one of the skeleton points [163].
8. Smoothness: The variation in tangent directions of the skeleton points needs to be as low as possible.
9. Component-wise differentiation: A skeleton can lead to mesh recognized segments.
10. Robustness: The skeleton sensitivity to noise in mesh boundaries need to be low.
11. Hierarchical: A skeleton needs to reflect the mesh hierarchy [164].

Cornea and Min [102] also classified the available algorithms into four categories: Thinning and boundary propagation, using a distance field, geometric methods, and general field functions. The thinning algorithms remove the object layers or reduce its boundaries iteratively until it reaches the required skeleton

[165]. Using a distance field is the second category where a distance field function $D(P)_{P\in O} = \min_{Q\in B(O)}(d(P,Q))$ [166]. P is an interior point in a voxelized representation of an object O, $B(O)$ is the boundary of O, and $d(.,.)$ is a metric distance function. There is also geometric methods that use approaches such as Voronoi diagrams [167], cores and M-reps [168], and Reeb graphs [169]. Finally other functions than the distance field function were used such as potential field [170], electrostatic field [171], visible repulsive force [172], and radial basis function [173].

The proposed algorithm belongs to the fourth category. A novel function is used, which depends on the antipodal point location algorithm. The algorithm operates on boundary triangular mesh or point clouds. The newer function is intuitive and satisfies the above desired properties efficiently.

6.3 The proposed 3D mesh skeletonisation algorithm

The proposed algorithm main idea is to use an antipodal location algorithm to locate the skeleton composing set of points. The algorithm takes a point p that belongs to the mesh M and returns another point $q \in M$, which is the closest to be antipodal point to p. This is a search problem in the set of M vertices. The search is guided through a set of heuristics that limit the search and evaluate each candidate.

6.3.1 Preliminaries

A challenge in the underlying problem is the lack of formal definitions. Unlike the medial axis of a mesh, the 3D mesh skeleton is not properly defined in the literature [156]. Also, an antipodal point of another point is not well defined in 3D domain [7]. However, as shown in previous sections, certain properties and heuristics exist that can differentiate a plausible output.

The definition used in this study, is that a 3D mesh skeleton $S = \{v_1, v_1, ..., v_n\}$ is formed by a set of finite points that should each meet the following constraints:

- $v_i, p, q \in \mathbf{P}$
- $\min\{dist(v_i, p) - dist(v_i, q)\}$

where p and q form a pair of antipodal points that belong to the boundary of a mesh M. $\mathbf{P}$ is a plane formed by a pair of antipodal points and v_i.

Another supporting definition is the antipodal point definition in 3D. A linguistic definition is that the antipodal point needs to be diametrically opposite [100]. We add to that the line between the two points need to be totally inside the mesh. This is important in order to avoid out of mesh antipodal points as the 3D mesh is not always totally convex. Figure 6.2 shows the used definitions.

The 3D skeleton extraction algorithm can be split into three main parts: preprocessing, main module, and post processing. The three parts perform the required transformation in a serial manner to produce the final output. This modularity in the algorithm enables easy changes and improvement if specific application requirements are desired.

6.3.2 Preprocessing

The preprocessing phase aims to transform the inputs into a suitable format for the next phases. The input here is in the form of a 3D triangular boundary mesh. The mesh is required to be closed and of a high resolution. The closed boundary is naturally required to avoid outliers and undefined calculations. High resolution is also required, as will be seen in the next phase, because 2D points' projections is calculated. High resolution helps in identifying the enclosing footprints of the projections, which is important in locating the antipodal points.

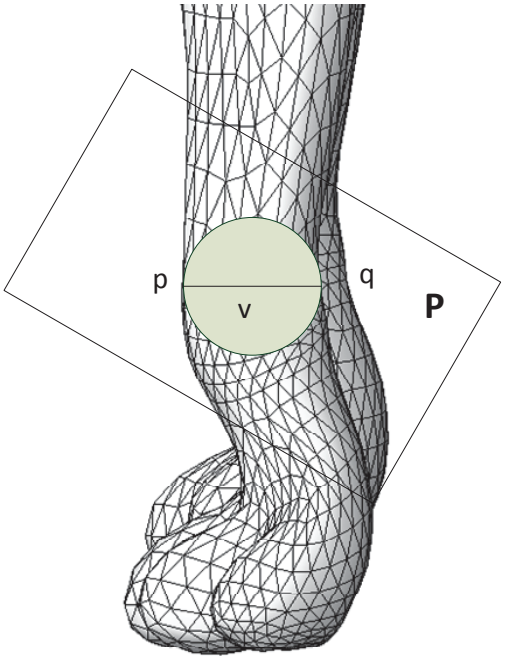


Figure 6.2: Antipodal point and skeleton points definition. p is the source, q is the located antipodal point, and v is the center of the line connecting p and q. All the three points belongs to the plane P.

In case the mesh is of a low resolution, remeshing techniques can be utilized or the faces can be subdivided iteratively until an acceptable threshold area is met. The search for antipodal points is done exhaustively for all the mesh points. However, if faster processing is required, then random seeds can be calculated first that can guarantee a level of coverage and the results are then oversampled in the post processing phase. This is important for real time processing because the algorithm can utilize parallel architectures.

6.3.3 Main module of the skeletonisation algorithm

The main core of the 3D skeleton extraction algorithm is to define an antipodal point for another point, where both belong to the mesh boundary. An antipodal point is

diametrically opposite. In order to define the diameter, a tangent plane to the source point needs to be defined. The problem in 3D mesh is that a point is formed as a result of the intersection of multiple faces. These faces do not, in general, share a common plane. Thus, acquiring this plane needs more information on the global topology of the mesh. Figure 6.3 shows the data flow diagram of computing the 2D and 3D antipodal points.

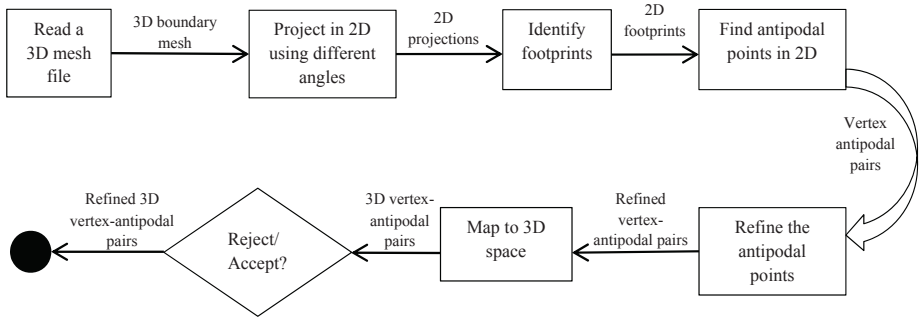


Figure 6.3: Data flow diagram of the algorithm steps to compute the 2D and 3D antipodal points.

In the 2D domain, the situation is easier. A point is a member of a curve, and therefore a tangent can easily be acquired. Thus, the problem is transferred to the 2D domain. Multiple projections are calculated by rotating the mesh around x, y, and z axes with fixed small steps. The 2D footprint [83] of each projection is calculated to separate the boundary points only. The resultant discrete boundary points' tangents are then calculated for every projection to get the antipodal point for each boundary point.

The result of the previous step is that every 3D point has multiple antipodal points' candidates. These candidates are filtered to only one that satisfies certain conditions. The first one is that the line connecting the antipodal point and the source point needs to be entirely inside the mesh. The second one then, which is

then tested after the first one, is that it is the farthest point from the source. This is required to ensure that point is with the largest diameter. To check that the line is entirely inside the mesh, an algorithm called point in polygon [109] is used. This extends the line from both sides and makes sure that every entry to the mesh is matched with an exit. If the original line two end points are an exit followed by an entry, then it is rejected.

Once every point has a corresponding antipodal one, the skeleton can be then defined. The skeleton is formed as a result of connecting all the centers of the lines connecting a point with its antipodal one. However, this requires post processing to make sure it has the desired properties mentioned in the literature review section.

The last step in the proposed algorithm is to filter the extracted skeleton. A skeleton needs to have general properties. The most important properties that require post processing are thinness, connectivity and centeredness. The resulting points from the last step form a thick and unconnected set of points. Therefore, a line fitting algorithm is required to induct proper lines out of these points. This makes sure that if some outliers exist they cannot have a large effect on the end result.

6.4 Experimental results of the proposed 3D mesh skeletonisation algorithm

The algorithm was tested on several 3D object classes that were acquired from Chen's public database [6]. The database was chosen as it was used for benchmarking 3D mesh segmentation, which is a close field to 3D skeletonisation. Currently, 3D mesh skeletons are subjective and future directions are towards benchmarking results against human samples as demonstrated in Chen's database [6].

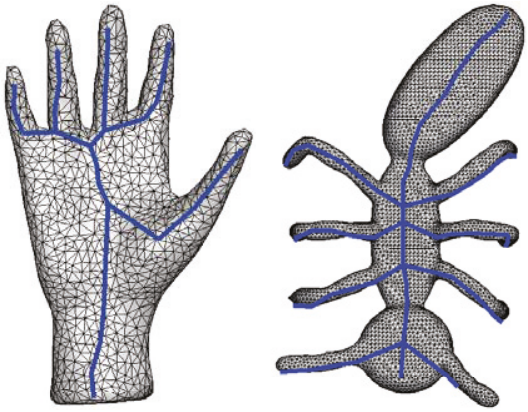
The algorithm is run offline and the input was the boundary mesh files in simple formats such as wavefront object format. Another optional input is the rotation step around the x, y, and z axes of the projections. This affects the amount of calculated antipodal points and hence the final fitting quality. It also affects the algorithm running time. Thus, there is a tradeoff between accuracy and time. However, this is generic for any object and can be fixed as the algorithms are meant to be run offline.

The results are shown in Figure 6.4. There is no specific benchmarks in the literature yet [102] as a result of the lack in formal definition of the 3D mesh skeleton itself. However, the extracted 3D mesh skeletons follow the desired properties mentioned in the literature review section and are comparable with the literature results. The used rotation step is 10 degrees in each direction.

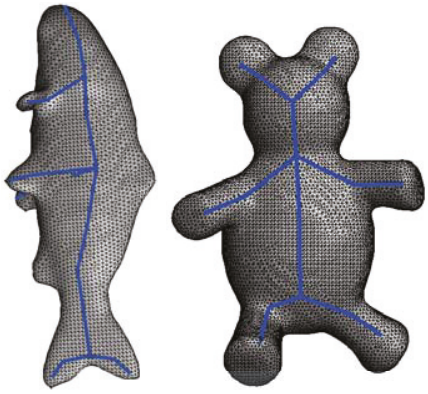
Figure 6.5 shows a comparison between the proposed algorithm results and literature algorithms for a 3D hand model. The literature algorithm results are acquired from [27]. The proposed algorithm is superior in terms of quality and desired properties. Besides, the proposed algorithm is astomous and operates on 3D boundary mesh input format. These features are not always available in corresponding literature algorithms.

6.5 Conclusion

The 3D skeleton extraction is a challenging problem in computer graphics. This chapter provides an algorithm that can calculate the 3D mesh skeleton efficiently by individually locating the antipodal point of every boundary point in the mesh. The centers of the lines connecting the antipodal point and the source are augmented and fitted to produce the desired skeleton. The algorithm has been tested on different



(a) Hand skeleton. (b) Ant skeleton.



(c) Fish skeleton. (d) Teddy skeleton.

Figure 6.4: Skeleton results for selected 3D objects classes with a different vertices count. The skeletons have the 11 proper properties defined in section 2.

objects with different classes and vertices count. The results follow the proper properties that a skeleton needs to have. Algorithm weakness is in the case of concave 3D mesh such as a a ball or a donut. Future research can be directed towards testing the proposed algorithm on parallel architecture using the advances in modern graphical processing units. Also, the 3D mesh skeleton benchmarks require further investigations following the path of similar domains such as 3D mesh segmentation.

(a) Proposed approach. (b) Mesh contraction. (c) Reeb graph.

(d) Potential field. (e) Distance field. (f) Thinning.

(g) Medial surface.

Figure 6.5: Skeleton results for the hand 3D object when compared with literature methods.

Chapter 7

Conclusion

This thesis provided a novel automatic algorithm to calculate 3D mesh segmentation. The algorithm has other advantages such as the ability to perform on different criterion, modularity and being ready for parallelization on multi core processors and architectures. In this chapter we will state the concluding remarks and possible future extensions.

7.1 Thesis Contributions

The proposed algorithm and heuristics were applied on several 3D object classes and also deployed in detailed applications. Several topics were addressed and they can be summarized as follows:

3D Vertex antipodal point location: An antipodal of a boundary vertex is the diametrically opposite vertex that belongs to the same 2D shape or 3D mesh boundary. Locating the antipodal point in the 3D case is challenging [7]. This thesis proposes a novel algorithm to locate the 3D antipodal of a boundary vertex. The algorithm searches a set of 2D footprints of different 2D

projections of the mesh to ensure coverage. Individual antipodal points can be located concurrently and hence the algorithm can be implemented on parallel architectures. Besides, the provided algorithm helped in the definition of a 3D antipodal point as this definition is still not final in the case of 3D boundary mesh.

Parameterless 3D mesh segmentation: Most of the 3D mesh segmentation literature methods require a parameter or more to perform. These parameters are not efficient when they are dependent on the input mesh such as the number of required segmentations. The proposed algorithm is practically parameterless and any additional parameter is optional to match the subjective nature of the 3D mesh segmentation process. Different users would produce different segmentations for the same input mesh. This can be the result of oversegmentation or the opposite of being extra cautious in laying cutlines. Thus, the proposed algorithm considers the segmentation criterion as an optional input. The default is the widely accepted minima rule, but any other criterion can be used without any modifications to the algorithm core modules.

Variable segmentation criterion: The segmentation criterion is base on which a segment is split from the main mesh. In our algorithm, this is an optional input and the default is the minima rule. Making the segmentation criterion variable is an important advantage of the proposed algorithm. This allows the algorithm to be deployed in a wide range of applications. Future commercial applications of the technology will always favor such generalization.

Real time 3D mesh segmentation: The proposed algorithm can be implemented on parallel architectures as its main modules can be run independently. This allows the algorithm to be used for online application. Online processing is

vital for many applications such as in robotics. A great aid for a robot is to be able to interactively structure its surrounding environment.

The proposed algorithm is efficient in terms of results, autonomy, speed and flexibility. The 3D mesh segmentation is relatively a new domain and many features are still under development. Having generality and adaptability in mind in the early stages would lead to better designs and more robust applications.

In terms of limitations, the proposed algorithm is still an offline one if implemented in single core hardware. The algorithm is also not pose invariant. Thus, a closed hand mode would be treated differently to an open one. However, some applications consider this as an advantage because the segmentation process has a large dependency on the application requirements and definitions.

7.2 Potential applications

There are several potential applications that can make use of the 3D mesh segmentation algorithm output. Segmentation is an efficient methodology of dividing the problem into simpler and smaller parts. Here are some examples:

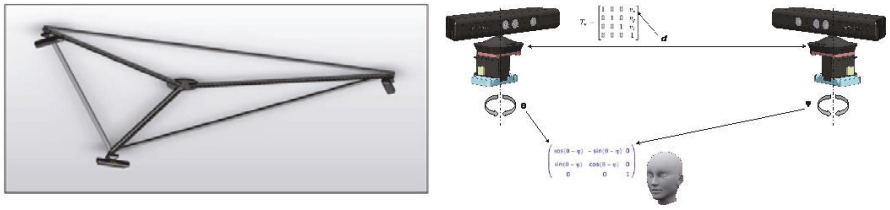
7.2.1 Online generic 3D skeletonisation using depth cameras

Depth cameras have become popular in recent years. They have become available to end users with the introduction of Microsoft Kinect [174]. Since then several hardware and software were developed for such technology. The technology allows the user to obtain the 3D point cloud of the environment. This is then can be used to infer the depth of the objects and reconstruct 3D parts.

So far, the technology is affected by the gaming industry and hence the focus is on players. Most of the devices provide 3D skeletons of the human players so the

games can sense their actions in a markerless sense. This is however quite limited for only one class of objects, which is the human operator.

The proposed algorithm can help in moving this technology to the next level. The target is to have an online skeleton of any separable object in front of the screen. Players can then use these objects in their games or even the sensors can be used by robotics to recognize and interact with surroundings. To obtain a complete 3D point cloud rather than a 2.5D one, the system can use multiple sensors such as the structure shown in Figure 7.1a [175]. Stereo vision and camera calibration algorithms can then fuse the resultant point clouds to obtain a complete 3D mesh. This is shown in Figure 7.1b [175] and similar structure was demonstrated in [176].



(a) Multi Microsoft Kinect system that captures full 3D.

(b) Calibration of multipe Microsoft Kinect cameras.

Figure 7.1: Acquisition system of full 3D point cloud using multiple depth sensors.

Having a generic online skeletonisation device, will be of great use to many applications. This includes, but not limited to, biometrics, robotics and the gaming industry.

7.2.2 Shape correspondence for data-driven haptic simulation of deformable models

Haptic simulation of deformable models is an interesting point of research as it facilitates a new dimension in human machine interactions. The simulation allows the user to touch and feel the material of virtual reality objects and be able to identify the

material properties. This requires the system to calculate the external force exerted by the user touch and calculate the visual and haptic deformations accordingly.

Many algorithms have been proposed in this area. They can be broadly classified into two main categories: parametric and data-driven [177]. Parametric methods such as finite element methods (FEM) are efficient in terms of accuracy. However, they cannot be used in real-time applications. On the other hand, data-driven methods [137] collect data from multiple interactions with the object through proper sensors and build the model through machine learning techniques. This model can then be used quickly in real-time applications.

The main problem with data-driven methods is the data collection phase. The process is time consuming and is valid only for one object. An important addition to such systems is the ability to generalize the collected data for similarly structures objects. This reduces the collection time from one object to a whole class of similar objects. Figure 7.2 shows an example of two liver objects that are quite similar. Data can be re-used for the similar areas and the data can be collected only for the different part.

(a) Liver 1. (b) Liver 2.

Figure 7.2: Shape correspondence can identify the areas of similarities and differences. The data collection can be only done for the different areas.

The proposed algorithm in this thesis can identify the similarities and differences as the mesh will be decomposed into smaller parts.

7.3 Future research directions

The 3D mesh segmentation field still requires further research. This can be easily verified for instance when we compare the amount of publications against 2D mesh segmentation field for instance. The ultimate goal is to have smart segmentation that is real-time, adaptive, and accurate at the same time. The main future directions can be summarized as follows:

Improved 3D mesh segmentation benchmarking database: Currently, the available benchmarking database is generated subjectively by human users. This requires a lot of effort to cover different classes of 3D objects and also it might not cover all cases. A future direction is to research on how these databases can be generated automatically. This requires a better understanding of the human approach in 3D segmentation.

Assisted 3D mesh segmentation: Online databases of images are now widely used by end users. An interesting concept is to use pattern recognition and matching to suggest segmentations based on the collected results and meta data. This can give a boost to the segmentation results as the algorithm will start with better input. Collective intelligence methods can then be used to build special purpose databases for the 3D segmentation process.

Improved metrics for algorithm comparison: The 3D mesh segmentation algorithms metrics are still in the early stages of development. More metrics are required to make it easier and clearer to distinguish between different algorithms.

A new 3D mesh representation that is easily separable: This requires a real paradigm

shift. However, if meta data could be embedded in the mesh formulation process, it would make the digitization process much easier.

References

[1] S. Shlafman, A. Tal, and S. Katz, "Metamorphosis of polyhedral surfaces using decomposition," in *Computer Graphics Forum*, vol. 21, pp. 219–228, 2002.

[2] Y.-K. Lai, Q.-Y. Zhou, S.-M. Hu, and R. R. Martin, "Feature sensitive mesh segmentation," in *Proceedings of the ACM symposium on Solid and physical modeling*, SPM '06, pp. 17–25, ACM, 2006.

[3] A. Shamir, "A survey on mesh segmentation techniques," *Computer Graphics Forum*, vol. 27, no. 6, pp. 1539–1556, 2008.

[4] A. Shamir, "A formulation of boundary mesh segmentation," in *3D Data Processing, Visualization and Transmission, 3DPVT 2004. Proceedings. 2nd International Symposium on*, pp. 82–89, IEEE, 2004.

[5] J. Schmid, J. A. I. Guitián, E. Gobbetti, and N. Magnenat-Thalmann, "A gpu framework for parallel segmentation of volumetric images using discrete deformable models," *The Visual Computer*, vol. 27, no. 2, pp. 85–95, 2011.

[6] X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3d mesh segmentation," *ACM Transaction on Graphics*, vol. 28, no. 3, pp. 1–12, 2009.

[7] L. Shapira, A. Shamir, and D. Cohen-Or, "Consistent mesh partitioning and skeletonisation using the shape diameter function," *Visual Computer*, vol. 24, no. 4, pp. 249–259, 2008.

[8] Y. Lee and Y. Kwak, "3d content industry in korea: Present conditions and future development strategies," *Communications in Computer and Information Science*, vol. 184 CCIS, no. PART 1, pp. 358–363, 2011.

[9] B. Chazelle, D. P. Dobkin, N. Shouraboura, and A. Tal, "Strategies for polyhedral surface decomposition: An experimental study," *Computational Geometry*, vol. 7, no. 5-6, pp. 327 – 342, 1997.

[10] D. Johnson and M. Garey, *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman, San Francisco, 1979.

[11] J. Peng, C.-S. Kim, and C.-C. J. Kuo, "Technologies for 3d mesh compression: A survey," *Journal of Visual Communication and Image Representation*, vol. 16, no. 6, pp. 688 – 733, 2005.

[12] C. Zhang and T. Chen, "Efficient feature extraction for 2d/3d objects in mesh representation," in *Proceeding on International Conference of Image Processing*, vol. 3, pp. 935–938, IEEE, 2001.

[13] A. Golovinskiy and T. Funkhouser, "Randomized cuts for 3d mesh analysis," *ACM Transactions on Graphics*, vol. 27, no. 5, pp. 1–12, 2008.

[14] Y. Zheng, A. Barbu, B. Georgescu, M. Scheuering, and D. Comaniciu, "Fast automatic heart chamber segmentation from 3d ct data using marginal space learning and steerable features," in *IEEE 11th International Conference on Computer Vision, ICCV 2007*, pp. 1–8, IEEE, 2007.

[15] S. Pan and B. Dawant, "Automatic 3d segmentation of the liver from abdominal ct images: a level-set approach," in *Proceedings of SPIE*, vol. 4322, pp. 128–138, 2001.

[16] C. Whlby, I.-M. SINTORN, F. Erlandsson, G. Borgefors, and E. Bengtsson, "Combining intensity, edge and shape information for 2d and 3d segmentation of cell nuclei in tissue sections," *Journal of Microscopy*, vol. 215, no. 1, pp. 67–76, 2004.

[17] D. D. Hoffman and W. A. Richards, "Parts of recognition," *Cognition*, vol. 18, no. 1-3, pp. 65 – 96, 1984.

[18] L. Lu, Y.-K. Choi, W. Wang, and M.-S. Kim, "Variational 3d shape segmentation for bounding volume computation," *Computer Graphics Forum*, vol. 26, no. 3, pp. 329–338, 2007.

[19] B. M. Chazelle, "Convex decompositions of polyhedra," in *Proceedings of the thirteenth annual ACM symposium on Theory of computing*, pp. 70–79, ACM, 1981.

[20] P. Heckbert, "Survey of texture mapping," *IEEE Computer Graphics and Applications*, vol. 6, no. 11, pp. 56–67, 1986.

[21] P. Alliez, M. Meyer, and M. Desbrun, "Interactive geometry remeshing," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3, pp. 347–354, 2002.

[22] P. Cignoni, C. Montani, and R. Scopigno, "A comparison of mesh simplification algorithms," *Computers and Graphics*, vol. 22, no. 1, pp. 37–54, 1998.

[23] S. Gumhold, S. Guthe, and W. Strasser, "Tetrahedral mesh compression with the cut-border machine," in *Proceedings of the conference on Visualization*, pp. 51–58, IEEE, 1999.

[24] M. Alexa, "Recent advances in mesh morphing," *Computer Graphics Forum*, vol. 21, no. 2, pp. 173–198, 2002.

[25] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, vol. 24, no. 4, pp. 509–522, 2002.

[26] T. Matsuyama, X. Wu, T. Takai, and S. Nobuhara, "Real-time 3d shape reconstruction, dynamic 3d mesh deformation, and high fidelity visualization for 3d video," *Computer Vision and Image Understanding*, vol. 96, no. 3, pp. 393–434, 2004.

[27] O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee, "Skeleton extraction by mesh contraction," *ACM Transaction on Graphics*, vol. 27, no. 3, pp. 1–10, 2008.

[28] M. Gissler, R. Schmedding, and M. Teschner, "Time-critical collision handling for deformable modeling," *Computer Animation and Virtual Worlds*, vol. 20, no. 2-3, pp. 355–364, 2009.

[29] E. Jang, "3d animation coding: its history and framework," in *IEEE International Conference on Multimedia and Expo-ICME*, vol. 2, pp. 1119–1122, IEEE, 2000.

[30] M. Meng, L. Fan, and L. Liu, "A comparative evaluation of foreground/background sketch-based mesh segmentation algorithms," *Computers & Graphics*, vol. 35, no. 3, pp. 650 – 660, 2011.

[31] M. Attene, S. Katz, M. Mortara, G. Patane, M. Spagnuolo, and A. Tal, "Mesh segmentation - a comparative study," in *Proceedings of the IEEE International Conference on Shape Modeling and Applications*, pp. 7–7, IEEE Computer Society, 2006.

[32] A. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis, and P. Azariadis, "3d mesh segmentation methodologies for cad applications," *Computer-Aided Design and Applications*, vol. 4, no. 6, pp. 827–841, 2007.

[33] H. Benhabiles, J.-P. Vandeborre, G. Lavou, and M. Daoudi, "A comparative study of existing metrics for 3d-mesh segmentation evaluation," *Visual Computer*, vol. 26, no. 12, pp. 1451–1466, 2010.

[34] G. Lavou, J. P. Vandeborre, H. Benhabiles, M. Daoudi, K. Huebner, M. Mortara, and M. Spagnuolo, "Shrec'12 track: 3d mesh segmentation," in *Proceedings of the 5th Eurographics conference on 3D Object Retrieval* (Eurographics, ed.), pp. 93–99, 2012.

[35] Z. Ji, L. Liu, Z. Chen, and G. Wang, "Easy mesh cutting," *Computer Graphic Forum*, vol. 25, no. 3, pp. 283–291, 2006.

[36] H. Wu, C. Pan, J. Pan, Q. Yang, and S. Ma, "A sketch-based interactive framework for real-time mesh segmentation," in *Computer Graphics International*, 2007.

[37] M. Giaquinta, G. Modica, and Sou, "The dirichlet energy of mappings with values into the sphere," *manuscripta mathematica*, vol. 65, no. 4, pp. 489–507, 1989.

[38] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin, "Fast mesh segmentation using random walks," in *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling 2008, SPM'08*, pp. 183–191, ACM, 2008.

[39] C. Xiao, H. Fu, and C.-L. Tai, "Hierarchical aggregation for efficient shape extraction," *Visusl Computer*, vol. 25, no. 3, pp. 267–278, 2009.

[40] L. Papaleo and L. De Floriani, "Manual segmentation and semantic-based hierarchical tagging of 3d models," in *Eurographics Italian Chapter Conference*, pp. 25–32, The Eurographics Association., 2010.

[41] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin, "Modeling by example," *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 652–663, 2004.

[42] S. Brown, B. Morse, and W. Barrett, "Interactive part selection for mesh and point models using hierarchical graph-cut partitioning," in *Proceedings of Graphics Interface*, pp. 23–30, Canadian Information Processing Society, 2009.

[43] L. Fan, L. Lic, and K. Liu, "Paint mesh cutting," *Computer Graphics Forum*, vol. 30, no. 2, pp. 603–612, 2011.

[44] K. Shimada and D. Gossard, "Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing," in *Proceedings of the third ACM symposium on Solid modeling and applications*, pp. 409–419, ACM, 1995.

[45] Y. Zheng and C.-L. Tai, "Mesh decomposition with cross-boundary brushes," *Computer Graphics Forum*, vol. 29, no. 2, pp. 527–535, 2010.

[46] A. E. Lefohn, J. E. Cates, and R. T. Whitaker, "Interactive, gpu-based level sets for 3d segmentation," in *Medical Image Computing and Computer-Assisted Intervention-MICCAI 2003*, pp. 564–572, Springer, 2003.

[47] Y. Zheng, C.-L. Tai, and O. K.-C. Au, "Dot scissor: A single-click interface for mesh segmentation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 8, pp. 1304–1312, 2012.

[48] M. Attene, B. Falcidieno, and M. Spagnuolo, "Hierarchical mesh segmentation based on fitting primitives," *The Visual Computer*, vol. 22, no. 3, pp. 181–193, 2006.

[49] A. Kalvin and R. Taylor, "Superfaces: polygonal mesh simplification with bounded error," *IEEE Computer Graphics and Applications*, vol. 16, no. 3, pp. 64 –77, 1996.

[50] G. Lavou, F. Dupont, and A. Baskurt, "A new cad mesh segmentation method, based on curvature tensor analysis," *Computer-Aided Design*, vol. 37, no. 10, pp. 975 – 987, 2005.

[51] V. K. D. J. A. Sheffer, "Shuffler: Modeling with interchangeable parts," *Visual Computer journal*, 2007.

[52] X. Zhang, G. Li, Y. Xiong, and F. He, "3d mesh segmentation using mean-shifted curvature," 2008.

[53] B. Lvy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3, pp. 362–371, 2002.

[54] A. P. Mangan and R. T. Whitaker, "Partitioning 3d surface meshes using watershed segmentation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 308–321, 1999.

[55] Y. Zhou and Z. Huang, "Decomposing polygon meshes by means of critical points," in *10th International Conference of Multimedia Modelling*, pp. 187–195, IEEE, 2004.

[56] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[57] K. Wu and M. Levine, "3d part segmentation using simulated electrical charge distributions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 11, pp. 1223–1235, 1997.

[58] Y. Sun, D. L. Page, J. K. Paik, A. Koschan, and M. A. Abidi, "Triangle mesh-based edge detection and its application to surface segmentation and adaptive surface smoothing," in *IEEE International Conference on Image Processing*, pp. 825–828, 2002.

[59] D. PAGE, A. KOSCHAN, and M. ABIDI, "Perception-based 3d triangle mesh segmentation using fast marching watersheds," in *IEEE Conference on Computer Vision and Pattern Recognition.*, vol. 2, pp. 27–32, IEEE Computer Society, 2003.

[60] A. Sheffer, "Model simplification for meshing using face clustering," *Computer-Aided Design*, vol. 33, no. 13, pp. 925 – 934, 2001.

[61] N. Gelfand and L. J. Guibas, "Shape segmentation using local slippage analysis," in *Proceedings of the Eurographics/ACM Siggraph symposium on Geometry processing SGP '04:*, pp. 214–223, ACM, 2004.

[62] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images," in *Eighth IEEE International conference on Computer Vision, ICCV 2001*, vol. 1, pp. 105 –112, 2001.

[63] S. Katz and A. Tal, "Hierarchical mesh decomposition using fuzzy clustering and cuts," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 954–961, 2003.

[64] J.-M. Lien, J. Keyser, and N. M. Amato, "Simultaneous shape decomposition and skeletonization," in *Proceedings of the ACM symposium on Solid and physical modeling SPM '06*, pp. 219–228, ACM, 2006.

[65] X. Li, T. W. Woon, T. S. Tan, and Z. Huang, "Decomposing polygon meshes for interactive applications," in *Proceedings of the 2001 symposium on Interactive 3D graphics*, pp. 35–42, ACM, 2001.

[66] R. Raab, C. Gotsman, and A. Sheffer, "Virtual woodwork: Generating bead figures from 3d models," *International Journal on Shape Modeling*, vol. 10, no. 1, pp. 1–30, 2004.

[67] X. Gu, S. Gortler, and H. Hoppe, "Geometry images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 355–361, 2002.

[68] I. M. Boier-Martin, "Domain decomposition for multiresolution analysis," in *SGP '03: Proceedings of the 2003 Eurographics/ACM Siggraph symposium on Geometry processing*, pp. 31–40, Eurographics Association, 2003.

[69] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 2006.

[70] J. Wu and L. Kobbelt, "Structure recovery via hybrid variational surface approximation," *Computer Graphics Forum*, vol. 24, no. 3, pp. 277–284, 2005.

[71] D. Julius, V. Kraevoy, and A. Sheffer, "D-charts: Quasi-developable mesh segmentation," *Computer Graphics Forum*, vol. 24, no. 3, pp. 581–590, 2005.

[72] I. Shatz, A. Tal, and G. Leifman, "Paper craft models from meshes," *The Visual Computer.*, vol. 22, no. 9, pp. 825–834, 2006.

[73] B. Hendrickson and R. Leland, "An improved spectral graph partitioning algorithm for mapping parallel computations," *SIAM Journal on Scientific Computing*, vol. 16, no. 2, pp. 452–469, 1995.

[74] R. Liu and H. Zhang, "Mesh segmentation via spectral embedding and contour analysis," *Computer Graphics Forum (Special Issue of Eurographics 2007)*, vol. 26, no. 3, pp. 385–394, 2007.

[75] R. Liu and H. Zhang, "Segmentation of 3d meshes through spectral clustering," in *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, pp. 298–305, IEEE, 2004.

[76] J. Zhang, J. Zheng, C. Wu, and J. Cai, "Variational mesh decomposition," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 3, pp. 21:1–21:14, 2012.

[77] H. Benhabiles, J.-P. Vandeborre, G. Lavou, and M. Daoudi, "A framework for the objective evaluation of segmentation algorithms using a ground-truth of human segmented 3d models," in *IEEE International Conference on Shape Modeling and Applications*, no. 36-43, IEEE, 2009.

[78] S. Berretti, A. D. Bimbo, and P. Pala, "3d mesh decomposition using reeb graphs," *Image and Vision Computing*, vol. 27, no. 10, pp. 1540–1554, 2009.

[79] R. Unnikrishnan, C. Pantofaru, and M. Hebert, "Toward objective evaluation of image segmentation algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 929–944, 2007.

[80] T. Han and T. Abdelrahman, "hicuda: High-level gpgpu programming," *IEEE Transactions on Parallel and Distributed Systems,*, vol. 22, no. 1, pp. 78–90, 2011.

[81] J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, and J. Phillips, "Gpu computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[82] T. Zaharia and F. Preteux, "Shape-based retrieval of 3d mesh models," in *IEEE International Conference on Multimedia and Expo ICME'02*, vol. 1, pp. 437–440, IEEE, 2002.

[83] M. Dupenois and A. Galton, "Assigning footprints to dot sets: An analytical survey," in *Spatial Information Theory* (K. Hornsby, C. Claramunt, M. Denis, and G. Ligozat, eds.), vol. 5756 of *Lecture Notes in Computer Science*, pp. 227–244, Springer Berlin Heidelberg, 2009.

[84] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Transactions on Information Theory*, vol. 29, no. 4, pp. 551–559, 1983.

[85] H. Alani, C. Jones, and D. Tudhope, "Voronoi-based region approximation for geographical information retrieval with gazetteers," *International Journal of Geographical Information Science*, vol. 15, no. 4, pp. 287–306, 2001.

[86] A. Galton and M. Duckham, "What is the region occupied by a set of points?," *Geographic, Information Science*, pp. 81–98, 2006.

[87] A. Moreira and M. Santos, "Concave hull: a k-nearest neighbours approach for the computation of the region occupied by a set of points," in *Grapp 2007: Proceedings of the Second International Conference on Computer Graphics Theory and Applications*, vol. Gm/R, pp. 61–68, INSTICC Press, 2007.

[88] F. Aurenhammer, "Voronoi diagrams a survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 345–405, 1991.

[89] L. Ertoz, M. Steinbach, and V. Kumar, "A new shared nearest neighbor clustering algorithm and its applications," in *Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining*, pp. 105–115, 2002.

[90] B. Pateiro-Lpez and A. Rodrguez-Casal, "Generalizing the convex hull of a sample: The r package alphahull," *Journal of Statistical Software*, vol. 34, no. 5, pp. 1–28, 2010.

[91] R. D. C. Team, "R: A language and environment for statistical computing," *Vienna, Austria R Foundation for Statistical Computing*, pp. 1–1731, 2011.

[92] J. Li and G. Lu, "Skeleton driven animation based on implicit skinning," *Computers and Graphics*, vol. 35, no. 5, pp. 945 – 954, 2011.

[93] T.-Y. Lee, P.-H. Lin, S.-U. Yan, and C.-H. Lin, "Mesh decomposition using motion information from animation sequences: Animating geometrical models," *Computer Animation and Virtual Worlds*, vol. 16, no. 3-4, pp. 519–529, 2005.

[94] A. Gregory, A. State, M. C. Lin, D. Manocha, and M. A. Livingston, "Interactive surface decomposition for polyhedral morphing," *The Visual Computer*, vol. 15, no. 9, pp. 453–470, 1999.

[95] E. Zuckerberger, "Polyhedral surface decomposition with applications," *Computers and Graphics*, vol. 26, no. 5, pp. 733–743, 2002.

[96] A. Stone and J. W. Tukey, "Generalized sandwich theorems," *Duke Mathematical Journal*, vol. 9, no. 2, pp. 356–359, 1942.

[97] J. W. Gregory, *The Making of the Earth*, vol. 54. H. Holt, 1912.

[98] V. Guillemin and A. Pollack, *Differential Topology*, vol. 370. American Mathematical Soc., 2010.

[99] G. Johns and K. Sleno, "Antipodal graphs and digraphs," *International Journal of Mathematics and Mathematical Sciences*, vol. 16, no. 3, pp. 579–586, 1993.

[100] Y.-B. Jia, "Computation on parametric curves with an application in grasping," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 827–857, 2004.

[101] V.-D. Nguyen, "Constructing force-closure grasps," *International Journal of Robotics Research*, vol. 7, no. 3, pp. 3–16, 1988.

[102] N. D. Cornea and P. Min, "Curve-skeleton properties, applications, and algorithms," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 3, pp. 530–548, 2007.

[103] I. Baran and J. Popović, "Automatic rigging and animation of 3d characters," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, p. 72, 2007.

[104] H. Martini and V. Soltan, "Antipodality properties of finite sets in euclidean space," *Discrete Mathematics*, vol. 290, no. 23, pp. 221 – 228, 2005.

[105] A. Iusem and A. Seeger, "On pairs of vectors achieving the maximal angle of a convex cone," *Mathematical Programming*, vol. 104, no. 2-3, pp. 501–523, 2005.

[106] I.-M. Chen and J. Burdick, "Finding antipodal point grasps on irregularly shaped objects," *Robotics and Automation, IEEE Transactions on*, vol. 9, no. 4, pp. 507 –512, 1993.

[107] M. Nguyn and V. Soltan, "Lower bounds for the numbers of antipodal pairs and strictly antipodal pairs of vertices in a convex polytope," *Discrete & Computational Geometry*, vol. 11, no. 1, pp. 149–162, 1994.

[108] Y.-B. Jia, "Geometry and computation of antipodal points on plane curve," tech. rep., Department of Computer Science, Iowa State University, Ames, IA 50011-1040, USA, 2001.

[109] I. E. Sutherland, R. F. Sproull, and R. A. Schumacker, "A characterization of ten hidden-surface algorithms," *ACM Computing Surveys*, vol. 6, no. 1, pp. 1–55, 1974.

[110] P. L. Rosin, "Ellipse fitting by accumulating five-point fits," *Pattern Recognition Letters*, vol. 14, no. 8, pp. 661 – 669, 1993.

[111] M. Meng, L. Fan, and L. Liu, "icutter: A direct cut out tool for 3d shapes," *Journal of Computer Animation and Virtual World*, vol. 22, no. 4, pp. 335–342, 2011.

[112] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel, "Mesh scissoring with minima rule and part salience," *Computer Aided Geometric Design*, vol. 22, no. 5, pp. 444–465, 2005.

[113] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International journal of computer vision*, vol. 1, no. 4, pp. 321–331, 1988.

[114] R. Kumar, A. Vázquez-Reina, and H. Pfister, "Radon-like features and their application to connectomics," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 186–193, IEEE, 2010.

[115] M. Agrawala, A. C. Beers, and M. Levoy, "3d painting on scanned surfaces," in *Proceedings of the 1995 symposium on Interactive 3D graphics*, I3D '95, pp. 145–ff., ACM, 1995.

[116] K.-R. GrnhH, "Automatic mesh generation with tetrahedron elements," *International Journal for Numerical Methods in Engineering*, vol. 18, no. 2, pp. 273–289, 1982.

[117] P. Cignoni, M. Corsini, and G. Ranzuglia, "Meshlab: an open-source 3d mesh processing system," *Ercim news*, vol. 73, pp. 45–46, 2008.

[118] S. A. Canann, J. R. Tristano, and M. L. Staten, "An approach to combined laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes," in *7th international meshing roundtable*, pp. 479–494, Citeseer, 1998.

[119] P. E. Black, "big-o notation," *Dictionary of Algorithms and Data Structures*, 2007.

[120] Q. Li and J. G. Griffiths, "Least squares ellipsoid specific fitting," in *Proceedings of Geometric Modeling and Processing*, pp. 335–340, IEEE, 2004.

[121] S. Katz, G. Leifman, and A. Tal, "Mesh segmentation using feature point and core extraction," *The Visual Computer*, vol. 21, no. 8-10, pp. 649–658, 2005.

[122] R. W. Hamming, "Error detecting and error correcting codes," *Bell System technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[123] R. Kikuuwe, H. Tabuchi, and M. Yamamoto, "An edge-based computationally efficient formulation of saint venant-kirchhoff tetrahedral finite elements," *ACM Transaction on Graphics*, vol. 28, no. 1, pp. 1–13, 2009.

[124] E. Saddik, "The potential of haptics technologies," *IEEE Instrumentation & Measurement Magazine*, vol. 10, no. 1, pp. 10–17, 2007.

[125] W.-W. Xu and K. Zhou, "Gradient domain mesh deformation - a survey," *Journal of Computer Science and Technology*, vol. 24, no. 1, pp. 6–18, 2009.

[126] A. Nealen, M. Mueller, R. Keiser, E. Boxerman, and M. Carlson, "Physically based deformable models in computer graphics," *Computer Graphics Forum*, vol. 25, no. 4, pp. 809–836, 2006.

[127] W. Abdelrahman, S. Nahavandi, D. Creighton, and M. Harders, "Data-driven computation of contact dynamics during two-point manipulation of deformable objects," *ASME Conference Proceedings*, vol. 2011, no. 44328, pp. 377–384, 2011.

[128] X. Guo and H. Qin, "Meshless methods for physics-based modeling and simulation of deformable models," *Science in China Series F-Information Sciences*, vol. 52, no. 3, pp. 401–417, 2009.

[129] R. Hoever, G. Kosa, G. Szekely, and M. Harders, "Data-driven haptic rendering - from viscous fluids to visco-elastic solids," *IEEE Transactions on Haptics*, vol. 2, no. 1, pp. 15–27, 2009.

[130] P. Fong, "Sensing, acquisition, and interactive playback of data-based models for elastic deformable objects," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 630–655, 2009.

[131] D. Deo and S. De, "Phyness: A physics-driven neural networks-based surgery simulation system with force feedback," in *EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics. Third Joint*, pp. 30 –34, 2009.

[132] Y. Sahillioglu and Y. Yemez, "Minimum-distortion isometric shape correspondence using em algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2203 –2215, 2012.

[133] D. K. Pai, K. v. d. Doel, D. L. James, J. Lang, J. E. Lloyd, J. L. Richmond, and S. H. Yau, "Scanning physical interaction behavior of 3d objects," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pp. 87–96, ACM, 2001.

[134] A.-M. Cretu and E. Petriu, "Neural-network-based adaptive sampling of three-dimensional-object surface elastic properties," *IEEE Transactions on Instrumentation and Measurement*, vol. 55, no. 2, pp. 483–492, 2006.

[135] K. Morooka, X. Chen, R. Kurazume, S. Uchida, K. Hara, Y. Iwashita, and M. Hashizume, "Real-time nonlinear fem with neural network for simulating soft organ model deformation," in *Proceedings of the 11th International*

Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI, pp. 742–749, Springer Berlin Heidelberg, 2008.

[136] R. Hover, M. Di Luca, G. Szekely, and M. Harders, "Computationally efficient techniques for data-driven haptic rendering," in *EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics. Third Joint*, pp. 39–44, 2009.

[137] W. Abdelrahman, S. Farag, S. Nahavandi, and D. Creighton, "Data-based dynamic haptic interaction model with deformable 3d objects," in *8th IEEE International Conference on Industrial Informatics (INDIN)*, pp. 314 –318, IEEE, 2010.

[138] J.-y. Noh and U. Neumann, "Expression cloning," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pp. 277–288, ACM, 2001.

[139] M. Ben-Chen, O. Weber, and C. Gotsman, "Spatial deformation transfer," in *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pp. 67–74, ACM, 2009.

[140] R. Zayer, C. Rassl, Z. Karni, and H.-P. Seidel, "Harmonic guidance for surface deformation," *Computer Graphics Forum*, vol. 24, no. 3, pp. 601–609, 2005.

[141] I. Baran, D. Vlasic, E. Grinspun, and J. Popović, "Semantic deformation transfer," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 36:1–36:6, 2009.

[142] K. Zhou, W. Xu, Y. Tong, and M. Desbrun, "Deformation transfer to multi-component objects," *Computer Graphics Forum*, vol. 29, no. 2, pp. 319–325, 2010.

[143] M. Staten, S. Owen, S. Shontz, A. Salinger, and T. Coffey, "A comparison of mesh morphing methods for 3d shape optimization," in *Proceedings of the 20th International Meshing Roundtable, IMR 2011*, pp. 293–311, Springer Berlin Heidelberg, 2011.

[144] N. Pietroni, M. Tarini, and P. Cignoni, "Almost isometric mesh parameterization through abstract domains," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 4, pp. 621 –635, 2010.

[145] T. Darom and Y. Keller, "Scale-invariant features for 3-d mesh models," *Image Processing, IEEE Transactions on*, vol. 21, no. 5, pp. 2758 –2769, 2012.

[146] C. Domokos, J. Nemeth, and Z. Kato, "Nonlinear shape registration without correspondences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 943–958, 2012.

[147] O. Van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or, "A survey on shape correspondence," in *Computer Graphics Forum*, vol. 30, pp. 1681–1707, Wiley Online Library, 2011.

[148] V. G. Kim, W. Li, N. J. Mitra, S. DiVerdi, and T. Funkhouser, "Exploring collections of 3d models using fuzzy correspondences," *ACM Transaction on Graphics*, vol. 31, no. 4, pp. 54:1–54:11, 2012.

[149] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas, "Functional maps: a flexible representation of maps between shapes," *ACM Transaction on Graphics*, vol. 31, no. 4, pp. 30:1–30:11, 2012.

[150] W. Feng, J. Huang, T. Ju, and H. Bao, "Feature correspondences using morse smale complex," *The Visual Computer*, pp. 1–15, 2012.

[151] M. Meyer, M. Desbrun, P. Schröder, and A. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds," *Visualization and mathematics*, vol. 3, no. 7, pp. 34–57, 2002.

[152] A. Asundi, Z. Wensen, *et al.*, "Fast phase-unwrapping algorithm based on a gray-scale mask and flood fill," *Applied optics*, vol. 37, no. 23, pp. 5416–5420, 1998.

[153] R. Kumar, J. Talton, S. Ahmad, T. Roughgarden, and S. Klemmer, "Flexible tree matching," in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume*, vol. 3, pp. 2674–2679, AAAI Press, 2011.

[154] S. Farag, W. Abdelrahman, S. Nahavandi, and D. Creighton, "Towards a parameterless 3d mesh segmentation," in *International Conference on Graphic and Image Processing (ICGIP)*, 2012.

[155] T. K. Dey and J. Sun, "Defining and computing curve-skeletons with medial geodesic function," in *Proceedings of the fourth Eurographics symposium on Geometry processing SGP '06*, pp. 143–152, Eurographics Association, 2006.

[156] A. Tagliasacchi, I. Alhashim, M. Olson, and H. Zhang, "Mean curvature skeletons," *Computer Graphics Forum*, vol. 31, no. 5, pp. 1735–1744, 2012.

[157] C. Willcocks and F. Li, "Feature-varying skeletonization," *The Visual Computer*, vol. 28, no. 6-8, pp. 775–785, 2012.

[158] N. Pantuwong and M. Sugimoto, "Skeleton growing: an algorithm to extract a curve skeleton from a pseudonormal vector field," *The Visual Computer*, vol. 29, no. 3, pp. 203–216, 2013.

[159] Q. Zhang, X. Song, X. Shao, R. Shibasaki, and H. Zhao, "Unsupervised skeleton extraction and motion capture from 3d deformable matching," *Neurocomputing*, vol. 100, pp. 170 – 182, 2013.

[160] T. Kong and A. Rosenfeld, "Digital topology: Introduction and survey," *Computer Vision, Graphics, and Image Processing*, vol. 48, no. 3, pp. 357 – 393, 1989.

[161] N. Gagvani and D. Silver, "Parameter-controlled volume thinning," *Graphical Models and Image Processing*, vol. 61, no. 3, pp. 149 – 164, 1999.

[162] G. S. di Baja and S. Svensson, "A new shape descriptor for surfaces in 3d images," *Pattern Recognition Letters*, vol. 23, no. 6, pp. 703 – 711, 2002.

[163] T. He, L. Hong, D. Chen, and Z. Liang, "Reliable path for virtual endoscopy: ensuring complete examination of human organs," *IEEE Transactions on Visualization and Computer Graphics,*, vol. 7, no. 4, pp. 333 –342, 2001.

[164] N. Cornea, D. Silver, X. Yuan, and R. Balasubramanian, "Computing hierarchical curve-skeletons of 3d objects," *The Visual Computer*, vol. 21, no. 11, pp. 945–955, 2005.

[165] G. Borgefors, I. Nyström, and G. Di Baja, "Computing skeletons in three dimensions," *Pattern Recognition*, vol. 32, no. 7, pp. 1225–1236, 1999.

[166] I. Bitter, A. Kaufman, and M. Sato, "Penalized-distance volumetric skeleton algorithm," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 3, pp. 195 –206, 2001.

[167] J. W. Brandt and V. Algazi, "Continuous skeleton computation by voronoi diagram," *Computer Vision Graphics and Image Processing (CVGIP): Image Understanding*, vol. 55, no. 3, pp. 329 – 338, 1992.

[168] S. Pizer, P. Fletcher, S. Joshi, A. Thall, J. Chen, Y. Fridman, D. Fritsch, A. Gash, J. Glotzer, M. Jiroutek, C. Lu, K. Muller, G. Tracton, P. Yushkevich, and E. Chaney, "Deformable m-reps for 3d medical image segmentation," *International Journal of Computer Vision*, vol. 55, no. 2-3, pp. 85–106, 2003.

[169] M. Attene, S. Biasotti, and M. Spagnuolo, "Shape understanding by contour-driven retiling," *The Visual Computer*, vol. 19, no. 2, pp. 127–138, 2003.

[170] N. Ahuja and J.-H. Chuang, "Shape representation using a generalized potential field model," *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, vol. 19, no. 2, pp. 169 –176, 1997.

[171] G. Abdel-Hamid and Y.-H. Yang, "Multiresolution skeletonization an electrostatic field-based approach," in *IEEE International Conference on Image Processing ICIP-94*, vol. 1, pp. 949 –953, 1994.

[172] F. Wu, W. Ma, P. Liou, R. Laing, and M. Ouhyoung, "Skeleton extraction of 3d objects with visible repulsive force," in *Computer Graphics Workshop*, pp. 124–131, 2003.

[173] W.-C. Ma, F.-C. Wu, and M. Ouhyoung, "Skeleton extraction of 3d objects with radial basis functions," in *Shape Modeling International*, pp. 207 – 215, 2003.

[174] J. Blake *et al.*, "Openkinect," 2011.

[175] M. Hossny, D. Filippidis, W. Abdelrahman, H. Zhou, M. Fielding, J. Mullins, L. Wei, D. Creighton, V. Puri, and S. Nahavandi, "Low cost multimodal facial recognition via kinect sensors," in *Land Warfare Conference*, pp. 77–86, 2012.

[176] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan, "Scanning 3d full human bodies using kinects," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 4, pp. 643 –650, 2012.

[177] W. Abdelrahman, S. Farag, S. Nahavandi, and D. Creighton, "Adaptive automatic deformation basis generation for haptic interaction with physically deformable models," in *Proceedings of Virtual Reality International Conference (VRIC 2010)* (R. Simon and S. Akihiko, eds.), (Laval, France), pp. 1–7, 2010.