UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

# Human Perception-Oriented Segmentation for Triangle Meshes

**Rui Sérgio Viegas Rodrigues**

Tese para obtenção do Grau de Doutor em
**Engenharia Informática**
($3^{o}$ Ciclo de Estudos)

Orientador: Prof. Doutor Abel João Padrão Gomes
Co-orientador: Prof. Doutor José Francisco Monteiro Morgado

**Covilhã, maio de 2016**

*To my family*

# Acknowledgements

To my advisor, Professor Doutor Abel João Padrão Gomes, I would like to express my gratitude and thanks, not only for his constant encouragement in my research career, but also for his wisdom, as well as in supporting me during these years.

To my co-advisor, Professor Doutor José Francisco Monteiro Morgado, I would also like to express my gratitude and thanks for helping to make my way in research and teaching at Instituto Politécnico de Viseu.

To Rui Pedro Duarte, I would like to thank you for your support and friendship over all these years of research work.

To my colleagues of the Departamento de Informática, who during these years have always shown interest and support through their encouragement and good humor.

Lastly, I would also like to thank my wife Graça and daughter Margarida. They were always supporting me and encouraging me with their best wishes.

O meu obrigado a todos vós.

# Resumo

A segmentação de malhas é um tópico importante de investigação em computação gráfica, em particular em modelação geométrica. Isto deve-se ao facto de as técnicas de segmentação de malhas terem várias aplicações, nomeadamente na produção de filmes, animação por computador, realidade virtual, compressão de malhas, assim como em jogos digitais. Em concreto, as malhas triangulares são amplamente usadas em aplicações interativas, visto que sua segmentação em partes significativas (também designada por segmentação significativa, segmentação perceptiva ou segmentação perceptualmente significativa ) é muitas vezes vista como uma forma de acelerar a interação com o utilizador ou a deteção de colisões entre esses objetos 3D definidos por uma malha, bem como animar uma ou mais partes significativas (por exemplo, a cabeça de uma personagem) de um dado objeto, independentemente das restantes partes.

Acontece que não se conhece nenhuma técnica capaz de segmentar correctamente malhas arbitrárias —ainda que restritas aos domínios de formas livres e não-livres— em partes significativas. Algumas técnicas são mais adequadas para objetos de forma não-livre (por exemplo, peças mecânicas definidas geometricamente por quádricas), enquanto outras são mais talhadas para o domínio dos objectos de forma livre. Só na literatura recente surgem umas poucas técnicas que se aplicam a todo o universo de objetos de forma livre e não-livre. Pior ainda é o facto de que a maioria das técnicas de segmentação não serem totalmente automáticas, no sentido de que quase todas elas exigem algum tipo de pré-requisitos e assistência do utilizador. Resumindo, estes três desafios relacionados com a proximidade perceptual, generalidade e automação estão no cerne do trabalho descrito nesta tese.

Para enfrentar estes desafios, esta tese introduz o primeiro algoritmo de segmentação baseada nos contornos ou fronteiras dos segmentos, cuja técnica se inspira nas técnicas de segmentação baseada em arestas, tão comuns em análise e processamento de imagem, por contraposição às técnicas e segmentação baseada em regiões. A ideia principal é a de encontrar em primeiro lugar a fronteira de cada região para, em seguida, identificar e agrupar todos os seus triângulos internos. As regiões da malha encontradas correspondem a saliências e reentrâncias, que não precisam de ser estritamente convexas, nem estritamente côncavas, respectivamente. Estas regiões, designadas regiões relaxadamente convexas (ou saliências) e regiões relaxadamente côncavas (ou reentrâncias), produzem segmentações que são menos sensíveis ao ruído e, ao mesmo tempo, são mais intuitivas do ponto de vista da perceção humana; por isso, é designada por segmentação orientada à perceção humana (ou, *human perception- oriented* (HPO), do inglês). Além disso, e ao contrário do atual estado-da-arte da segmentação de malhas, a existência destas regiões relaxadas torna o algoritmo capaz de segmentar de maneira bastante plausível tanto objectos de forma não-livre como objectos de forma livre.

Nesta tese, enfrentou-se também um quarto desafio, que está relacionado com a fusão de segmentação e multi-resolução de malhas. Em boa verdade, já existe na literatura uma variedade grande de técnicas de segmentação, bem como um número significativo de técnicas de multi-resolução, para malhas triangulares. No entanto, não é assim tão comum encontrar estruturas de dados e algoritmos que façam a fusão ou a simbiose destes dois conceitos, multi-resolução e segmentação, num único esquema multi-resolução que sirva os propósitos das aplicações que lidam com malhas simples e segmentadas, sendo que neste contexto se entende que uma malha simples é uma malha com um único segmento. Sendo assim, nesta tese descreve-se um novo esquema (entenda-se estruturas de dados e algoritmos) de multi-resolução e segmentação, designado por *extended Ghost Cell* (xGC). Este esquema preserva a forma das malhas, tanto em termos globais como locais, ou seja, os segmentos da malha e as suas fronteiras, bem como os seus vincos e ápices são preservados, não importa o nível de resolução que usamos durante a/o simplificação/refinamento da malha. Além disso, ao contrário de outros esquemas de segmentação, tornou-se possível ter segmentos adjacentes com dois ou mais níveis de resolução de diferença. Isto é particularmente útil em animação por computador, compressão e transmissão de malhas, operações de modelação geométrica, visualização científica e computação gráfica.

Em suma, esta tese apresenta um esquema genérico, automático, e orientado à percepção humana, que torna possível a simbiose dos conceitos de segmentação e multi-resolução de malhas trianguladas que sejam representativas de objectos 3D.

## Palavras-chave

Segmentação de malhas
Convexidade relaxada
Perceção humana
Malhas multiresolução

# Abstract

The mesh segmentation is an important topic in computer graphics, in particular in geometric computing. This is so because mesh segmentation techniques find many applications in movies, computer animation, virtual reality, mesh compression, and games. In fact, triangle meshes are widely used in interactive applications, so that their segmentation in meaningful parts (i.e., human-perceptually segmentation, perceptive segmentation or meaningful segmentation) is often seen as a way of speeding up the user interaction, detecting collisions between these mesh-covered objects in a 3D scene, as well as animating one or more meaningful parts (e.g., the head of a humanoid) independently of the other parts of a given object.

It happens that there is no known technique capable of correctly segmenting any mesh into meaningful parts. Some techniques are more adequate for non-freeform objects (e.g., quadric mechanical parts), while others perform better in the domain of freeform objects. Only recently, some techniques have been developed for the entire universe of objects and shapes. Even worse it is the fact that most segmentation techniques are not entirely automated in the sense that almost all techniques require some sort of pre-requisites and user assistance. Summing up, these three challenges related to *perceptual proximity*, *generality* and *automation* are at the core of the work described in this thesis.

In order to face these challenges, we have developed the first contour-based mesh segmentation algorithm that we may find in the literature, which is inspired in the edge-based segmentation techniques used in image analysis, as opposite to region-based segmentation techniques. Its leading idea is to firstly find the contour of each region, and then to identify and collect all of its inner triangles. The encountered mesh regions correspond to ups and downs, which do not need to be strictly convex nor strictly concave, respectively. These regions, called relaxedly convex regions (or saliences) and relaxedly concave regions (or recesses), produce segmentations that are less-sensitive to noise and, at the same time, are more intuitive from the human point of view; hence it is called human perception- oriented (HPO) segmentation. Besides, and unlike the current state-of-the-art in mesh segmentation, the existence of these relaxed regions makes the algorithm suited to both non-freeform and freeform objects.

In this thesis, we have also tackled a fourth challenge, which is related with the *fusion* of mesh segmentation and multi-resolution. Truly speaking, a plethora of segmentation techniques, as well as a number of multiresolution techniques, for triangle meshes already exist in the literature. However, it is not so common to find algorithms and data structures that fuse these two concepts, multiresolution and segmentation, into a symbiotic multi-resolution scheme for both plain and segmented meshes, in which a plain mesh is understood as a mesh with a single segment. So, we introduce such a novel multiresolution segmentation scheme, called extended Ghost Cell (xGC) scheme. This scheme preserves the shape of the meshes in both global and local terms, i.e., mesh segments and their boundaries, as well as creases and apices are preserved, no matter the level of resolution we use for simplification/refinement of the mesh. Moreover, unlike other segmentation schemes, it was made possible to have adjacent segments with two or more resolution levels of difference. This is particularly useful in computer animation, mesh compression and transmission, geometric computing, scientific visualization, and computer graphics.

In short, this thesis presents a fully automatic, general, and human perception-oriented scheme that symbiotically integrates the concepts of mesh segmentation and multiresolution.

## Keywords

Mesh segmentation
Relaxed convexity
Human perception
Multiresolution meshes

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| 1D | One-Dimensional |
| 2D | Bi-Dimensional |
| 3D | Three-Dimensional |
| ACD | Approximate Convex Decomposition |
| AGD | Average Geodesic Distance |
| AIF | Adjacency and Incidence Framework |
| BS | Boundary Segmentation |
| BVH | Bounding Volume Hierarchy |
| CAD | Computer Aided Design |
| CD | Cut Discrepancy |
| CE | Consistency Error |
| CMD | Constrained Morphological Decomposition |
| CSD | Convex Shape Decomposition |
| CSG | Constructive Solid Geometry |
| DRG | Discret Reeb Graph |
| ECD | Exact Convex Decomposition |
| FACD | Fast Approximate Convex Decomposition |
| GC | Geometric Contraction |
| GLUI | OpenGL User Interface Library |
| GLUT | OpenGL Utility Toolkit |
| GMM | Gaussian Mixture Models |
| HC | Hierarchical Clustering |
| HD | Hamming Distance |
| HMS | Heat Mean Signature |
| HPO | Human Perception- Oriented |
| IC | Iterative Clustering |
| LBO | Laplace-Beltrami Operator |
| LOD | Level of Detail |
| MA | Medial Axis |
| MAT | Medial Axis Transform |
| MC | Mesh Contraction |
| MDS | Multi-Dimensional Scaling |
| MNCD | Minimum Near-Convex Decomposition |
| MRF | Markov Random Fields |
| MSP | Minimum Slice Perimeter |
| MVBB | Minimum Volume Bounding Box |
| PCMS | Perceptually Consistent Mesh Segmentation |
| PMT | Point Membership Test |
| PRST | Planar-Reflective Symmetry Transform |

| | |
|---|---|
| RG | Region Growing |
| RI | Rand Index |
| SP | Space Partitioning |
| SDF | Shape Diameter Function |
| SMC | Set Membership Classification |
| VM | Volumetric Meshes |
| VCS | Volumetric Contraction |
| WS | Watershed Segmentation |
| xGC | extended Ghost Cell |
| XMR | eXtended Multi-Ring |

# Chapter 1

# Introduction

The mesh segmentation is an important topic in geometric computing and computer graphics, largely because of its applications in movies, computer animation, virtual reality, mesh compression, and games. In fact, triangle meshes are widely used in interactive applications, so that their segmentation in meaningful parts (i.e., human-perceptually segmentation, perceptive segmentation or meaningful segmentation) is a way of speeding up the interaction, detecting collisions between these mesh-covered objects in a 3D scene, as well as animating one or more meaningful parts (e.g., the head of a humanoid) independently of the other parts of a given object. It happens that there is no known technique capable of correctly segmenting any mesh into meaningful parts. Some techniques are more adequate for non-freeform objects (e.g., quadric mechanical parts), while others perform better in the domain of freeform objects. Only recently, some techniques have been developed for the entire universe of objects and shapes. Even worse it is the fact that most segmentation techniques are not entirely automated in the sense that almost all techniques require some sort of pre-requisites and user assistance. Summing up, these three challenges related to *perceptual proximity*, *generality* and *automation* are at the core of the work described in this thesis.

## 1.1 Thesis Statement

In general, mesh segmentation falls into two major categories: perceptive segmentation and non-perceptive segmentation. Perceptive segmentation, also called human perception-based segmentation, divides a mesh into meaningful regions (e.g., the legs of a human body). On the other hand, non-perceptive segmentation divides a mesh into charts sharing a common set of geometric features. Additionally, the process of segmentation may involve interaction by the user, and may take advantage of prior knowledge we have about the objects modelled by such meshes. In addition, the segmentation often depends on the geometry of objects coated by mesh triangles, i.e., non-freeform and freeform objects.

In this work, we aim at segmentation techniques that satisfy the following criteria:

- Meaningful segmentation;
- Without user interaction;
- Only use of information available on the mesh itself;

- Apply to the largest possible number of objects.

Taking into consideration the requirements mentioned above, we can formulate the thesis statement as follows:

> *Is it feasible to segment different categories of meshes in a single, automated manner as the humans perceptually do, regardless of their level of detail?*

As shown throughout this thesis, we intend to prove that this statement is true in practice, in particular using the concept of relaxed convexity, which has much to do with how humans perceive the ups and downs of real-world objects, and mentally organize their shapes in a hierarchical manner, the so-called meaningful or perceptive segmentation.

## 1.2 Research Questions

In order to validate the thesis statement above, it is necessary to answer to the following research questions:

**Is it possible to segment a given object into meaningful parts as they are perceived by the human being?**

The principal objective of the meaningful or perceptive segmentation is to divide a mesh into meaningful segments, i.e., segments that match the way how humans perceive objects and their parts in a hierarchical manner. According to Hoffman's assertion [HS97], humans normally identify a given object as a set of its convex regions, which look separate from each other along concave boundaries. The third chapter of this thesis describes an algorithm that outputs a perceptive segmentation of 2D meshes in 3D space.

**Is it possible to segment both freeform and non-freeform objects into meaningful parts through a single segmentation algorithm?**

Usually, a segmentation algorithm is designed for a particular category of objects, e.g., mechanical parts, humanoids, animals, handicraft, and so forth. That is, the geometric coverage of a segmentation algorithm is limited. In third chapter of this thesis, and following recent trends in mesh segmentation techniques, we describe a segmentation algorithm that applies to both freeform and non-freeform objects, though their geometries have been previously triangulated. Therefore, we use triangle meshes as piecewise-linear approximations for any sort of geometric object.

**Is it possible to design and implement an algorithm to automatically extract segments without requiring user interaction?**

One of the problems with many segmentation algorithms is that they are dependent upon some form of user interaction or assistance; in particular, it is often necessary to indicate the final number of segments, to pick up at least a point of each region by direct interaction, of even to provide some training set of mesh segmentations of the same object (i.e., the so-called co-segmentation). In third chapter of this thesis, an automated mesh segmentation algorithm is proposed, which is capable of calculating the number of segments beforehand in an automatic manner.

**Is it possible to use mesh segmentation within a multiresolution scheme that preserves the segments?**

Usually, mesh segmentation is designed and implemented on plain meshes, i.e., meshes without multiresolution. In the fourth chapter of this thesis, the concept of multiresolution mesh segmentation was taken to a point such that we can say that the concepts of segmentation and multiresolution seem to be merged. Additionally, this scheme preserves the shape of the mesh by preserving its segments, as well as its apices and creases.

## 1.3 Research Context

Segmented meshes have multiple applications in computer graphics, namely morphing, texture mapping, shape reconstruction, collision detection, 3D shape retrieval, and so forth. For example, in collisions detection, using segmented meshes allows us to speed up the computation of collisions between objects in motion because its is easier to construct a bounding volume hierarchy (BVH) for a segmented mesh than for a plain mesh.

It is clear that many geometric operations —including the segmentation of a plain mesh— can be performed manually or with the user assistance. But, truly speaking, most —not to say all— segmentation algorithms fail to divide one or more meshes into meaningful parts from the human perception point of view. The rationale behind this issue relative to the perceptive correctness, or at least the perceptual proximity, lies in fact that it is rather difficult to know in advance the exact number of segments of a mesh. This is so in spite of being consensually accepted that humans perceive objects as hierarchical collections of parts [Bie87], with boundaries of parts perceived along the negative minima of principal curvatures [HS97]; hence the so-called Hofman's assertion, which implies that the meaningful parts are essentially convex.

The *a priori* unknown number of segments or parts also means that most mesh segmentation algorithms do not work in an automated manner. In fact, most of these algorithms require the user to enter the number of segments beforehand. In this respect, we found that mesh segmentation in computer graphics ranks behind image segmentation techniques (e.g., histogram techniques) —commonly found in the field of image analysis and processing— in terms of innovative cutting-edge solutions. One of the leading ideas of this thesis is thus to elaborate new image analysis-inspired techniques to determine the number of meaningful mesh segments.

Furthermore, mesh segmentation algorithms suffer from the lack of generality. Some algorithms readily apply to freeform objects like humanoids, animals, handicraft, and the like. Some algorithms are more adequate for the segmentation of non-freeform objects like mechanical parts and engineering artifacts. We rarely find an algorithm that works equally well for both categories of objects, freeform and non-freeform objects. In this thesis, we were also motivated by this generality issue, which shows us that this research topic is far from its complete resolution.

Moreover, with the increasing complexity of the geometric models, largely propelled by the new data acquisition devices (e.g., laser scanners), modern geometric schemes capable of symbiotically combining segmentation and multiresolution in triangle meshes have become a need to be satisfied by geometric modeling tools and systems at the disposal of common users and industrial designers.

## 1.4 The Course of the Research Work

The work that resulted in this document involved a number of tasks, as listed in sequel:

1. **Literature review**. This stage took so long because the entire literature on mesh segmentation has been reviewed carefully. This review focused on meaningful segmentation, but chart segmentation was also briefly approached. Besides, some literature related to cognitive sciences was also approached and studied in order to better understand how humans perceive objects and the surrounding world.
2. **Implementation of a few state-of-the-art segmentation algorithms**. Four classical segmentation algorithms were implemented, more specifically those region growing and watershed algorithms introduced by Mangan and Whitaker [MW99] and Zuckerberger et al. [ZTS02]. This stage was important to grasp the mathematics behind mesh segmentation, namely the notions of curvature and convexity. Additionally, it was also relevant to realize the limitations and open issues of the current segmentation methods.
3. **Design and implementation of our perceptive mesh segmentation algorithm**. Likely, this was the most time-consuming task of the entire doctoral programme. This was so because we took some time to come up to the concept of relaxed convexity, which —we believe— better translates the way how humans perceive the ups and downs of a mesh. Another difficulty of the algorithm was to come up to the number of segments in an automated manner. For that, we ended up using histogram techniques so common in image analysis and processing.
4. **Testing of our perceptive mesh segmentation algorithm**. Our algorithm was extensively tested and compared to standard mesh segmentations of the Princeton

benchmark [CGF09]. The corresponding quantitative evaluation has shown that our algorithm outperforms the state-of-the-art segmentation algorithms in the sense that the resulting segmentations are closer to those of the ground truth (i.e., human segmentation).

5. **Design and implementation of a multiresolution scheme for perceptive meshes**. The challenge here was how to combine the concepts of mesh segmentation and multiresolution mesh, with the particularity of preserving the shape of the mesh in terms of its segments, creases, and apices. Besides, it was made possible to have segments with distinct resolutions or levels of detail.

6. **Writing of the thesis**. The thesis was being written whilst the papers were being submitted to conferences and journals. Therefore, each core chapter of this thesis corresponds to a separate paper.

## 1.5 Contributions

The contributions of this thesis are the following:

- To our best knowledge, we here propose the first contour-based segmentation in the domain of mesh segmentation in 3D. For this purpose, we use the point membership test (PMT) as a convexity classifier, i.e., we use PMT to classify edges as convex, concave or flat. PMT is a particular case of the SMC (set membership classification) test, which is very popular in CSG (constructive solid geometry) modeling, and has been around for the last three to four decades [Til80].

- Nevertheless, recesses and saliences on the mesh are not classified in a so strict manner in terms of convexity. We introduce the notion of *relaxed convexity* to classify those shape features of the mesh. A salience is a relaxedly convex region, while a recess is a relaxedly concave region. A relaxedly convex region is not strictly convex, i.e., it admits small concavities. On the other hand, a relaxedly concave region is not strictly concave, i.e., it admits small convexities. This shape relaxation allows us to minimize the effects of noise related to over-segmentation, and makes it suited to the segmentation of freeform objects.

- The computation of the number of segments of a mesh is performed in an automated manner using a histogram-based technique, i.e., without the common user assistance or interaction. This histogram-based technique was inspired in the histogram-based segmentation used in image analysis and processing, but it has never been used in mesh segmentation.

- At our best knowledge, this thesis introduces the first *general* data structure for multi-resolution segmented meshes. The generality comes from the fact that two adjacent segments may possess quite distinct levels of detail. It is true that in the literature we also find multiresolution progressive meshes, but they only admit adjacent segments with one level of detail of difference.

- We also introduce shape-preserving simplification and refinement operators for multiresolution segmented meshes. This requires that the boundaries that separate mesh segments from each other must hold after all. Moreover, creases and apices are also preserved after finding them through the covariance matrix norm. Also, simplification and refinement operators can be applied to the entire mesh or to any number of segments in an independent manner. This is so because the segments may have distinct levels of detail.

## 1.6 Publications

**Rui S.V. Rodrigues, José F.M. Morgado, and Abel J.P. Gomes. 2016. Mesh Segmentation: A Literature Review. ACM Computing Surveys (submitted), 2016.**

*Abstract.* Mesh segmentation is a fundamental research topic in geometry processing and computer graphics. This paper presents an exhaustive literature review of mesh segmentation techniques and algorithms, with a focus on meaningful segmentation, i.e., segmentation that divides a mesh into perceptually meaningful regions. Such perceptually meaningful segmentation is also called perceptive segmentation (or human perception-orientation segmentation) because it attempts to mimic how humans perceive the surrounding world and organize its constituent objects into parts. This means that we do not consider here mesh segmentation into non-meaningful parts (i.e., charts). Additionally, we only consider segmentation techniques with reference to a single mesh at a time, so that no co-segmentation techniques are considered either. Finally, the taxonomy followed in this paper is new in the sense that several mesh segmentation algorithms are classified in terms of the structural dimension (i.e., 1D, 2D, and 3D) at the heart of each technique. The leading idea behind this literature review is to identify the properties and limitations of state-of-the-art algorithms in order to shed light on the challenges for future work.

**Rui S.V. Rodrigues, José F.M. Morgado, and Abel J.P. Gomes. 2015. A contour-based segmentation algorithm for triangle meshes in 3D space. *Computers & Graphics* 49 (2015), pp. 24-35.**

*Abstract.* This paper introduces the first contour-based mesh segmentation algorithm that we may find in the literature, which is inspired in the edge-based segmentation techniques used in image analysis, as opposite to region-based segmentation techniques. Its leading idea is to firstly find the contour of each region, and then to identify and collect all of its inner triangles. The encountered mesh regions correspond to ups and downs, which do not need to be strictly convex nor strictly concave, respectively. These regions, called relaxedly convex regions (or saliences) and relaxedly concave regions (or recesses), produce segmentations that are less-sensitive to noise and, at the same time, are more intuitive from the human point of view; hence it is called human perception-oriented (HPO) segmentation. Besides, and unlike current state-of-the-art in mesh segmentation, the existence of these relaxed regions makes the algorithm suited to both non-freeform and freeform objects.

**Rui S.V. Rodrigues, José F.M. Morgado, and Abel J.P. Gomes. 2016. A Shape-Preserving Multiresolution Segmentation Scheme for Triangle Meshes with Creases and Apices. *Computers & Graphics* (submitted), 2016.**

*Abstract*. A plethora of segmentation techniques, as well as a number of multiresolution techniques, for triangle meshes already exist in the literature. However, it is not so common to find algorithms and data structures that fuse these two concepts, multiresolution and segmentation, into a symbiotic multi-resolution scheme for both plain and segmented meshes, in which a plain mesh is understood as a mesh with a single segment. In this paper, we introduce such a novel multiresolution segmentation scheme, called extended Ghost Cell (xGC) scheme. This scheme preserves the shape of the meshes in both global and local terms, i.e., mesh segments and their boundaries, as well as creases and apices are preserved, no matter the level of resolution we use for simplification/refinement of the mesh. Moreover, unlike other segmentation schemes, it was made possible to have adjacent segments with two or more resolution levels of difference. This is particularly useful in computer animation, mesh compression and transmission, geometric computing, scientific visualization, and computer graphics.

## 1.7 Software and Hardware Tools

In this work, all algorithms were designed and implemented on an Intel Core Duo 2.4 computer running a Mac OS X operating system, and using the OpenGL User Interface Library (GLUI), which is a C++ user interface library based on the OpenGL Utility Toolkit (GLUT). In testing of the our segmentation algorithm, we compared the mesh segmentations produced by our algorithm with those made available by the Princeton benchmark [CGF09], in order to better evaluate how close they were to human ground truth.

## 1.8 Thesis Outline

**Chapter** 1. The thesis statement, the principal research questions, the main contributions, the main publications and software, as well as the motivation that has led to the writing of this thesis, are presented in a brief manner.

**Chapter** 2. This chapter makes an exhaustive literature review on mesh segmentation, with a focus on perceptive mesh segmentation algorithms and techniques.

**Chapter** 3. This chapter describes the first contour-based mesh segmentation algorithm. Starting from the region boundaries or contours, the algorithm iteratively builds the relaxedly convex regions (or saliences) and relaxedly concave regions (or recesses), which are less-sensitive to the noise, allowing at the same time to create perceptually meaningful regions from the human point of view.

**Chapter** 4. This chapter presents a multi-resolution scheme for segmentation of triangle meshes, which preserves the shape of the meshes in both global and local terms, no matter the level of resolution we use for simplification/refinement of the mesh. Furthermore, and unlike other schemes, it was made possible to have adjacent segments with two or more resolution levels of difference.

**Chapter** 5. This chapter presents the main conclusions of the research work behind this thesis, advancing with some open issues for future work.

# Chapter 2

# Meaningful Mesh Segmentation: a Survey

Mesh segmentation is a fundamental research topic in geometry processing and computer graphics. This chapter presents an exhaustive literature review of mesh segmentation techniques and algorithms, with a focus on meaningful segmentation, i.e., segmentation that divides a mesh into perceptually meaningful regions. Such perceptually meaningful segmentation is also called perceptive segmentation (or human perception-orientation segmentation) because it attempts to mimic how humans perceive the surrounding world and organize its constituent objects into parts. This means that we do not consider here mesh segmentation into non-meaningful parts (i.e., charts). Additionally, we only consider segmentation techniques with reference to a single mesh at a time, so that no co-segmentation techniques are considered either. Finally, the taxonomy followed in this thesis is new in the sense that the several mesh segmentation algorithms are classified in terms of the structural dimension (i.e., 1D, 2D, and 3D) at the heart of each technique. The leading idea behind this literature review is to identify the properties and limitations of the state-of-the-art algorithms in order to shed light on the challenges for future work.

## 2.1 Introduction

The mesh segmentation consists in dividing a mesh into parts. Mesh segmentation in the 3D space is a challenging problem in geometry processing and computer graphics, which has been around for two to three decades. Online repositories like AIMSHAPE (`http://visionair.ge.imati.cnr.it/ontologies/shapes/`) and benchmarks like Princeton's [CGF09] were created to mainly sustain the mesh segmentations research, yet they have been instrumental in object retrieval techniques [FMA+10] .

In general, there are three major categories of mesh segmentation algorithms (Figure 2.1):

- *Chart segmentation*. In this case, a given mesh is decomposed into charts, with the geometric entities (i.e., vertices, edges and facets) of each chart satisfying the same geometric property (e.g., planarity, convexity, and curvature) within a specific threshold or range. The resulting charts do not necessarily match the ones produced by humans through their vision, i.e., they are not necessarily meaningful from the human point of view.
- *Meaningful segmentation*. The main objective of the meaningful segmentation is

to decompose a given mesh into submeshes or patches that correspond to meaningful parts (e.g., the fingers of a hand) as perceived by humans. Therefore, the cognitive science plays here an important role [HS97] [Bie87].

- *Co-segmentation.* Unlike other segmentation algorithms, co-segmentation algorithms makes usage of two or more mesh segmentations of the same object, with each segmentation featuring a distinct object pose. Nevertheless, the leading idea is to produce a meaningful segmentation of a mesh, yet with reference to a set of already known co-segmentations.

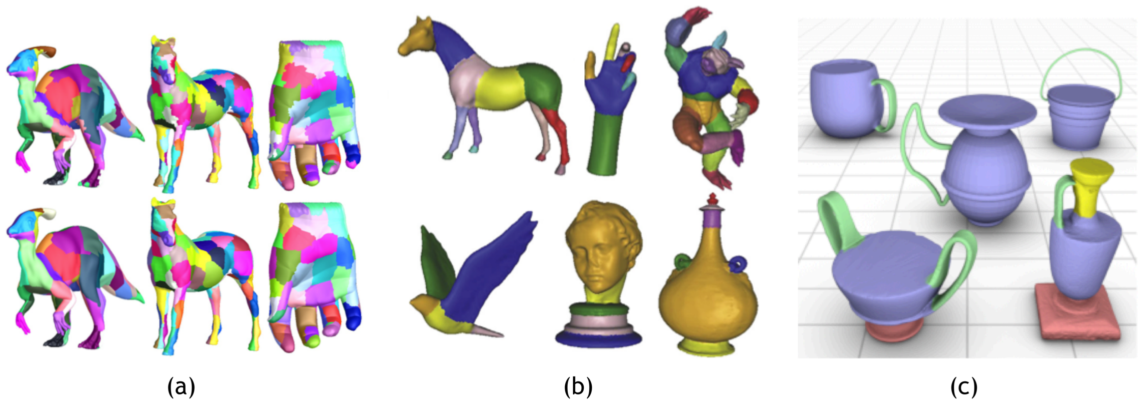


Figure 2.1: Different types of segmentation: (a) chart segmentation [SSGH01]; (b) meaningful segmentation [WLAT14]; (c) co-segmentation [SvKK$^+$11].

In a word, mesh parts can be meaningful from the human perception point of view or not. This chapter focuses on meaningful segmentation techniques, though co-segmentation techniques are not at all considered. One of the contributions of this chapter has to do with a new way of classifying *meaningful* mesh segmentation algorithms, which is carried out with reference to the structural dimension of the segments or regions. With this in mind, we have identified three main categories of meaningful mesh segmentation techniques:

1. *Volume-based segmentation.* In this case, the segments are volumes. The input is a 3D volumetric mesh, which is then partitioned into 3D volumetric submeshes in the attempt of matching human perception in a particular context. In fact, as argued by Hoffman and Richards [HR84], the volumetric convexity is often related to the human perception about shape and, consequently, shape segmentation.

2. *Surface-based segmentation.* In this technique, the segments are 2D submeshes or regions of a 2D triangle mesh. Each region consists of a number of connected facets that have similar geometric properties (e.g., convexity, curvature, etc.).

3. *Skeleton-based segmentation.* In this technique, also known as *skeletonization*, the segments are line segments. The input is either a 3D volumetric mesh or a 2D

surface mesh, but the output is a 1D skeleton that represents the structural shape of the mesh.

Note that meaningful segmentations have a number of applications in computer graphics, computer animation, computational cinematography, games, and so forth, namely:

- *Shape modeling* [JLCW06] – to be able to model the object parts in a separate manner;
- *Multiresolution and mesh compression* [MGH11] – to be able to simplify/refine each object part separately in the mesh;
- *Morphing* [STK02] – to be able to transform an object into another by morphing homologous parts separately;
- *Texture mapping* [SWG+03] – to be able to mapping textures onto object parts or regions separately;
- *Shape reconstruction* [KFK+14] – to be able to recover the object shape from its significant parts;
- *Collision detection* [LWTH01] – to be able to decompose an object into parts in order to more easily to compute bounding-volume hierarchies for collision detection purposes;
- *3D shape retrieval* [FMA+10] – to be able to retrieve a 3D model from a mesh repository or database through its shape decomposition into parts.
- *Character animation* [dGGV08] – to be able to animate the human parts separately or in a combined manner.

Summing up, the above three meaningful mesh segmentation categories are all based on geometry, because they deal with concepts such as convexity, topology, homology, etc. Nevertheless, the ultimate objective is always to meet a mesh segmentation that matches the human perception about the shape of an object in a particular context, regardless of whether eventual ambiguities exist or not. In this sense, we can say that geometry-based segmentation precedes perception-based segmentation. For example, in 3D shape retrieval, it may be relevant to be able to identify arms and legs in a mesh that represents the human body. Thus, geometry pervades the entire shape segmentation pipeline. Figure 2.2 shows the proposed taxonomy of the meaningful segmentation algorithms.

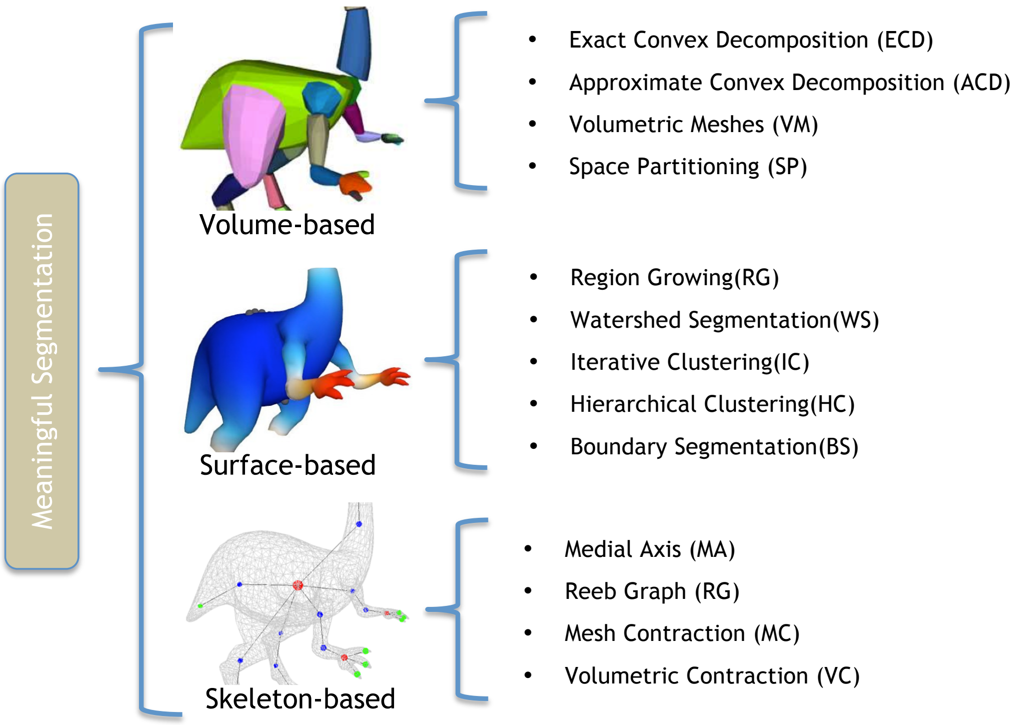


Figure 2.2: Taxonomy of mesh segmentation algorithms (pictures taken from [LA06][dGGV08][TVD07]).

## 2.2 Volume-Based Segmentation

Volume-based segmentation algorithms can be considered as the former category of shape segmentation algorithms. These algorithms divide a 3D volumetric object into a set of (approximately) convex sub-objects or volumes, but they may also divide the 3D space into 3D regions. This family of algorithms originated in the seminal work of Chazelle [Cha84], in the context of computational geometry. The leading idea is to partition complex structures into simpler convex components, as needed in other geometric algorithms such as, for example, the computation of intersection of solids, collision detection, and geometric point location.

There are four main classes of volume-based segmentation methods:

***Exact Convex Decomposition*** (ECD). In this case, the sub-meshes represent exact convex volumes. Chazelle [Cha84] was not concerned with mesh segmentation at the time, so that the partition of an object into convex sub-objects does not necessarily results in a meaningful segmentation. Therefore, this class of algorithms is here included only for historical reasons.

***Approximate Convex Decomposition*** (ACD). As the name on it says, the sub-meshes represent approximate convex volumes. In fact, and unlike ECD methods, these methods have been introduced for approaching the problem of mesh segmentation.

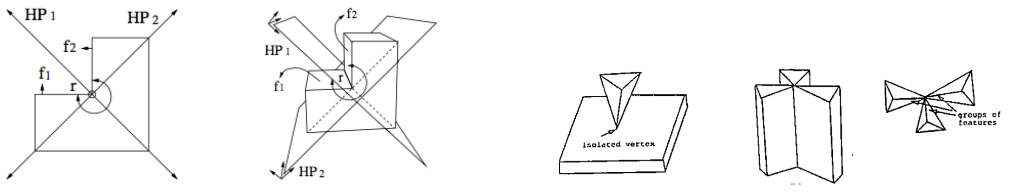
***Volumetric Meshes*** (VM). The distinctive feature of volumetric meshes is that they are made up of a set of small volumetric elements such as tetrahedra, hexahedra, or voxels

(3D cubical cells with volume). This means that VM-based methods take advantage of this distinctive feature to carry out the segmentation of a volumetric mesh into volumetric sub-meshes.

***Space Partitioning*** (SP). In this case, the 3D domain itself that contains the mesh is divided into 3D regions, inducing somehow the decomposition of the mesh into sub-meshes or sub-volumes.

### 2.2.1 Exact Convex Decomposition

The problem of convex decomposition of polyhedra was firstly addressed by Chazelle [Cha81] [CP90] [CP94]. The partition of a polyhedron into sub-polyhedra follows the notch cutting principle, i.e., the cutting is performed along concave edges of the surface. It is noteworthy that a notch of a manifold polyhedron is an edge with a dihedral angle less than 180 degrees, as shown in Figure 2.3(a). Later, Bajaj and Dey [BD92] and Hershberger and Snoeyink in [HS98] extended this technique to more complex objects, including non-manifold objects, as shown in Figure 2.3(b).



(a) (b)

Figure 2.3: ECD based on notch cutting (pictures taken from [Cha81] and [BD92]): (a) manifold objects; (b) non-manifold objects.

More recent works in this class of algorithms have been presented by Juttler et al. [JKP13] and Nguyen et. al [NPJ14], which recursively decompose a three-dimensional solid into a small number of topological hexahedra, i.e., components with a decreasing number of notches. In the end, the resulting topological hexahedra likely are devoid of notches, but they are not necessarily convex. These methods search for a cutting loop in an edge graph, which can be used to decompose the solid into two smaller solids with fewer non-convex edges. The cutting loop is selected with reference to values of planarity and geodesic curvature.

However, the ECD methods usually generate an unmanageable number of components or sub-polyhedra, particularly in the case of objects with a significant number of convexity/concavity changes, which is costly in terms of time performance and storage. In addition to this performance drawback, the most important downside of the ECD technique is that it does not necessarily generates meaningful parts from the segmentation point of view.

### 2.2.2 Approximate Convex Decomposition

The partitioning of an object into *approximately* convex components is more efficient than to compute exact convex components, in the sense that it results in a less number of components. The approximate convex decomposition (ACD) has the further advantage that it generates more meaningful components, i.e., it is closer to what we understand as a meaningful segmentation (Figure 2.4).

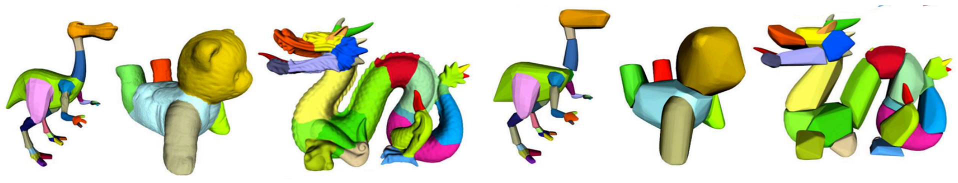


Figure 2.4: ACD based on a measure of concavity (picture taken from [LA06]).

Seemingly, the first ACD method was proposed by Lien and Amato [LA04]. This technique consists in cutting a polyhedron into approximately convex pieces. Lien and Amato's method builds on a measure of concavity of each notch. Essentially, they measure the distance from each notch to the convex hull that encloses the original polyhedron, i.e., a longer distance means deeper concavity. This means that we end up having a concavity measure for each notch, as well for the polyhedron itself as a sum of concavity values for all notches found on the polyhedron surface. The polyhedron it then divided so that the concavity values of the identified features are reduced [LA06]. This process is repeated until the decomposition achieves an acceptable overall concavity value. If necessary a merge step is applied to remove some redundant parts.

Unlike Lien and Amato [LA06], Kraevoy et al. [KJS06] [KJS07] use a measure of convexity instead of concavity (Figure 2.5). For that purpose, one measures the average distance from each triangle to the convex hull of its hosting object. Furthermore, they use a supplementary criterion, called compactness, in order to correctly delineate the boundary of each segment or part, even when there is no cycle of concave edges (e.g., non-concave cycle that separates index finger from hand palm). With these two criteria at hand, each segment grows from a seed triangle according to a region-growing approach that selects the best vertex to attach and update the segment; hence, the convex hull of the segment needs to be updated accordingly.

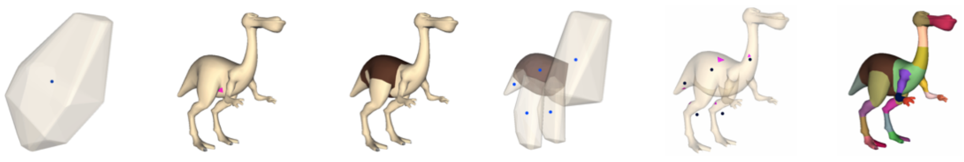


Figure 2.5: ACD based on a measure of convexity (picture taken from [KJS07]).

In 2010, Liu et al. [LLL10] introduced a new segmentation technique, called convex shape decomposition (CSD), as shown in Figure 2.6(a). In formal terms, the problem of the convex decomposition of an object is formulated as an integer linear programming problem, i.e., an object is decomposed into nearly convex parts at minimal cost. It is noteworthy that one uses Morse functions in the process of transforming the problem of the convex decomposition into an integer linear programming problem. The resulting approximate optimal decomposition is thus obtained through the minimization of its total cost under some concavity constraints. Therefore, the number of parts (and also cuts) of the decomposition tends to be minimal, and, similar to Lien and Amato [LA06], it uses a concavity measure.

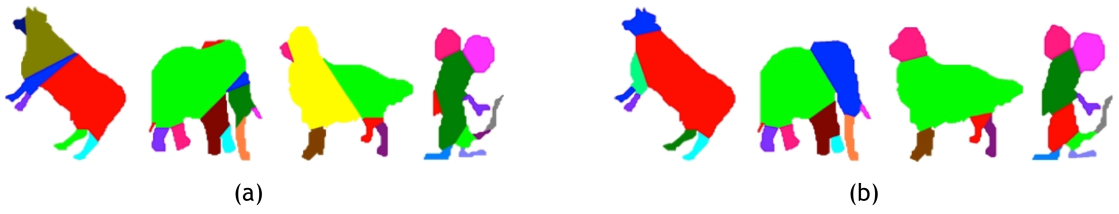


Figure 2.6: ACD based on a measure of (pictures taken from [LLL10] and [RYLL11]): (a) concavity (CSD); (b) near-convexity (MNCD).

Similar to CSD, Ren et al. [RYLL11] developed another approximate convex decomposition method called minimum near-convex decomposition (MNCD), as illustrated in Figure 2.6(b). This method has been put forward to solve two major drawbacks of the methods above, namely *redundancy* and lack of *naturalness* of the nearly convex parts of the decomposition. Consequently, the decomposition problem is formulated as a discrete optimization problem in the sense that one intends to minimize the number of non-intersecting cuts and, at the same time, to meet the requirement of high visual naturalness. The naturalness of the decomposition into nearly convex parts is what we call here meaningful segmentation, which is reinforced using two perception rules: the minima rule [HS97] and the short cut rule [SSH99].

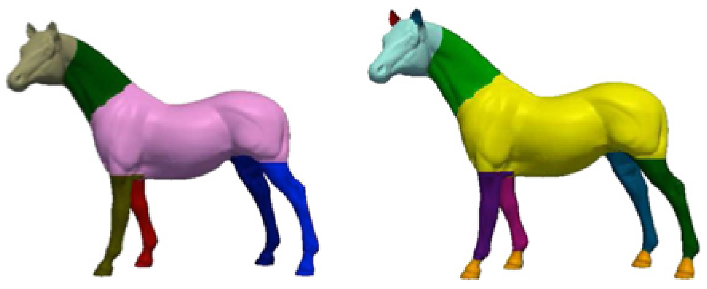


Figure 2.7: ACD based on a measure of relative concavity (picture taken from [GALL13]).

Similarly, Ghosh et al. [GALL13] proposed a fast approximate convex decomposition (FACD) in order to comply with the *quality* of the partitioning of an object into approximately convex components (Figure 2.7). The quality is related to the increasing of naturalness and the minimization of redundancy of components (i.e., reducing over-segmentation), as in MNCD segmentation. The novelty of the FACD segmentation lies in the dynamic programming-based strategy followed to speed up the decomposition of the original object into components, i.e., its *efficiency*. To achieve this goal, the decomposition of components is accomplished in parallel as a n-ary decomposition instead of using a binary decomposition so common in the previous decomposition schemes. Such a n-ary decomposition sustains on the relative concavity measure, instead of the absolute concavity measure, for each cut in order to better quantify the impact of a given cut on the neighborhood of a given concavity in the object.

Using a spectral method, Asafi et al. [AGCO13] decomposes a shape into weakly convex regions. A weakly convex region basically is an approximate convex region as above. This method is based on the notion of weak convexity, which is nothing more than the classical definition of convexity, though relaxed by the notion of inner visibility between points on the surface of a given shape. Interestingly, this method does not use an explicit measure of convexity, hence avoiding any complex operations aiming at measuring the concavity/convexity of regions or parts. The resulting meaningful segmentation consists of regions/parts with high mutual visibility of pairs of points, i.e., the so-called weakly convex regions. As a consequence of its simplicity, this method has the advantage of being also applied to incomplete shapes (i.e., with missing parts), and eventually to point clouds.

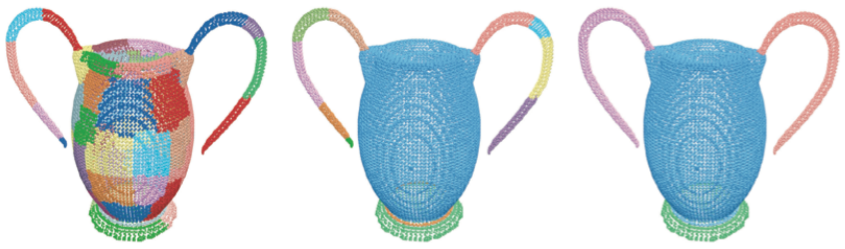


Figure 2.8: ACD based on a measure of weakly convexity (picture taken from [KFK$^+$14]).

Finally, Kaick et al. [KFK$^+$14] improved on the work of Asafi et al. [AGCO13]. Basically, their work describes an algorithm especially designed for the segmentation of point clouds into weakly convex parts, but its also works for meshes and incomplete shapes. In order to cope with over-segmentation, the algorithm includes an additional step to merge neighboring weakly convex segments with similar properties (Figure 2.8). This merging step assumes that identifying similarities between parts is straightforward when such parts are weakly convex. It is clear that the merging step aims at getting a meaningful or natural segmentation of a given shape.

### 2.2.3 Volumetric Meshes

A volumetric mesh of an object consists of a number of volumetric elements of the same type: cubes (or voxels), tetahedra, hexahedra, or, in general, regular polyhedra. One of the first methods of this category of methods is due to Kim et al. [KYL05]. A preliminary step consists in performing the voxelization of the object, as shown in

Figure 2.9. Then, one proceeds to the constrained morphological decomposition (CMD) of the voxelized mesh into voxelized parts [Xu96]. This splitting process is repeated recursively for each part, and is based on a morphological operation, called the opening operation with the ball-shaped structuring element. Finally, one merges adjacent parts if the resulting part/cluster of voxels does not change its weighted convexity relative to their original parts. Once again, this split-and-merge approach aims at obtaining a natural or meaningful decomposition as much as possible.

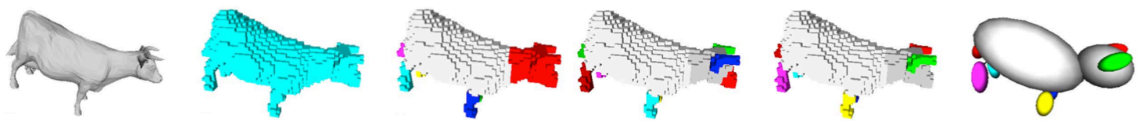


Figure 2.9: VM-based decomposition into voxels (picture taken from [KYL05]).

Instead of using a cubical mesh, Attene et al. [AMSF08] decompose a tetrahedral mesh filling an object into a number of convex parts made up of tetrahedra. As in the CMD process above, the decomposition is accomplished recursively, so that we end up creating a tree of convex polyhedra, whose tree leaves are tetrahedra, intermediate nodes are approximate convex sub-meshes, and its root is the convex hull of the whole mesh. This method automatically stops when the minimum value of concavity is reached in all parts or segments, as illustrated in Figure 2.10.

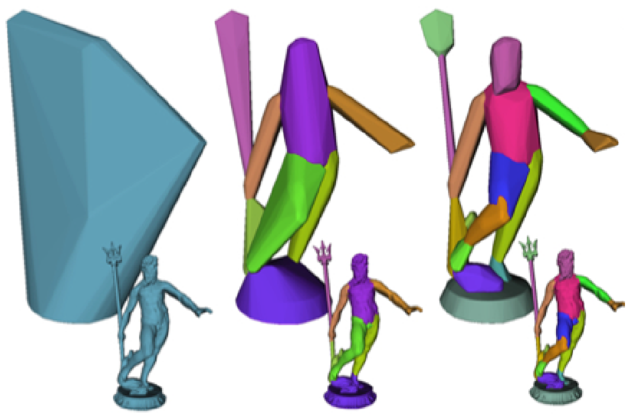


Figure 2.10: VM-based decomposition into tetrahedra (picture taken from [AMSF08]).

Finally, Xian et al. [XGZ11] used a different approach to automatically segment a (tetrahedral and hexahedral) volumetric mesh into semantic parts. First, the 2D boundary surface mesh is decomposed using an improved watershed segmentation algorithm. The resulting 2D segments are used to construct an adjacent graph of the regions or surface features. Second, since the inner side of the surface features occupy some volume space, the interior of the object is also partitioned using the graph cut algorithm, being the partition thus guided by the surface mesh segmentation into semantic features.

### 2.2.4 Space Partitioning

Unlike the previous partitioning schemes for objects, the space partitioning schemes subdivide the space where the object lies, instead of partitioning the object directly. But, it is object's shape that determines how the surrounding space is divided. Space subdivision means here recursive division of the space into 3D regions. Typically, the stopping criterion of the subdivision usually satisfies one of the following conditions: (i) each object part obtained from decomposition must be within a predefined volume threshold relative to its enclosing 3D region; (ii) the level of decomposition attained a pre-defined value; (iii) the number of parts attained a pre-defined value.

Huebner et al. [HRK08] proposed a space partitioning solution that is based on the computation of minimum volume bounding box (MVBB) due to Barequet and Har-Peled [BHP01]. Then, the initial MVBB is recursively split into two parts by a plane that contains the barycenter of the corners and is perpendicular to the longest axis of such box, i.e., we have a binary space partitioning approach. The method stops as soon as the difference of volume between boxes of consecutive levels of the binary tree is negligible. At the end of the hierarchical process, all the 3D segments of the object have a box representation, which can be a good solution to collision detection or even to establish neighborhood relations as needed in geometric reasoning (Figure 2.11). The main drawback of the binary space partitioning technique is that it does not necessarily lead to a meaningful segmentation of the original object. For that purpose, it would be necessary to design a merging procedure of neighbor boxes with the same convexity, somehow.

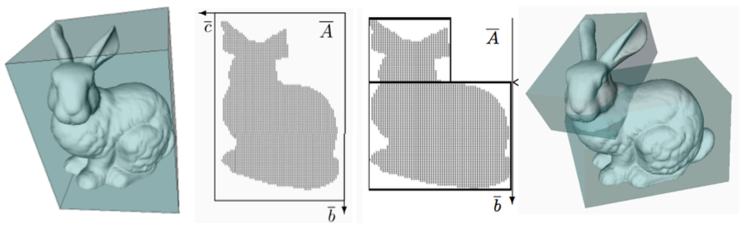


Figure 2.11: Space partitioning-based decomposition using minimum volume bounding boxes (picture taken from [HRK08]).

On the other hand, Simari et al. [SNKS09] introduced a weighted Voronoi partitioning to implicitly create a mesh segmentation. The mesh segmentation is created with a $k$-means algorithm ($k$ is defined by the user) using the Voronoi centers and weights to define each segment. In the case of articulated shapes, to optimize the Voronoi partitioning, a multi-dimensional scaling (MDS) of the object is created by embedding geodesic distances into Euclidean space. Since a traditional Voronoi region is delimited by planar regions, they apply a distance scaling weight to increase the influence of each region center, so allowing for curved boundaries and non-convex regions (Figure 2.12).

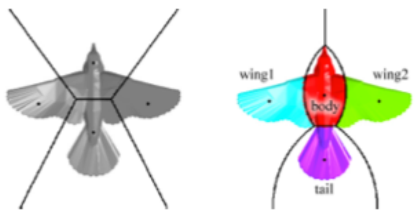


Figure 2.12: Shape partitioning-based decomposition using weighted Voronoi partitioning (picture taken from [SNKS09]).

### 2.2.5 Volume-Based Segmentation: a Discussion

Looking at the timeline in Figure 2.13, we see that most volume-based segmentation algorithms were proposed after 2000. Furthermore, most of them belong to the category of ACD algorithms, which are closer to the theory of human vision. That is, in a way they follow the minima rule. Thus, ACD methods tend to retrieve segmentations that approximate those generated by human perception.

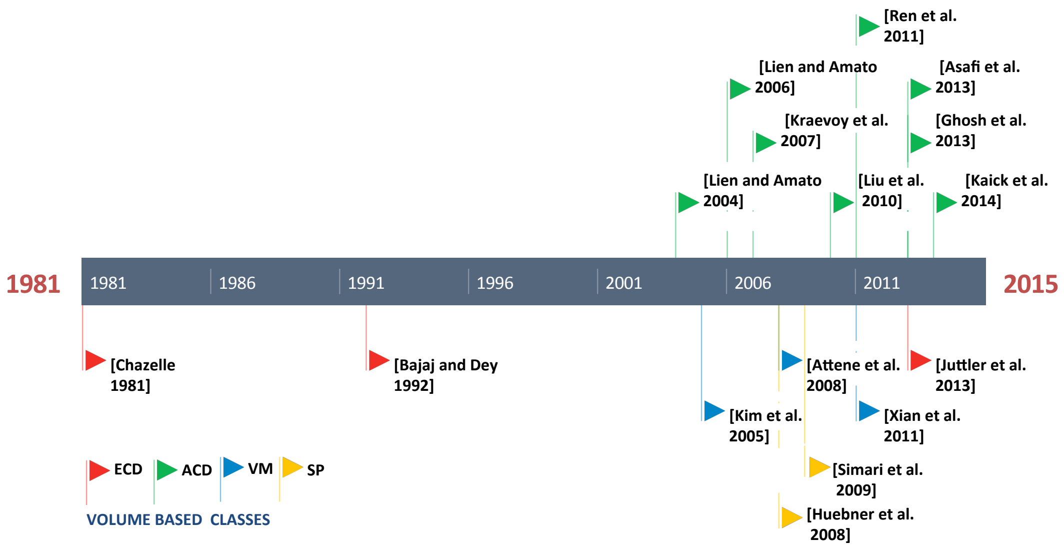


Figure 2.13: Timeline of the volume based algorithms.

Also, a brief glance at Table 2.1 shows the following. The input is mostly a 2-dimensional polygonal mesh, but some methods use solid polyhedra, and more rarely point clouds and tetrahedral/hexahedral meshes. Interestingly, there are several methods applicable to both non-freeform and freeform objects, including articulated objects. In general, the segmentation criterion is based on the concept of convexity, directly or indirectly, because the notion of concavity is a variant of the notion of convexity, yet some methods were elaborated on the basis of the notion of curvature.

Moreover, with the exception of the ECD methods, the segmentations produced by volume-based methods tend to be meaningful. Those ECD methods suffer from over-segmentation. Alias, many other volume-based methods tend to suffer from over-segmentation, but they are able to overcome this problem using a further step that merges neighboring volumes.

Surprisingly, with the exception of ECD methods, most volume-based methods are not automated. This means that a method is not capable of finding the number of segments beforehand or, alternatively, it requires user assistance in terms of input parameters (i.e., a threshold tied to a given stopping condition), or still some sort of user interaction during the process of volume decomposition.

In fact, ACD methods require the user specification of a threshold for the global concavity value of the set of sub-volumes resulting from the object decomposition, and this may have impact on the naturalness of the final segmentation. In the case of VM-based methods, the user specification of the resolution of the lattice of voxels (or, alternatively, tetrahedra or hexahedra) may also influence the course of the decomposition process and, consequently, the naturalness of the final segmentation. In regard to SD-based methods, they are scarce in the literature, but those based on the weighted Voronoi partitioning are potentially automatic, but this needs further research in the near future.

Finally, let us refer that VM-based methods are in general very time-consuming because of the geometric computations incurred in convexity analysis and slicing the input object into sub-objects (or sub-volumes) in a recursive manner.

Table 2.1: Volume-based segmentation methods.

| Method | Reference | Input | | | | Object type | | Class | | | | Criterion | | | | Feature | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | M | PC | S | THM | Ff | NFf | ECD | ACD | VM | SP | Cv | Cc | CH | O | N | OS | A |
| Chazelle | [Cha81] | | | ● | | | ● | ● | | | | | | | ● | | ● | ● |
| Bajaj and Dey | [BD92] | | | ● | | | ● | ● | | | | | | | ● | | ● | ● |
| Lien and Amato | [LA04] | | | ● | | ● | ● | | ● | | | | ● | ● | | ● | | |
| Kim et al. | [KYL05] | ● | | | | ● | ● | | | ● | | ● | | | | ● | ● | |
| Lien and Amato | [LA06] | | | ● | | ● | ● | | ● | | | | ● | ● | | ● | ● | |
| Kraevoy V. | [KJS06] | ● | | | | ● | | | ● | | | ● | | ● | | ● | ● | |
| Attene et al. | [AMSF08] | | | | ● | ● | | | | ● | | | ● | ● | | ● | | |
| Huebner et al. | [HRK08] | ● | | | | ● | ● | | | | ● | ● | | | | ● | | |
| Simari et al. | [SNKS09] | ● | | | | ● | ● | | | | ● | | | | ● | ● | | |
| Liu et al. | [LLL10] | ● | | | | ● | | | ● | | | ● | | | | ● | | |
| Ren et al. | [RYLL11] | ● | | | | ● | | | ● | | | ● | | | | ● | | |
| Xian et al. | [XGZ11] | | | | ● | | ● | | | ● | | | ● | | | ● | | |
| Juttler et al. | [JKP13] | | | ● | | | ● | ● | | | | | | | ● | | ● | ● |
| Ghosh et al. | [GALL13] | ● | | | | ● | | | ● | | | | ● | | | ● | | |
| Asafi et al. | [AGCO13] | ● | ● | | | ● | ● | | ● | | | ● | | | | ● | | |
| Kaick et al. | [KFK+14] | ● | ● | | | ● | ● | | ● | | | ● | | | | ● | ● | |

Abbreviations:
**M**: Mesh; **PC**: Point Cloud ; **S**: Solid/Polyhedron; **THM**: Tetrahedral/Hexahedral Mesh;
**Ff**: Freeform; **NFf**: Non-Freeform or CAD;
**ECD**: Exact Convex Decomposition; **ACD**: Approximate Convex Decomposition; **VM**: Volumetric Meshes; **SP**: Space Partitioning;
**Cv**: Convexity; **Cc**: Concavity; **CH**: Convex Hull; **O**: Other Criterion;
**N**: Naturalness; **OS**: Over-Segmentation; **A**: Automatic;

## 2.3 Surface-Based Segmentation

Surface-based algorithms use 2D meshes as input. Therefore, segments are 2D sub-meshes or regions of a polygonal mesh. There are five main classes of surface-based algorithms, namely:

***Region Growing*** (RG). These methods decompose a mesh into convex polygonal regions. Each region starts with a seed vertex or polygon and grows in the number of clustered polygons until a pre-defined condition or stopping criterion is satisfied. That is, the region keeps growing while a pre-defined condition holds. In other words, if a convexity-related condition is used, the stopping criterion has to do with the violation of such condition (see [CDST95] and [KT96] for further details). What makes a method different from any other one is the criterion used to select the element to be added to a given region. But, there are other features that vary from one algorithm to another; for example, the way how the seed elements are selected, as well as the number of seed elements (multiple region are created in parallel), a seed element per region. This category usually originates over-segmentation, so that almost all region-growing methods consist of two steps, region growing and region merging. The region merging step aims at eliminating the redundancy of segments as much as possible.

***Watershed Segmentation*** (WS). These methods are inspired in the watershed segmentation algorithms first proposed in the field of image segmentation and analysis [Ser82], which in turn are based on the concept watershed of the real world. Watershed segmentation algorithms decompose a given mesh into “catchment basins” or “watersheds”. Traditionally, there are two possible watershed approaches: bottom-up and top-down. The bottom-up approach (also called flooding) aims at finding catchment basins as follows. It starts the flooding process at a local minimum and incrementally floods the region, stopping it as soon as the flooding overflows its catchment basin. The second approach, the top-down approach, concerns to the placement of a token at a certain point and move the token along a steepest descent until it reaches a minimum or a point which has already been associated with a minimum. In this class of algorithms, the process consists in first computing a height function (usually based in the curvature value [PRF01]) at every element (vertices or edges or faces of a mesh) and second identifying the local minima, with each minimum as the seed element of each region or segment. Then, starting from the highest points, the algorithm follows a downward path down the slope, until one of the regions (or its seed element) already formed is found. All elements found along the path are associated to this region. This process is repeated until all elements are associated to a region. Like region growing methods, the watershed algorithms also have a region merging step due to over-segmentation of the mesh.

***Iterative Clustering*** (IC). These methods are based on the well-known $k$-means algorithm due to Loyd [Llo82]. Basically, one starts from a set of $k$ representatives seeds to generate $k$ regions. The segmentation is repeated while at least a new or updated representative seed of a region is calculated in a sequence of iterations. In a way, the segmentation follows a convergence process to an end segmentation.

***Hierarchical Clustering*** (HC). This category includes two types of hierarchical clustering approaches: bottom-up and top-down. In the bottom-up approach, each face is initialized with its own separate cluster. To each cluster is assigned a cost that is used to select the regions to be merged (lowest cost pair). This strategy allows the creation of a hierarchal structure of clusters, with each cluster representing a single region. There are several ways to define a cost associated to each cluster; for example, Garland et al [GWH01] use the planarity as clustering criterion, which can be expressed using a dual form of the quadric error metric, while Attene et al. [AFS06] use a minimum approximation error calculated from a set of of fitting primitives. In the top-down approach, the models are hierarchically and recursively decomposed until one of the following three conditions is reached: (i) each part attained a certain threshold; (ii) a given simplification level is achieved; (iii) a given number of parts is achieved. Note that, as in other categories of methods, the number of resulting clusters is usually unknown beforehand.

***Boundary Segmentation*** (BS). These methods aim at locating the boundary (i.e., a connected sequence of edges and vertices) of each segment, instead of any of its interior elements or seeds (i.e., vertex, edge or face). They normally follow the Hoffman's assertion [HS97], i.e., the human vision delineates the boundaries of object's regions along the negative minima of principal curvatures. However, finding these region boundaries is not an easy task because not always they are not closed contours in the mesh.

### 2.3.1 Region Growing

Seemingly, region-growing algorithms originated in the field of computational geometry; more specifically, they have evolved from the idea of polyhedral surface decomposition, which is due to Chazelle et al. [CDST95]. They proposed a *flooding* heuristic to decompose a polygonal surface into convex regions. The idea of this convex decomposition algorithm consists in traversing the dual graph (whose nodes represent triangles and edges represent adjacency relationships between facets) of the mesh from a seed facet, collecting facets along the way while a convexity-based condition is not violated.

Another flooding algorithm was proposed by Zuckerber et al. [ZTS02], which in a way extends the method of Chazelle et al. [CDST95]. The algorithm adds new faces to a given region while two convexity-based conditions are not violated, the local condition and the global condition. The local condition is violated when an edge on which a face is attached is concave; one uses the computation of the dihedral angle to determine the local convexity of each edge. The global condition is violated if the triangles of the region do not belong to its convex hull, even though the region is locally convex everywhere. The small regions are merged into their neighboring larger regions.

In the algorithm proposed by Zhou and Huang [ZH04], the leading idea is to leverage critical points (i.e., extrema and saddles) of the mesh to decompose it into meaningful parts or regions. After determining such critical points, the algorithm uses a sort of backwards flooding from maxima to saddles —though using Dijkstra's finder to calculate the geodesic distance — to aggregate triangles into regions, as illustrated in Figure 2.14. Similar algorithms were proposed by Katz et al. [KLT05] and Aghatos et al. [APPS10].

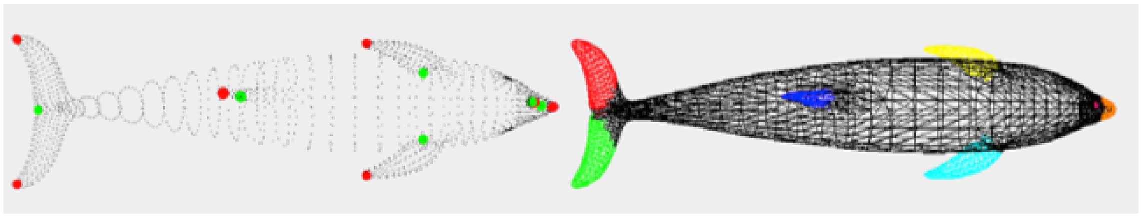


Figure 2.14: Region-growing segmentation using critical points (picture taken from [ZH04]).

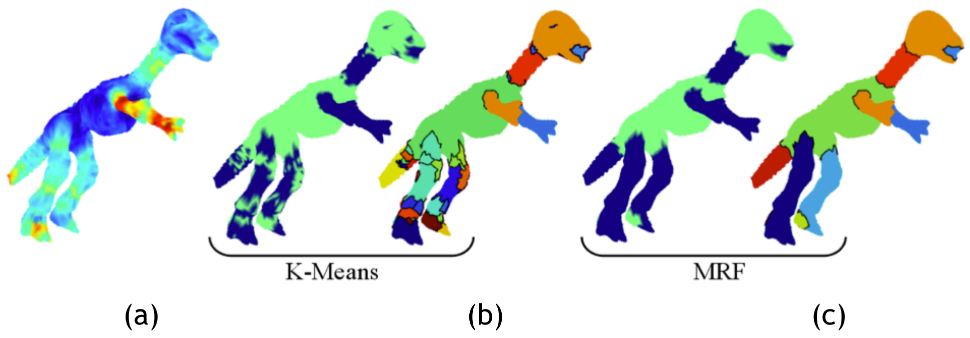


Figure 2.15: Region-growing using Markov random fields (picture taken from [LW08]): (a) curvature scalar field with geodesic radius of 6%; (b) 2-clustering using $K$-means and the resulting region-growing segmentation; (c) 2-clustering using MRF and the resulting region-growing segmentation.

Lavoue and Wolf [LW08] proposed a segmentation algorithm based on Markov Random Fields (MRF). The MRF is a graphical probabilistic model that allows for the integration of different attributes like curvature (computed as the mean curvature integrated over a large geodesic radius), roughness and geometry in the $k$-means clustering step. It is clear that mesh triangles of the same cluster have similar attribute values. After the clustering step, the region-growing segmentation takes place in order to create regions enjoying similar attribute values and smooth boundaries (Figure 2.15).

Another region-growing algorithm was proposed by Zhang et al. [ZLXH08], which consists of five steps, as illustrated in Figure 2.16. The algorithm starts with a preprocessing step to enhance the perception of the mesh feature parts; in practice, this enhancement operator works as a low-pass filter in order to smooth the mesh at hand. Then, one estimates the mean curvature or combined curvature at each vertex. Next, the vertex curvature is mean-shifted iteratively while the inherent error is greater than a given threshold (i.e., stop condition). By using the region-growing approach, the vertices are grouped together with reference to proximity and similar curvature values, with each region growing from a vertex with maximal curvature. In addition, region boundaries are rectified using a minimum-cut algorithm, and thin regions are removed by merging them with adjacent regions.

Bergamasco et al. [BAT11] [BAT12] introduced a semi-supervised region-growing algorithm. The regions grow from an initial set of user-defined seeds. The region growing takes place by propagating region labels over a weighted dual graph of the mesh. The weight associated to each graph edge is calculated using the dot product between the

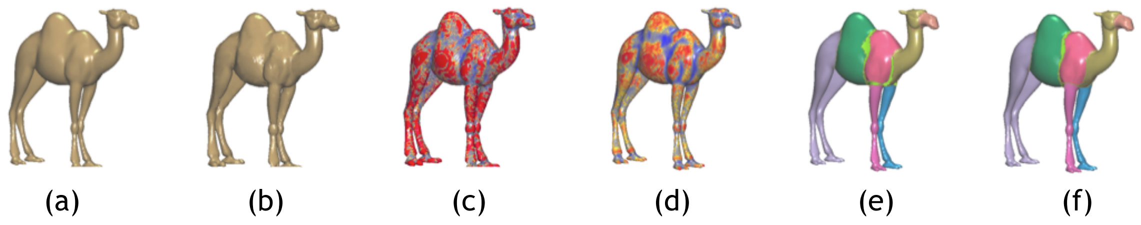


Figure 2.16: Region-growing segmentation using mean-shifted curvature (picture taken from [ZLXH08]: (a) the original mesh; (b) mesh after applying the enhancement filter; (c) enhanced mesh with the curvature map; (d) original mesh after mapping the curvature; (e) region-growing segmentation using proximity and curvature criteria; (f) segmented mesh after boundary rectification.

normals of its incident faces. The labelling follows a greedy approach that takes into consideration the curvature measured between triangles incident on each edge.

The algorithm due to Fan et al. [FL$^+$11] also produces an interactive (user-assisted) mesh segmentation, which is obtained using a progressive painting-based mesh cutting tool, called Paint Mesh Cutting. For that purpose, the user has only to draw a single stroke on the mesh to get the corresponding region on screen in real-time. Furthermore, the user may paint any region of interest using a brush, as is usual in 2D painting tools. This is achieved by means of an efficient local graph-cut based optimization, which is based on Gaussian mixture models (GMM) on the shape diameter function (SDF) as shape metric.

Xiao et al. [XLXG11] proposed a mesh segmentation algorithm specifically designed for CAD mesh models. It makes usage of a clustering approach to firstly divide a given mesh into a dense triangle region and a sparse triangle region. In addition, the sparse triangle region is partitioned into planar, cylindrical, and conical sub-regions, using for this purpose the Gauss map and randomized Hough transformation. In regard to the dense region, its segmentation is carried out applying the mean shift operation on the mean curvature field defined on the mesh.

### 2.3.2 Watershed-Based Segmentation

The *watershed* decomposition algorithms for 3D mesh segmentation was first introduced by Mangan and Whitaker [MW98] [MW99]. This is considered by many the first meaningful segmentation algorithm for 2-dimensional meshes in 3D. This algorithm computes the total curvature at every vertex in order to identify the local curvature minima, each one of which is associated to a watershed. Interestingly, this work was also the first to include the merging step as a way of solving the problem of mesh over-segmentation. Following the same line of research, Zuckerber et al. [ZTS02] proposed a second method that improved the watershed decomposition algorithm due to Mangan and Whitaker [MW98]. But, instead of defining the height function at each vertex of the mesh, they proposed the height function $h = 1 - cos(\alpha)$ over the edges of the mesh. In this case, the over-segmentation is mitigated by merging small regions with their neighbours.

Page et al. [PKA03] also proposed a flooding algorithm, called Fast Marching Watersheds (FMW). This algorithm takes advantage of the human vision theory, more specifically the minima rule. Basically, FMW algorithm calculates the principal curvatures and principal directions at each vertex of a mesh, after which one uses a hill-climbing watershed procedure to identify and aggregate triangles into regions delimited by contours of negative curvature minima. In order to avoid the over-segmentation, the algorithm uses morphology operators to clean up small holes and gaps.

Chen and Geoganas [CG06] proposed another watershed-based segmentation algorithm. In this case, the mesh segmentation requires the preliminary computation of Gaussian curvature and concaveness estimation at each vertex, which altogether determine whether a vertex is convex or concave. In order to get a more accurate computation of the convexity of each vertex, the aforementioned computation is extended beyond the normal 1-ring neighborhood of each vertex, i.e., it is performed for an eXtended Multi-Ring (XMR) neighborhood of each vertex. After this computation for each vertex, the algorithm is ready to go a step forward in order to form the watershed regions, as well as to merge some of those if needed. Indeed, the over-segmentation is mitigated by merging small segments with neighbours possessing the longest boundaries.

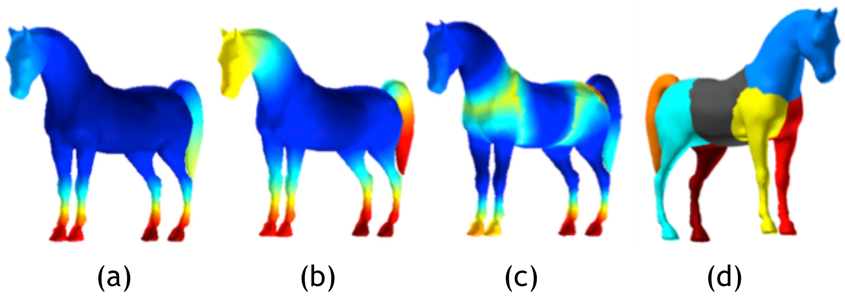


Figure 2.17: Heat diffusion mesh segmentation (picture team from [BPVR11]): (a) heat kernel signature (HKS) function; (b) KL divergence of the points from the uniform distribution; (c) KL divergence of the points from the cluster mean; (d) the final result of the heat diffusion mesh segmentation.

Benjamin et al. [BPVR11] introduced a sort of watershed algorithm, with the difference that one generates a heat flow on the mesh, instead of a water flow. The leading idea is that high curvature points tend to attract heat, and are called heat accumulators, while points in flat regions tend to dissipate heat faster than they receive, being for this called heat dissipators. This heat diffusion algorithm is called Heat Walk, and aims at identifying salient regions in first instance. In fact, the algorithm starts by finding the regions of heat accumulation. The second stage separates dissipative regions —which correspond to approximately planar regions— from those accumulative regions by using a Kullback-Liebler divergence (KL-divergence) based criterion. These two types of regions originates the segmentation of the input triangle mesh (Figure 2.17). Interestingly, this algorithm produces a segmentation that is not noise-sensitive, neither pose-sensitive.

### 2.3.3 Iterative Clustering

Iterative clustering algorithms allow for the creation of $k$ segments from $k$ initial seeds, iteratively. One of the first methods of this class was introduced by Shlafman et al. [STK02]. This algorithm comprises four steps. First, in a preprocessing step, one calculates the distances between all adjacent faces. Second, although a representative face of each region may be chosen randomly, one opted by selecting the initial $k$ representative faces of $k$ clusters or regions, one per region, in order to speed up the convergence to local minima. This aims at maximizing distances between the initial representative faces. Third, each face is assigned to the cluster whose representative is the closest, with the distance between each face and each representative calculated using Dijkstra's algorithm. This originates a segmentation over the mesh. Fourth, new representatives are chosen by minimizing the sum of distances between the faces and their representatives. In a way, this clustering is similar to the $k$-means clustering algorithm.

Seemingly, Liu and Zhang [LZ04] were the first to use spectral clustering techniques in mesh segmentation. This iterative clustering algorithm tends to favor segmentation along minima (i.e., concave boundaries of convex regions). A symmetric affinity matrix (i.e., adjacency matrix of a weighted graph whose nodes are the faces) allows us to encode the probability of a pair of faces being in the same cluster. Besides, one uses the distance matrix put forward by Katz et al. [KT03] to avoid that faces separated by deep concave regions from belonging to same cluster (minima rule). Then, one computes the first $k$ largest eigenvectors of the affinity matrix. These eigenvectors are used to calculate an embedding on a $k$-dimensional unit sphere, which allows us to cluster faces via $k$-means.

Later, Lai el al. [LHMR08] presented segmentation method based on random walks, a concept borrowed from image segmentation [Gra06]. Based on the observation that segments are in general insensitive to the precise location of cluster seeds (representative faces), the algorithm uses a sort of $k$-means clustering algorithm for segmentation. The non-representative faces are then clustered into the previously created regions in function of a similarity criterion. This criterion is based on edge concavity weight cost and edge length, and measures the probability that a random walk starting at each non-representative face reaches each seed face. Furthermore, the algorithm includes a second step, called merging step, that merges neighboring regions sharing similarities.

Fang et al. [FSKR11] proposed a perceptually consistent mesh segmentation (PCMS) that is based on the heat kernel concept. This mesh segmentation algorithm consists of three steps. The first step estimates the number of segments based on the analysis of the Laplacian spectrum, i.e., this is a spectral clustering algorithm. The second step corresponds to the heat center hunting algorithm, which supposedly finds the most representative vertices or centers of segments, one center per segment. Each heat center is defined as a mesh vertex at which the heat mean signature (HMS) has a local maximum. Third, a heat center-driven segmentation scheme produces the PCMS using a $k$-means clustering algorithm, in conformity with human perception, as illustrated in Figure 2.18(a); this segmentation is not sensitive to pose and noise (Figure 2.18(b)-(c)).

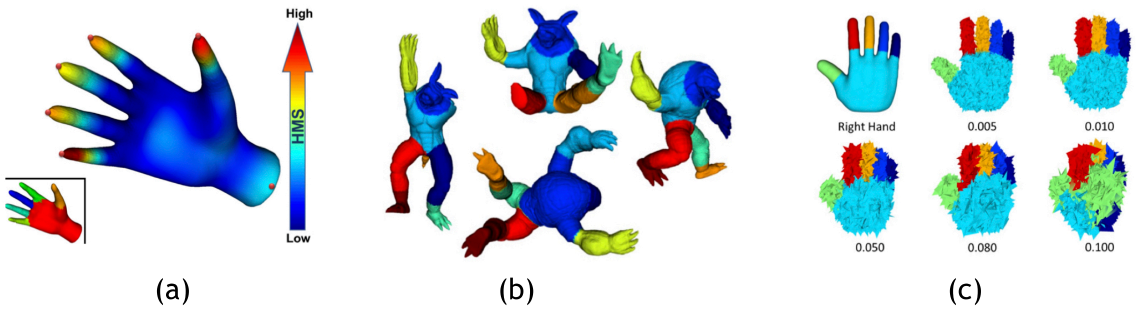


Figure 2.18: Perceptually consistent mesh segmentation (PCMS) based on heat kernel (pictures taken from [FSKR11]): (a) heat mean signature (HMS) of a human hand mesh with maxima at its finger tips; (b) PCMS is not pose-sensitive; (c) PCMS is not noise-sensitive.

Zhang et al. [ZZWC12] developed a variational mesh segmentation inspired in the Mumford-Shah model used in image segmentation, which contains a data term and a regularization term. The data term measures the variation inside each segment, which is evaluated using eigenvectors of the dual Laplacian matrix whose weights are in a way determined by the dihedral angle between triangles incident on edges. The regularization term measures the length of the boundary that separates segments. Therefore, unlike other methods, this spectral method is capable of handling segmentation and boundary smoothing simultaneously. Furthermore, in a pre-processing step, the method estimates the number of segments of the mesh. Computing the number of segments is carried out using a heuristic method based on the stability of the RatioCut values (see Hagen and Khang [HK92] for further details about the RatioCut model).

### 2.3.4 Hierarchical Clustering

As seen above, we have two sorts of hierarchical clustering: bottom-up and top-down. Katz and Tal [KT03] proposed a top-down hierarchical clustering-based segmentation as a way of avoiding over-segmentation and jaggy boundaries between meaningful parts. The hierarchical decomposition of a mesh into sub-meshes starts from an initial decomposition generated from a criterion that combines geodesic distance and distance angular. This allows us to build up a fuzzy decomposition by applying a $k$-means clustering scheme as proposed by Shlafman et al. [STK02]. The fuzzy boundaries (whose triangles may belong to more than one region) of regions are then subject to a graph cut procedure in order to make them exact boundaries (i.e., non-jaggy boundaries), as illustrated by the binary decomposition of a bird in Figure 2.19.

Using a top-down approach, Zhang and Liu [ZL05] proposed a mesh segmentation algorithm based on recursive spectral 2-way cut and Nyström approximation. In fact, the algorithm recursively divides a mesh into two parts using 1-dimensional embeddings in

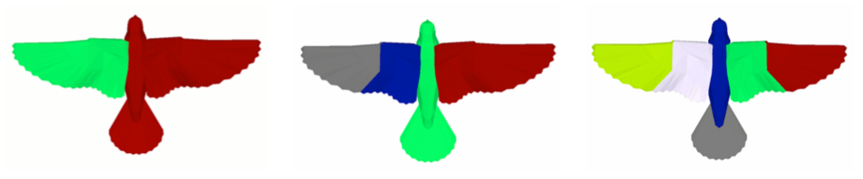


Figure 2.19: Hierarchical clustering-based binary segmentation using $k$-means and graph cut (picture taken from [KT03]).

order to identify the most salient cut, in a way as done for normalized cuts [SM00]; if the salience is below a user defined threshold, the cut is rejected. Unlike Kay and Tal [KT03], Zhang and Liu [ZL05] avoids calculating all pairwise face distances, using for that the Nyström approximation. Since the user can specify a maximum number of segments, the recursive algorithm stops when such number is reached or no more saliences are found. The cuts are obtained without explicit boundary smoothing.

Liu and Zhang [LZ07] extended the work of Zhang and Liu [ZL05]. Their top-down mesh segmentation algorithm also uses a recursive binary decomposition of the mesh into salient sub-meshes. But, it uses spectral projection of the sub-meshes (3D information) onto a plane (2D information) to extract sub-mesh contours in order to perform the segmentation of a given mesh at hand, so that the mesh segmentation problem is reduced to contour analysis. This algorithm also takes advantage of the Nyström approximation in order to speed up the hierarchical mesh segmentation procedure.

Lai et al. [LZHM06] proposed another clustering-based top-down hierarchical segmentation algorithm. Their hierarchical segmentation approach is similar to those due to Katz and Tal [KT03] and Shlafman et al. [STK02]. It differs from those two algorithms in regard to the distance computation, as well as to the use of feature-sensitive multiresolution remeshing, being this remeshing isotropic. The distance metric combines normal curvature, geodesic distance, and statistical measures of local properties such as geometric texture. As usual, the number $k$ of clusters can be determined by the user, or automatically calculated through an optimization technique similar to the one introduced by Katz and Tal [KT03]. The algorithm operates from coarsest level to finer level of mesh segmentation, so originating a hierarchy of regions into sub-regions, as illustrated in Figure 2.20. In the end, the region boundaries are subject to a smoothing procedure, as noticeably shown in Figure 2.20(c).

Attene et al. [AFS06] proposed one of the first bottom-up mesh segmentation algorithms. It is based on the hierarchical face clustering presented by Garland et al. [GWH01]. The leading idea of the algorithm is to initially consider each triangle a cluster (i.e., a sub-mesh), each one of which is approximated by a quadric primitive; for example, cylinder, sphere, and so forth. The algorithm proceeds iteratively merging adjacent clusters, which are once again approximated by the most fitting primitive. The bottom-up approach ends up producing a hierarchy as a binary tree of clusters.

Podolak et al. [PSG$^+$06] proposed a novel top-down hierarchical clustering algorithm

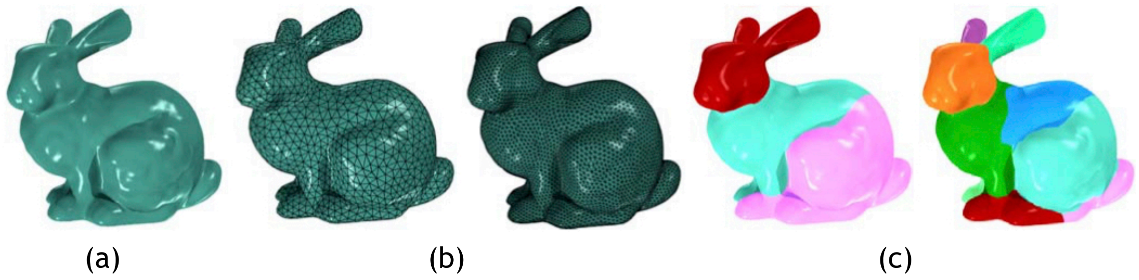


Figure 2.20: Hierarchical clustering based on feature-sensitive multiresolution remeshing (picture taken from [LZHM06]): (a) original mesh; (b) the same mesh after two distinct isotropic remeshing operations; (c) the corresponding segmentations.

for mesh segmentation. This hierarchical clustering is similar to the one introduced by Katz et al. [KT03], but the clustering is not based on a simplification strategy, as in Garland et al. [GWH01], neither on geodesic distance and angular distance between points, as in Katz and Tal [KT03]. Instead, it uses a $k$-means clustering (using $k = 2$ for each split) based on symmetry; more specifically, it is based on the values of local maxima in the planar reflective symmetry transform (PRST). This was made possible because PRST captures not only the global symmetry of a mesh, but also the symmetry of its salient parts. This is so because the triangles of a mesh part or segment have similar symmetries.

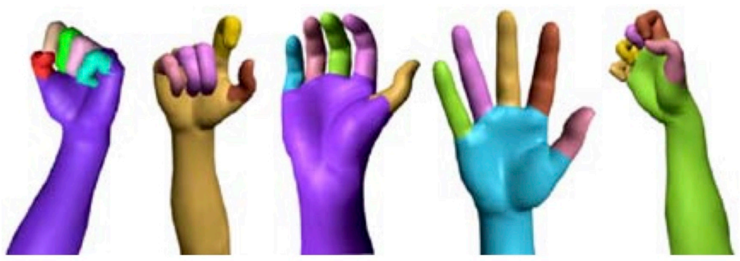


Figure 2.21: Hierarchical clustering based on shape diameter function (SDF) shows that it is not pose-sensitive (picture taken from [SSCO08]).

Another clustering algorithm was proposed by Shapira et al. [SSCO08]. It is based on the shape diameter function (SDF), which measures the diameter of the volume of the object in the neighbourhood of a point of each triangle (e.g., its centroid) belonging to a mesh. SDF is a subtle parameter because points of specific parts of the object, like hands, legs, and so on, have similar SDF values, i.e., SDF can work as a clustering criterion. This means that a cluster may contain multiple mesh parts. The segmentation process consists of two main steps. First, one creates $k$ clusters of triangles with reference to their SDF values, using Gaussian mixture model (GMM) to fit $k$ Gaussians to values of the SDF histogram in order to obtain a probability value for each triangle assigned to each SDF cluster. Second, taking into consideration that a mesh may have parts (e.g., the fingers of a hand) with similar SDF values, one has to proceed to the actual partitioning of each cluster using $k$-way graph-cut to include the probabilistic values of the previous step and local mesh geometric properties (concavity measure by dihedral angles). As a result of the graph cut, one obtains a smooth segmentation into distinct parts, and this is independent of the pose of the mesh, as shown in Figure 2.21.

The algorithm due to de Goes et al. [dGGV08] is capable of constructing a coarse-to-fine hierarchy of clusters or segments based on two concepts: diffusion distance and medial structure. The diffusion distance functions as a multi-scale metric that enjoys very useful properties like robustness to noise, poses, and enhancement of concavities. Therefore, unlike geodesic distance, diffusion distance highlights mesh concavities. In addition, the diffusion distance can be expressed in terms of Laplace-Beltrami operator (LBO), which is known to be invariant to isometries; hence the pose-invariance. On the other hand, a generalization of the medial structures [MPS$^+$04] allows us to segment a mesh by calculating a bijection between medial structures and segments. Note that a medial structure is medial relative to the segments, and in this sense it differs from medial axis. The media structure of a segment is calculated from the average diffusion distance function (ADD), which takes on high values (resp., small values) for boundary and extrema points (resp., points in the middle) of the segment. Taking into account the properties of the diffusion distance, and consequently those of its ADD variant, we easily realize that points with small ADD values are far away from concavities.

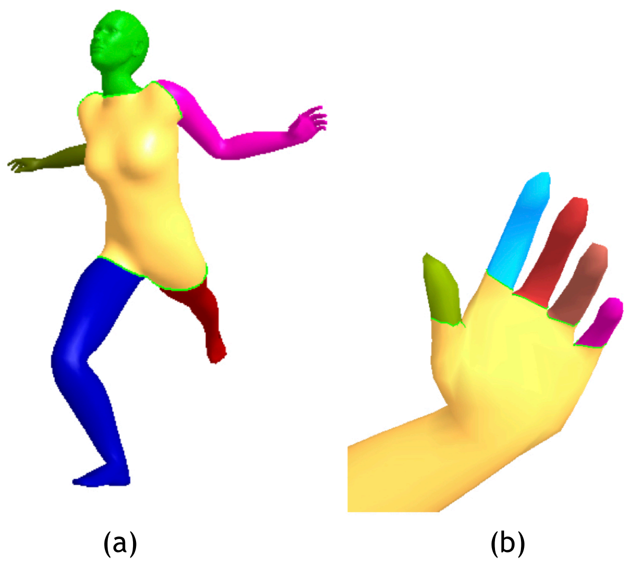


Figure 2.22: Reuter's method for segmentation based on eigenfunctions of LBO at different persistence levels (picture taken from [Reu10]): (a) using only the most relevant critical points of the mesh; (b) using the critical points (maxima at finger tips and saddles between fingers) of the hand.

A similar method was proposed by Reuter [Reu10] to hierarchically segment articulated models using LBO eigenfunctions (whose extrema identify protrusions and saddle points identify concavities) and topological features (level sets at saddles). It follows that Reuter's method is pose-invariant. Moreover, based on topological persistence, it was made possible to build hierarchical representation for the mesh segmentation that remains stable across a family of nearly isometric shapes and with regard to noise or mesh density/quality.

To decompose a mesh into several levels of meaningful components, Ho and Chuang [HC12] also proposed a hierarchical segmentation, in which the salience degree is similar for all regions of the same level, with such salience degree decreasing from root node to leaf nodes. The salience degree is determined by the gradient of the minimum slice perimeter (MSP) function, which provides us with a zonal shape measure of the mesh. It is noteworthy that MSP function is a surface function that represents the volume in the neighborhood of each surface point. Therefore, neighboring surface points with similar volumes tend to be clustered into a region; as a consequence, the gradient of the MSP function detects the variations of volume, which denote the existence of region boundaries. This means that the boundaries on each hierarchical level can be calculated in an automated manner using graph cuts [KT03], though considering measures of curvature and MSP gradient (Figure 2.23).

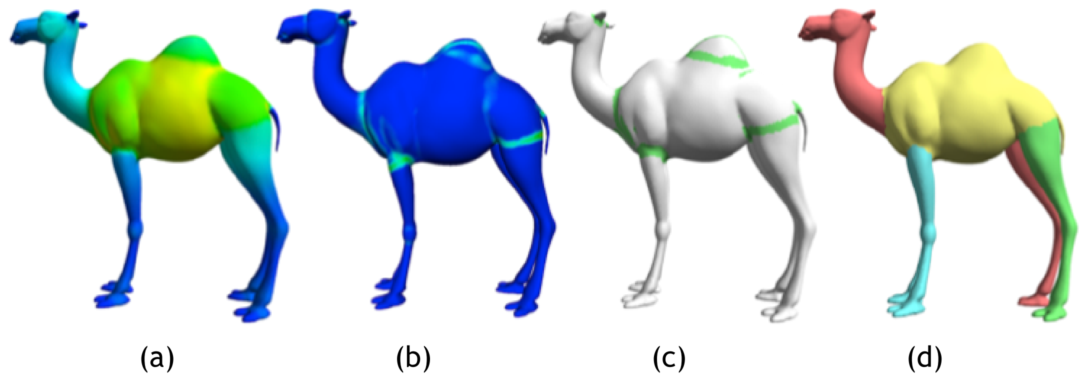


Figure 2.23: Hierarchical clustering based on MSP (picture taken from [HC12]): (a) MSP function on the mesh; (b) MSP gradient on the mesh; (c) region boundaries; (d) mesh regions.

Yan et al. [YWLY12] put forward a mesh segmentation method using quadric surface fitting, so that each segment is approximated by a general quadric surface. To measure the difference between the segments and their corresponding fitting quadrics, one uses a new error metric or function that combines geometric distance and normal derivation. The minimization procedure is based on Loyd's algorithm proposed by [CSAD04]. A distortion-minimizing flooding is used to the partition of the input mesh into a set of non-overlapping and connected regions. Given a new partition of the mesh into regions, we have two alternative steps. The surface fitting is accepted and the mesh partition comes to an end; alternatively, the new mesh partition further minimizes the error function, so that faces with the smallest fitting error are selected as new seed faces for regions. In the end, the jaggy boundary of each region or segment is smoothed in conformity with a graph-cut method.

At last, Zhang et al. [ZLG+15] proposed a hierarchical mesh segmentation based on splat clustering. Splats are mesh charts determined using a variational shape approximation algorithm (VSA) regulated by a metric $\mathcal{L}^2$ or $\mathcal{L}^{2,1}$. Thus, the algorithm essentially consists of two steps. First, an input mesh is decomposed into splats using an improved VSA method based on an optimized $\mathcal{L}^{2,1}$ metric. Second, neighboring clusters exhibiting similar metric values are hierarchically merged in order to create a hierarchy of regions.

As in other segmentation schemes, region boundaries are smoothed using graph cuts.

### 2.3.5 Boundary-Based Segmentation

Boundary-based mesh segmentation methods focus on finding the contours of segments in first instance, instead of clustering triangles into each segment. The main problem faced by these methods is that not always it is possible to find a closed contour for each segment that satisfies the minima rule. This open-contour problem is particularly noticeable in free-from meshes featuring 'natural' models (e.g., animals). Lee et al. [LLS+04] [LLS+05] solved this problem using the shortest path between two vertices that satisfies a specific cost function; consequently, we find a closed contour for each open contour (Figure 2.24). After finding a closed contour, one applies a snake movement, as is usual in image segmentation, in order to optimize such a contour in terms of length and smoothness.

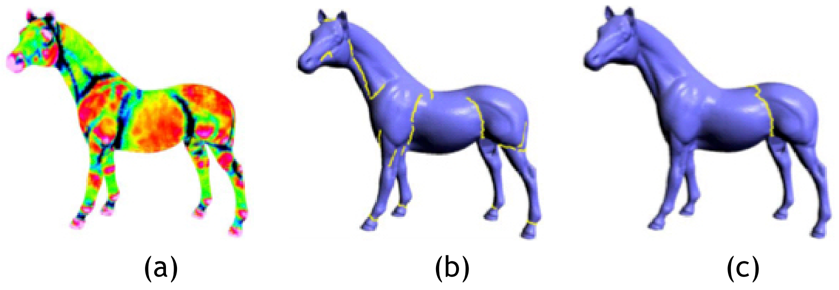


Figure 2.24: Boundary-based segmentation based on curvature and centricity (picture taken from [LLS+05]: (a) minimal curvatures; (b) contour extraction; (c) contour completion.

To select the best boundary, a different approach was implemented by Golovinskiy and Funkhouser [GF08]. They proposed a method called Randomized Cuts that generates multiple segmentations of the same object, which are then combined together with the goal of finding the optimal segmentation, as illustrated in Figure 2.25. In fact, from such set of segmentations, one deduces a partition function that indicates the probability of each edge belongs to a boundary (i.e., a random cut) of the final segmentation. Those segmentations whose boundary edges have the highest scores constitute the set of what we call the most consistent cuts. These segmentations are randomly generated using techniques like $k$-means, hierarchical clustering, and min cuts, but others can be used after all.

An interactive mesh decomposition was proposed by Zheng and Tai [ZT10], according to which the user interacts with the segmentation process by drawing one or more strokes across a desired cut. As a result, the algorithm performs the best cut along a closed isoline of a harmonic field that is generated in a transverse manner relative to one or more brush strokes. Such isolines (thin lines in red in Figure 2.26) are determined by solving a Poisson equation that uses a Laplace operator with a cotangent weighting. It is noteworthy that this segmentation technique is not not sensitive to noise. It is also

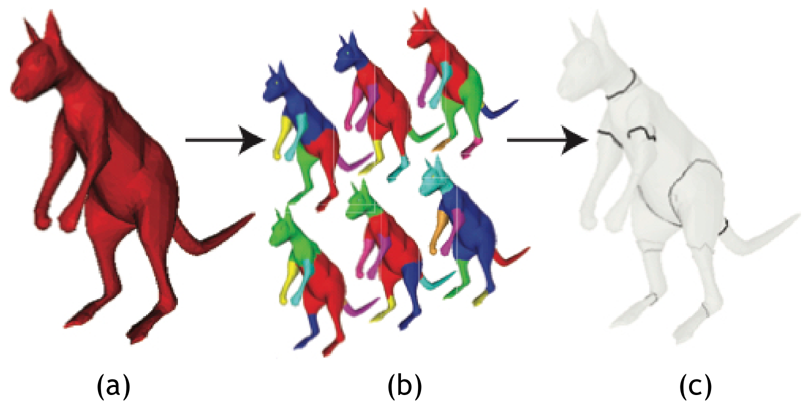


Figure 2.25: Boundary-based segmentation based on randomized cuts (picture taken from [GF08]): (a) input mesh; (b) randomized cuts; (c) partition function.

insensitive to pose variations, as shown in Figure 2.26.

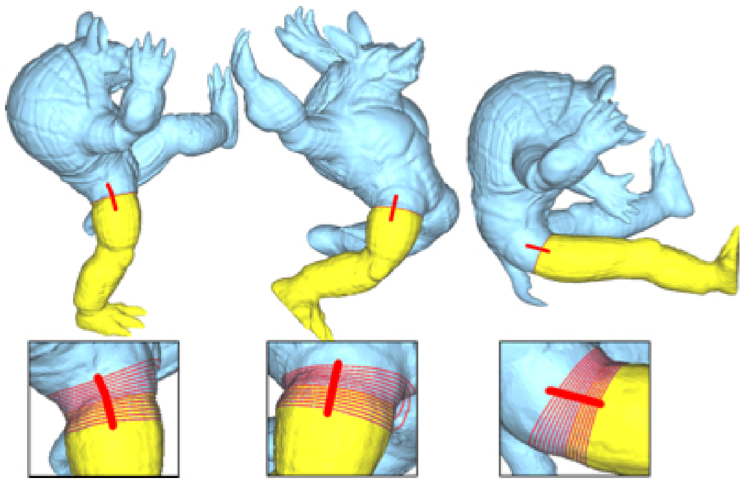


Figure 2.26: Boundary-based segmentation based on brush strokes is insensitive to pose variations (picture taken from [ZT10]).

Benhabilies et al. [BLVD11] presented a segmentation algorithm based on the concept of learning boundary edges. It consists of two major steps: learning and segmentation. The learning step is executed off-line. It uses the AdaBoost classifier that allows for the creation of a boundary edge function from a set of training data (i.e., a set of segmented models). This function operates on an input feature vector to produce a signed scalar value that determines how an edge is classified; more specifically, if the value is positive, the edge is considered as boundary; otherwise, it a non-boundary edge. The edge feature vector builds upon geometric criteria such as curvature, global curvature, dihedral angle, and shape diameter. It is necessary to bear in mind that the learning step produces non-connected fuzzy regions. The second step, also called segmentation step, transforms these fuzzy region boundaries into well-delineated, closed, smooth contours. This requires specific procedures for contour thinning, contour completion, and snake-based smoothing, as illustrated in Figure 2.27.

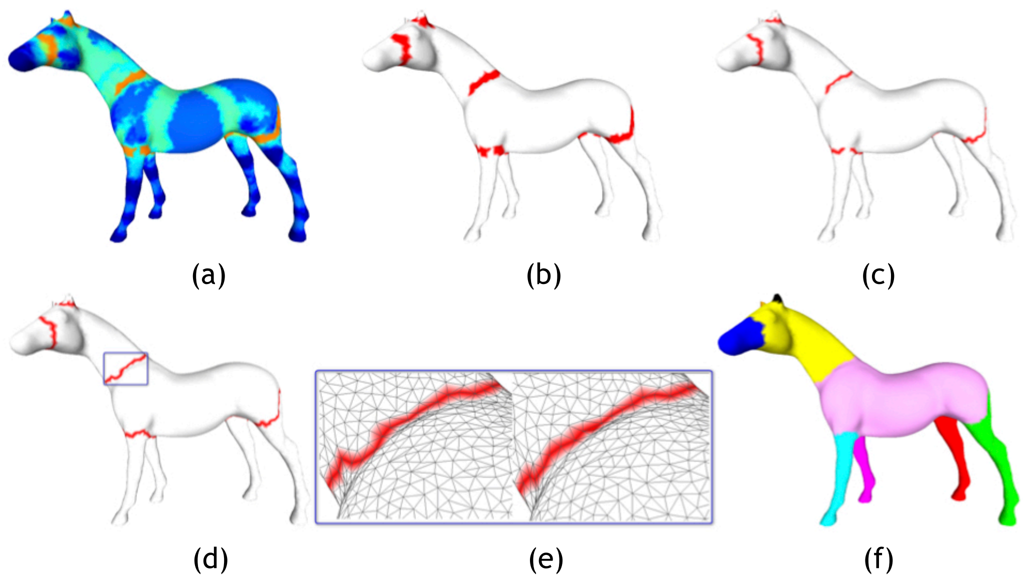


Figure 2.27: Boundary-based segmentation based on learning boundary edges (picture taken from [BLVD11]): (a) edge function; (b) interest region extraction; (c) region thinning; (d) contour completion; (e) snake movement; (f) final segmentation.

Zhang et al. [ZZC11] proposed a constrained random walk algorithm that combines random walks (borrowed from image segmentation [Gra06]) and user interaction (using easy mesh cutting [JLCW06]) to come up with mesh cuts as perceived by humans. The constraints imposed by the user divide into soft constraints and hard constraints. Soft constrains indicate where the cuts should be made nearby, while hard constrains mark the vertices through which the cuts must pass. These constraints are added to random walks in order to find the best mesh cuts. In the end, one uses the shortest path graph to smooth cut contours.

Au et al. [AZC$^+$12] put forward with a fully automatic boundary-based mesh segmentation method, which is exclusively based on shape concavity information of the mesh. This makes usage of a set of concavity-sensitive scalar fields over a mesh in order to locate concave creases and seams, which are good candidate cuts or boundaries between mesh regions. Computing such a scalar field (here called segmentation field) constitute a variant of the surface harmonic field used by Zheng and Tai [ZT10], as described above, requires solving the Poisson equation using the discrete Laplacian operator, as in Desbrun et al. [DMSB99], with a new concavity-sensitive weighting scheme. All concave seams are found using at least one segmentation field over the mesh, with each segmentation field generated in a distinct direction. Interestingly, the isolines of each segmentation field naturally gather at concave seams, so that their high density denotes the existence of a cut or region boundary. The best cuts are found through the scoring of each isoline relative to their field gradient magnitudes. It is also worth noting that the influence of mesh noise influence is mitigated aplying the Gaussian smoothing to the normals as a pre-processing step. Finally, and based on this work, let us also mention that Zheng et al. [ZTA12] proposed an interactive tool (called dot scissor) that combines user interaction with concavity-aware fields to select the best isoline cut.

Wang et al. [WLAT14] introduced a new automatic boundary-based mesh segmentation method, which can be considered as a followup of the Au et al. method [AZC+12] described above, with the difference that the new method only uses a single segmentation field. It comprises two main steps: hierarchical spectral analysis and isoline-based boundary detection. In the hierarchical spectral analysis step, one generates a segmentation field in order to come up with a decomposition of eigenvectors from a Laplacian, i.e., a specific number of eigenvectors is adaptively chosen and split into sub-eigenvectors via spectral clustering. At the sub-eigenvector level, one uses two measures together, inner variations and part oscillations, to assess the confidence in identifying a spectral-sensitive cut (or region contour) for each sub-eigenvector. During the isoline-based boundary detection step, the cuts are identified using a two-step procedure, which consists of a divide-merge algorithm and a cut score to respectively filters and measures isolines, with the goal of filtering and measuring the best isolines as cuts or contours, as illustrated in Figure 2.28.

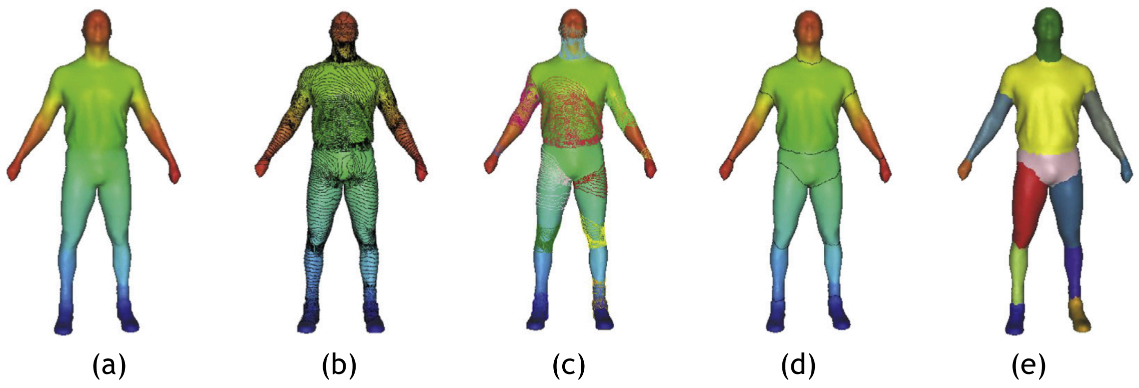
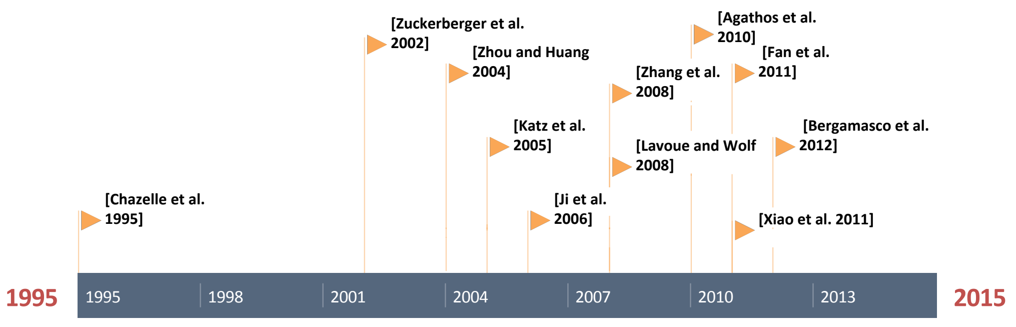


Figure 2.28: Boundary-based segmentation via hierarchical spectral analysis and segmentation field (picture taken from [WLAT14]): (a) single segmentation field; (b) all sampled isolines; (c) isoline grouping; (d) isoline selection; (d) final mesh segmentation.

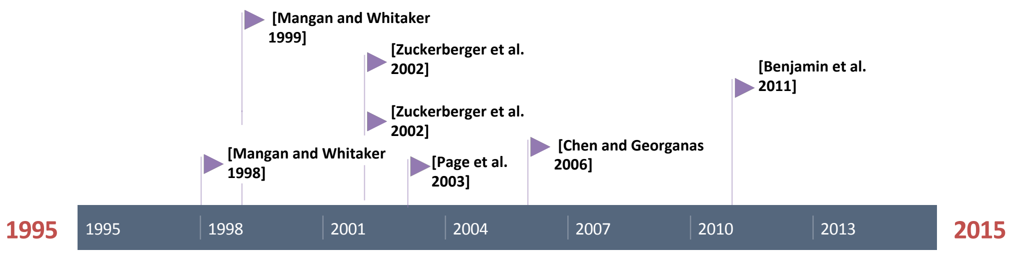
### 2.3.6 Surface-Based Segmentation: a Discussion

The majority of the methods described in this chapter fit the category of surface-based segmentation methods. The timeline of this category of methods is shown in Figure 2.29. Two of its sub-categories made their appearance in 90's, while the remaining three arose after 2000. All of them use a 2-dimensional mesh as input, and it is noticeable that in their majority apply to freeform objects, as well as to non-freeform objects. Usually, the segmentation-guiding criterion is based on convexity or on a convexity variant like concavity, dihedral angle, and so forth.

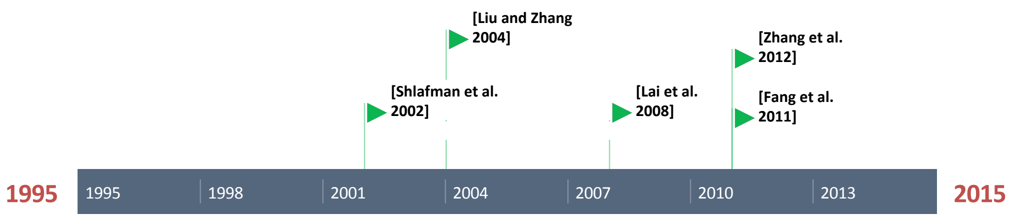
In terms of segmentation results, Table 2.2 (cf. last three columns on the right) shows that generally the methods produce meaningful (or natural) segmentations of a mesh into regions. However, a number of them suffer from over-segmentation, though this problem can be solved, or at least mitigated, using a subsequent region merging step.



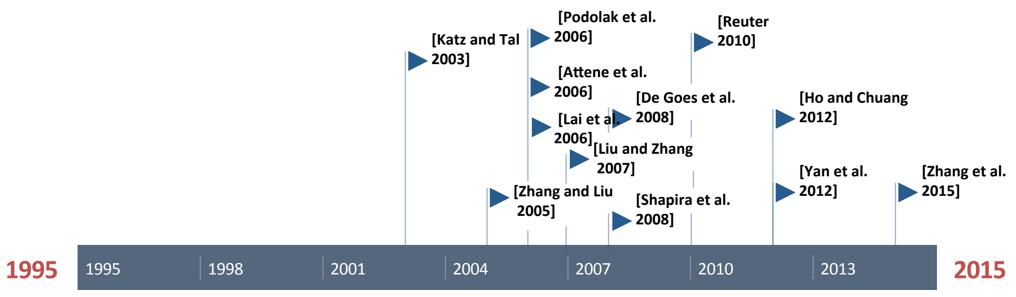
(a) Region Growing.



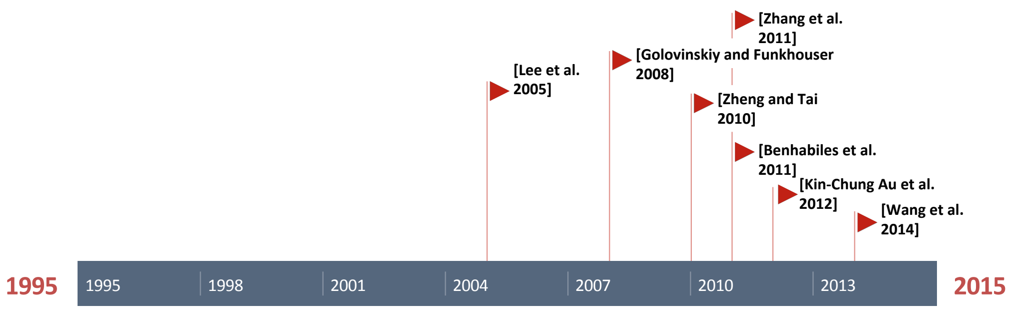
(b) Watershed.



(c) Iterative Clustering.



(d) Hierachical Clustering.



(e) Boundary.

Figure 2.29: Timeline of the surface-based algorithms.

Table 2.2: Surface based segmentation methods.

| Method | Reference | Input | Object type | | Class | | | | | Criterion | | | | | | | | | Feature | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | M | Ff | NFf | RG | WS | HC | IC | BS | Cv | Cc | GD | C | DA | CH | SDF | SA | O | N | OS | A |
| Chazelle et al. | [CDST95] | ● | | ● | ● | | | | | ● | | | | | ● | | | | ● | | |
| Mangan and Whitaker | [MW98] | ● | | ● | | ● | | | | | | | ● | | | | | | ● | ● | |
| Mangan and Whitaker | [MW99] | ● | | ● | | ● | | | | | | | ● | | | | | | ● | ● | |
| Zuckerberger et al. | [ZTS02] | ● | ● | ● | ● | | | | | ● | | | | ● | ● | | | | ● | ● | |
| Zuckerberger et al. | [ZTS02] | ● | ● | ● | | ● | | | | | | | | | | | | | ● | ● | |
| Shlafman et al. | [STK02] | ● | ● | ● | | | | ● | | | | ● | | ● | | | | | ● | | |
| Katz and Tal | [KT03] | ● | ● | ● | | | ● | | | ● | | ● | | ● | | | | | ● | | |
| Page et al. | [PKA03] | ● | ● | ● | | ● | | | | | | | ● | | | | | | ● | ● | |
| Zhou and Huang | [ZH04] | ● | ● | | ● | | | | | | | ● | | | | | | | ● | | |
| Liu and Zhang | [LZ04] | ● | ● | ● | | | | ● | | | ● | | | | | | ● | | ● | | |
| Katz et al. | [KLT05] | ● | ● | | ● | | | | | | | ● | | | ● | | | | ● | | |
| Lee et al. | [LLS+05] | ● | ● | ● | | | | | ● | | | | ● | | | | | ● | ● | | |
| Zhang and Liu | [ZL05] | ● | ● | ● | | | ● | | | | ● | | | | | | ● | | ● | | |
| Ji et al. | [JLCW06] | ● | ● | ● | ● | | | | | | ● | | ● | | | | ● | | ● | | |
| Attene et al. | [AFS06] | ● | ● | ● | | | ● | | | | | | | | | | | | ● | | |
| Chen and Georganas | [CG06] | ● | ● | ● | | ● | | | | | ● | | ● | | | | | | ● | ● | |
| Lai et al. | [LZHM06] | ● | ● | ● | | | ● | | | | | ● | ● | | | | | | ● | | |
| Podolak et al. | [PSG+06] | ● | ● | ● | | | ● | | | | | | | | | | | | ● | | |
| Liu and Zhang | [LZ07] | ● | ● | ● | | | ● | | | ● | | | ● | | | | ● | | ● | | |
| Shapira et al. | [SSCO08] | ● | ● | | | | ● | | | | ● | | | ● | | ● | | | ● | | |
| De Goes et al.] | [dGGV08] | ● | ● | | | | ● | | | | | | | | | | ● | | ● | | |
| Golovinskiy et al. | [GF08] | ● | ● | ● | | | | | ● | | ● | ● | | ● | | | | | ● | | |
| Lai et al. 2008 | [LHMR08] | ● | ● | ● | | | | ● | | | ● | | | | | | | ● | ● | ● | |
| Lavoue and Wolf | [LW08] | ● | ● | ● | ● | | | | | | | | ● | | | | | ● | ● | | |
| Zhang et al. | [ZLXH08] | ● | ● | | ● | | | | | | | | ● | | | | | | ● | ● | |
| Agathos et al. | [APPS10] | ● | ● | | ● | | | | | | | ● | | | | | | | ● | | |
| Reuter | [Reu10] | ● | ● | | | | ● | | | | ● | | | | | | ● | ● | ● | | |
| Zheng and Tai | [ZT10] | ● | ● | ● | | | | | ● | | ● | | | | | | ● | | ● | | |
| Benhabiles et al. | [BLVD11] | ● | ● | ● | | | | | ● | | | | ● | ● | | ● | | | ● | | |
| Benjamin et al. | [BPVR11] | ● | ● | | | ● | | | | | | | ● | | | | | | ● | ● | ● |
| Fan et al. | [FL+11] | ● | ● | ● | ● | | | | | | | | ● | | | ● | | | ● | | |
| Fang et al. | [FSKR11] | ● | ● | ● | | | | ● | | | | | | | | | ● | ● | ● | | ● |
| Xiao et al. | [XLXG11] | ● | | ● | ● | | | | | | | | ● | | | | | | ● | | |
| Zhang et al. | [ZZC11] | ● | ● | ● | | | | | ● | | | | ● | | | | | | ● | | |
| Ho and Chuang | [HC12] | ● | ● | ● | | | ● | | | | | | ● | | | | | ● | ● | | |
| Kin-Chung Au et al. | [AZC+12] | ● | ● | ● | | | | | ● | | ● | | ● | | | | ● | | ● | | ● |
| Bergamasco et al. | [BAT12] | ● | ● | ● | ● | | | | | | | | ● | | | | | ● | ● | | |
| Yan et al. | [YWLY12] | ● | | ● | | | ● | | | | | | | | | | | ● | ● | | |
| Zhang et al. | [ZZWC12] | ● | ● | ● | | | | ● | | | ● | | | ● | | | ● | | ● | | ● |
| Wang et al. | [WLAT14] | ● | ● | ● | | | | | ● | | ● | | | | | | ● | | ● | | ● |
| Zhang et al. | [ZLG+15] | ● | ● | ● | | | ● | | | | | | | | | ● | | | ● | | |

Abbreviations:
**M**: Mesh;
**Ff**: Freeform; **NFf**: Non-Freeform or CAD;
**RG**: Region Growing; **WS**: Watershed Segmentation; **HC**: Hierarchical Clustering; **IC**: Iterative Clustering; **BS**: Boundary Segmentation;
**Cv**: Convexity; **Cc**: Concavity; **GD**: Geodesic Distance;**C**: Curvature; **DA**: Dihedral Angle; **CH**: Convex Hull; **SDF**: Shape Diameter Function;
**SA**: Spectral Analysis - Laplace-Beltrami Operator; **O**: Other Criterion;
**N**: Naturalness; **OS**: Over-Segmentation; **A**: Automatic;

Similar to volume-based methods, perhaps the main issue inherent to surface-based methods has to do with the lack of an automated procedure to decompose a mesh into a number of meaningful regions. In fact, knowing such a number of segments in advance is not an easy task. The usual solution is to ask the user for such a number or, alternatively, ask the user to interactively select the region seeds on the mesh, or still to specify a specific threshold for the stopping condition of the mesh decomposition. A too fine threshold may originate over-segmentation of the mesh.

Seemingly, there is not any automated region-growing method in the literature (cf. Table 2.2) that is capable of finding the number of region seeds beforehand, as well as their number; as a consequence, the segmentation may result unnatural. Analogously, in general, watershed-based methods do not produce segmentations in an automated manner; the exception is the one due to Benjamin et al. [BPVR11], who used the concept of heat flow instead of water flow over the mesh, but their segmentations tend to be more coarse than human segmentations, and are not adequate for non-freeform objects.

Similarly, the main issue in iterative clustering-based methods, which are essentially based on some $k$-means clustering procedure, is indeed to determine the value of $k$ (i.e., the number of mesh regions) beforehand. In addition, it is necessary to well choose the representative centroids of regions, which is not an easy task either, though some methods use spectral information (see, for example, [FSKR11] [ZZWC12]) to guide the iterative clustering process, while others use a user input to get the initial representatives of each cluster or mesh region.

In regard to the hierarchical clustering class, and looking at Table 2.2, it seems that there is no method that automatically calculates the number of regions or clusters *a priori*. Recall that, there are two main approaches for hierarchical clustering: top-down and bottom-up. In the top-down approach, an important issue to known when it is time to stop the generation of new levels of segmentation; otherwise, there is a risk of producing over-segmentation at the deepest level of cluster hierarchy. In the bottom-up approach, the main difficulty lies in achieving a good merging criterion regions into larger regions in a meaningful manner.

Finally, boundary-based methods build upon the minima rule —from the theory of human vision—, which states that boundaries between regions are located in concave paths. Therefore, these methods aim at finding such regions from their boundaries, not from interior seeds. The main difficulties in using these methods are threefold. First, it is common to have various boundaries separating two regions, so a decision has to be taken about the most appropriate one; for example, there are various closed paths around the neck that separates head from trunk of a mesh representing a human body. Second, finding a closed boundary or loop of concave edges separating two regions is not always feasible. Nevertheless, a few methods have overcome this problem; see, for example, [LLS$^+$04]. Third, with the exception of [AZC$^+$12] [WLAT14], these methods are not accompanied by a technique to automatically determine the number of regions.

## 2.4 Skeleton-Based Segmentation

Generating a skeleton (also called curve-skeleton) is a process known as skeleton extraction or skeletonization [LWTH01] [CGC$^+$02] [KT03] [LA04]. Let us first recall that a skeleton is a 1-dimensional structure that captures the shape (i.e., geometry and topology) of a given object in a simplified manner. This technique has been used in several research fields and applications, in particular in reconstruction [ABK98], virtual navigation [WDK01], animation[CGC$^+$02], segmentation [LWTH01], etc. The reader is referred to Cornea et al. [CSM07] for a more detailed review on this topic.

There is a number of ways of constructing skeletons from a mesh, namely:

***Medial Axis*** (MA). The medial axis skeleton is based on medial axis transform (MAT) [Blu67] [SP08]. The medial axis of a 2-dimensional mesh in 3D can be seen as the locus of the centers of balls that are tangent to and maximally inscribed in the mesh, i.e., such balls contained in the mesh are tangential to three or more points of the mesh.

Therefore, the MAT of a given 2-dimensional mesh can be defined as the medial axis together with the radius function that determines those maximally inscribed balls. It is hard to compute MAT exactly. To simplify this process, several strategies are used to construct the skeleton based on the medial axis definition. Amenta et al. [ACK01] computed the Voronoi diagram to approximate the medial axis skeleton. Siddiqi et al. [SBTZ02] compute medial axis approximations across 3D voxel grids. The shock graph is a variant of the medial axis [SSDZ98] [CK01], which is obtained by viewing the medial axis as the locus of singularities (shocks).

***Reeb Graph*** (RG). A Reeb graph is a 1-dimensional structure whose nodes are critical points of a real-value function defined on a given surface. The skeleton is obtained by embedding the Reeb graph into the geometry. Essentially, a Reeb graph tracks the topology changes in the level set defined by a scalar function. The graph is obtained by contracting the connected components of the level set to points. Different real-value functions can be used such as geodesic function [HSKK01], shape convexity [LKA06], Morse functions [ABS01] [SKK01] or harmonic functions [AHLD07].

***Geometric Contraction*** (GC). A contraction skeleton is the result of progressively contracting a given shape to its 1-dimensional skeleton. There are two main strategies to get this 1-dimensional skeleton: mesh contraction (MC) and volumetric contraction (VC). Mesh contraction techniques operate directly on the 2-dimensional mesh, which is iteratively (smoothed and) contracted into a nearly zero-volume mesh that retains its global shape in terms of geometry and topology [ATC+08] [CTO+10] [TAOZ12] [YHN14]. Volumetric contraction techniques are predominant in skeleton extraction. These volumetric techniques use a volumetric discrete representation, either a voxelized representation or a discretized distance field. The voxelized representation is produced from the mesh, so that the sleketon is then obtained through a thinning procedure that peels the boundary voxels though preserving the topological shape of the original mesh. Distance-field methods make use of a distance transform (or other functions like the radial basis functions) to get a set of interior voxels that approximate the medial axis of the mesh.

Let us now detail the methods of each one of these categories.

### 2.4.1 Medial Axis-Based Segmentation

Seemingly, Mortara et al. in [MPS+04] were the first to segment a mesh into tubular features (cylinders or cones) and bodies using a sort of medial axis; this method is called Plumber. The skeleton is constructed by intersecting a radius-varying sphere with each tubular part of the mesh (Figure 2.30(a)), in order to obtain two closed curves, called medial loops (in red in Figure 2.30(b)). The barycenters of these two medial loops define a segment of the approximate medial axis. Then, the sphere is swept backwards and forwards to position it at each barycenter, so that further medial loops are calculated once again. With the barycenters of the medial loops, one builds the skeleton lines, from which it is possible to identify the tubular features in the segmentation step; the remaining parts are considered as bodies. This technique was latter used by Mortara et al. [MPS06] to segment human models. This method is efficient for objects having elongated features.

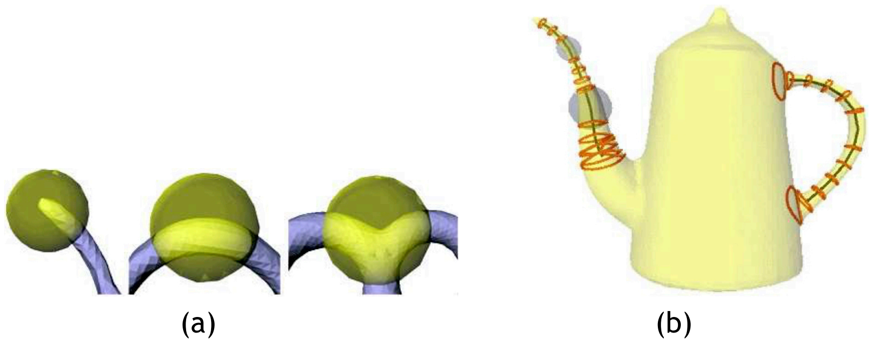


Figure 2.30: Plumber method to segment tubular features (picture taken from [MPS$^+$04]): (a) calculating medial loops by intersection between the mesh and a radius-varying sphere; (b) medial loops in red around tubular features.

Another medial axis-based method in mesh segmentation was proposed by Sharf et al. [SLSK07], which produces natural segmentations of objects. Its admissible input is a polygonal mesh or even a point cloud of articulated objects (namely human bodies). The method is based on a deformable model evolution in a way similar to how an incremental tracking triangulation (i.e., continuation-based triangulation) evolves from a front to several fronts on a point cloud or a surface, eventually with some merging in the meanwhile. Therefore, starting from a seed point of the cloud (or a vertex of an extant triangulation), this deformable model includes multiple fronts that reconstruct the fine features of the shape, as illustrated in Figure 2.31. Each front is here seen as a connected set of vertices of the mesh moving tangentially to the mesh outwards. Eventually, when the front vertices move outwards on the growing mesh, the front divides into connected sub-fronts —as those fronts concerning the hand fingers in Figure 2.31—, so that each sub-front ends up evolving separately. In terms of skeleton, this requires adding a new skeleton node per sub-front to the node of the original front.

### 2.4.2 Reeb Graph-Based Segmentation

Seemingly, Xiao et.al [XSW03] presented one of the first Reeb graph-based segmentation methods. This method uses a discrete Reeb graph (DRG), which is a variant of the Reeb graph as applied to unorganized point clouds, in particular unorganized point clouds of human body scans. Essentially, this method aims at detecting the critical nodes of the DRG concerning maxima, minima, and saddles, which identify the existence of extremities and special points of the human body (e.g., hand tips and armtips). By using DRG, this segmentation method is capable of extracting the six parts of the human body, but it is only able to recognize moderate variations of the standard posture, though it proved to be not sensitive to noise, irregular point sampling, and holes. Nevertheless,

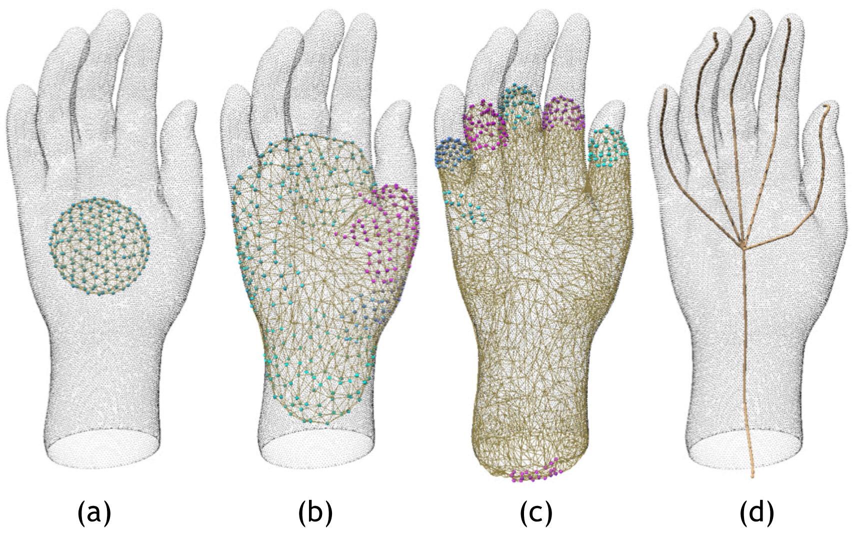


Figure 2.31: On-the-fly curve-skeleton computation using mesh fronts (picture taken from [SLSK07]): (a) mesh front expansion from a seed point; (b) a mesh front divides into two (including the front for thumb finger); (c) a mesh front divides into five (including those concerning the index, middle, ring, and pinky fingers); (d) the curve-skeleton for a hand.

Werghi et al. [WXS06] optimized this method using geodesic distance to build the DRG, with the intent of making it insensitive to different human body postures.

Tierny et al. [TVD07] proposed a hierarchical segmentation method based on Reeb graph. The method starts by determining an enhanced topological skeleton of the input mesh, as described in [TVD06]. This enhanced skeleton requires the computation of feature points, which amount to extremities of features (e.g., finger tips) of the mesh. Basically, this enhanced skeleton involves the construction of the Reeb graph upon the geodesic distance from each mesh vertex to the closest feature point. From this enhanced skeleton, one determines the boundary of each feature, as well as the hierarchy of features. They also use a region merging step to create a more natural segmentation from the human vision point of view, as shown in Figure 2.32.

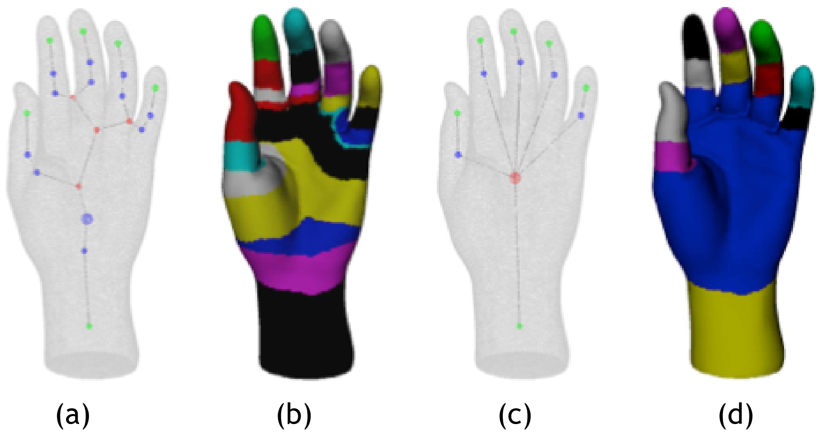


Figure 2.32: Enhanced topological skeleton-based segmentation (picture taken from [TVD07]): (a) enhanced topological skeleton; (b) raw segmentation with each patch referring to a skeleton node, resulting in an over-segmentation of the mesh; (c) simplified graph; (d) natural segmentation.

Another automatic Reeb graph-based segmentation algorithm was proposed by Berretti el al. [BDBP09], which is perceptually consistent with the human vision principles (i.e., minima rule). This means that the decomposition of mesh into salient regions is perceptually meaningful. The Reeb graph is here used to get structural and topological information of the input mesh, as well as to guide the decomposition process. The boundaries of mesh regions are subject to a curvature-based refinement in conformity to the minima rule. Thus, the mesh segmentation involves a two-step procedure in relation to Reeb graph: construction and refinement. Unlike the method due to Tierny et al. [TVD07], in which geodesic distances are calculated with reference to feature points of the mesh, the average geodesic distance (AGD) is here used as the function that induces the construction of the Reeb graph. More specifically, each mesh vertex is assigned the value of the AGD from it to all other vertices. Then, one uses a sort of non-uniform quantization of this function that takes into account the critical points and the number of components of each level set of such function, in order to get a more accurate identification of critical points, which unveil descontinuities tied to region boundaries, and at the same time to prevent over-segmentation resulting from an excessive number of quantization levels. The refinement of the region boundaries is performed using principal curvatures.

Finally, Aleotti and Caselli [AC12] also put forward a Reeb graph-based segmentation method, specifically designed for robot grasping. Recall that the Reeb graph allows us tracking the topological changes across successive level sets of a scalar function defined over a given mesh. This graph is determined by contraction of each component of each level set to a point. This method uses integral geodesic distances to build up the Reeb graph (Figure 2.33), whose implementation is the one proposed by Hilaga et al. [HSKK01]; this implementation is based on Dijkstra's algorithm.

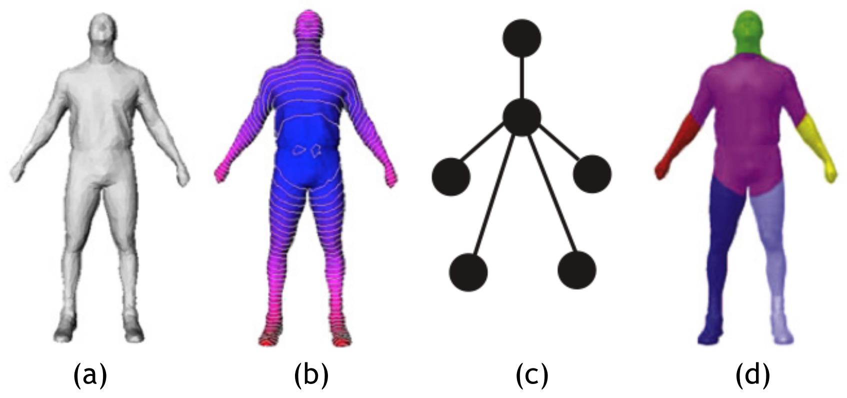


Figure 2.33: Reeb graph-based method using integral geodesic distance (picture taken from [AC12]): (a) the original mesh; (b) the level-sets of the integral geodesic distance function as curves on the mesh; (c) the Reeb graph; (d) the segmented mesh.

### 2.4.3 Mesh Contraction-Based Segmentation

Seemingly, the first skeleton-based segmentation method was proposed by Li et al. [LWTH01], in a way inspired in the work of Tan et al. [TCL99]). In this case, the mesh contraction is performed through an edge contraction-based simplification procedure, i.e., an edge collapse operator, which is a variant of the edge collapse operator due to Hope [Hop96]. The result of applying this edge collapse operator is a set of not necessarily connected edges and vertices, which will form a connected skeleton after adding virtual edges to it, what is done using a procedure similar to that one described by Verroust and Lazarus [VL00]. After concluding this skeletonization step, it follows the segmentation step.

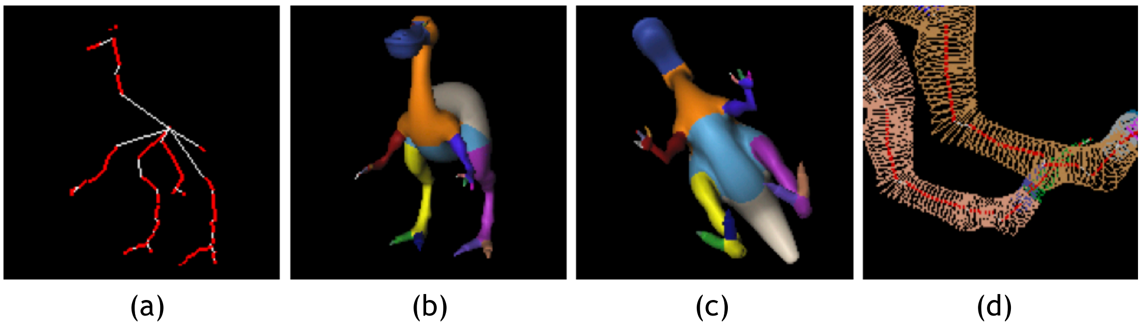


Figure 2.34: Mesh contraction-based segmentation using edge collapse (picture taken from [LWTH01]): (a) the skeleton of a dinopet; (b)-(c) the resulting segmentation; (d) the loops that result from intersecting mesh features with a sweeping plane perpendicular to skeleton branches.

This second step sweeps a plane perpendicular to each skeleton branch in order to identify critical points of each branch, i.e., the cutting loops (or cuts) on the mesh (Figure 2.34). The segments are implicitly created from these cuts.

Another algorithm based on mesh contraction was proposed by Au et.al. [ATC$^+$08]. The contraction of the mesh into a zero-volume skeletal shape is performed using implicit Laplacian smoothing, but with global positional constraints, so that vertices are moved along their curvature normal directions approximately. Therefore, this inwards smoothing procedure guarantees that the final skeleton is topologically equivalent to the original mesh (Figure 2.35), and at the same time reduces reduces the impact of noise. Additionally, the method is insensitive to pose and is rotation-invariant because it operates directly on the geometry of the mesh. This iterative smoothing process stops when the volume is nearly zero. But, unlike the method introduced Li et al. [LWTH01], the contraction keeps the mesh connectivity, not being thus necessary to reconstruct the sleketon.

### 2.4.4 Volumetric Contraction-Based Segmentation

It is noteworthy that volumetric contraction-based mesh segmentation methods require the voxelization of their input mesh. In the particular case of Raab et al.' method

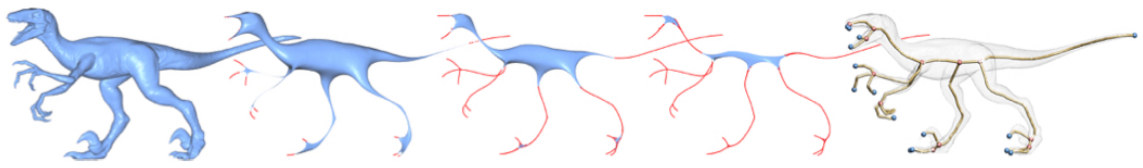


Figure 2.35: Mesh contraction-based segmentation using Laplacian smoothing (picture taken from [ATC+08]).

[RGS04], the voxelization is performed for the axis-aligned bounding box that encloses the input mesh, so that the voxels divide into outside voxels, boundary voxels, and inside voxels. Then, one defines a distance field on the inside voxels relative to boundary voxels, so that the medial voxels are the deepest interior voxels. These medial voxels form the discrete medial surface (DMS); a medial voxel is assigned a distance field value that is greater than the one of most of its 26 neighbor voxels. As a result, one obtains a preliminary, non-connected skeleton. In order to get the connected skeleton, one uses standard Dijkstra shortest pathfinder between medial voxels.

In the case of Brunner-Brunnett's method [BB04], the medial voxels are determined using a voxel peeling strategy, which starts with the removal of the boundary voxels, a process that is repeated for each external shell of interior voxels, in a way like the Matryoshka doll (Figure 2.36). It is clear that the curve-skeleton is then produced from the discrete medial skeleton. Then the junction points of the skeleton are used as reference points to segment the meshes through a function that determines the distance between the barycenter of each triangle and the skeleton.

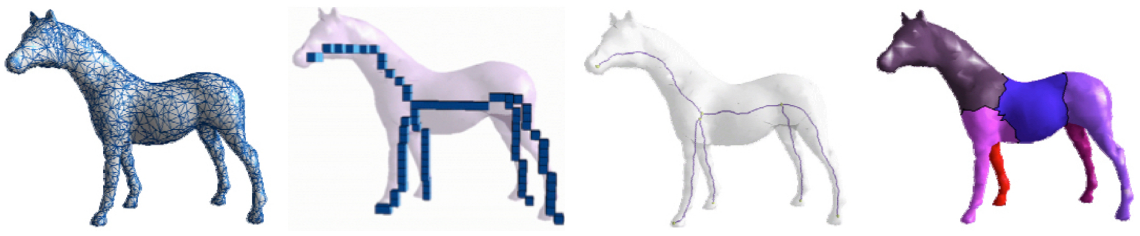


Figure 2.36: Volumetric contraction-based segmentation using voxel thinning (picture taken from [BB04]): (a) the original mesh; (b) the discrete medial skeleton obtained using voxel thinning; (c) the curve-skeleton; (d) the final mesh segmentation.

Another skeleton-based segmentation was proposed by Lovato et al. [LCG09], in particular for human bodies and the like. This algorithm incorporates a skeleton extraction method that takes advantage of voxel coding and active contours. For that purpose, the mesh has to be discretized in a 3-dimensional grid of voxels, after which the boundary voxels of the shape (i.e., those intersection the mesh triangles) are retrieved. The remaining voxels inside the 2-dimensional mesh are assigned a value that denotes the distance of each inside voxel to the closest boundary voxel; as a consequence, one obtains a "distance field from boundary" (DFB) map for the voxelization of the mesh. Therefore, the medial voxels are those the highest distances from boundary. However, Zhou and Toga [ZT98] showed that this medial procedure can produce very bad results for non-tubular shapes. This is so because the curve-skeleton may result in a non-continuous curve; hence the use of an active contour approach instead, so that the medial procedure described above is performed optionally as a pre-processing step.

### 2.4.5 Skeleton-Based Segmentation: a Discussion

As shown in Figure 2.37, skeleton-based segmentation methods made their appearance soon after 2000, with a slight predominance of Reeb graph-based methods. Moreover, two sorts of input data are admissible: 2-dimensional mesh and point cloud. These methods are particularly adequade for tubular (or articulated) freeform methods such as, for example, human-like shapes. They are not so suited for mechanical parts and hand-made artifacts, simply because the skeletons of those objects rarely have tubular branches. Moreover, slecketons do not capture small features as, for example, the curve that separates the hair or nose from the face of a human body mesh.

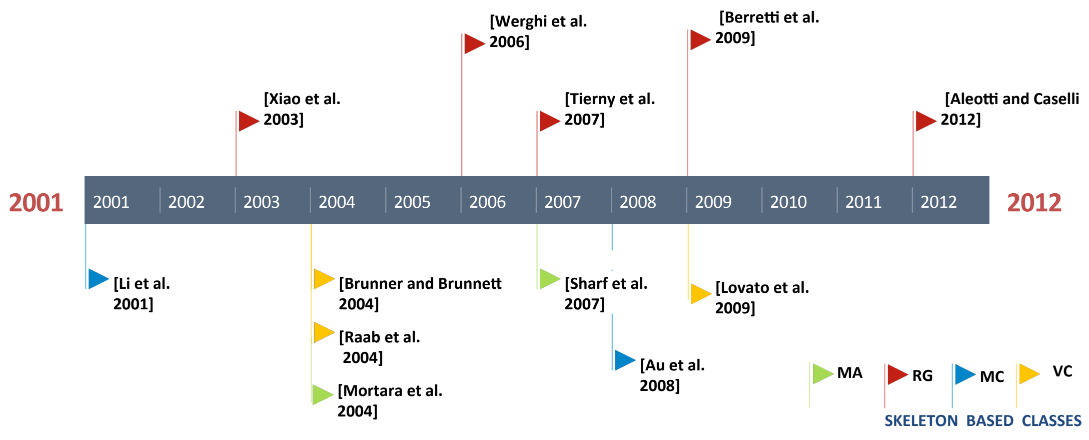


Figure 2.37: Timeline of the skeleton-based algorithms.

Table 2.3: Skeleton-based segmentation methods.

| Method | Reference | Input | | Object type | | Class | | | | Skeleton type | | | | Feature | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | M | PC | Ff | NFf | MA | RG | MC | VC | Sk | DRG | CSK | RG | Nt | OS | A |
| Li et al. | [LWTH01] | • | | • | | | | • | | • | | | | • | | • |
| Xiao et al. | [XSW03] | | • | • | | | • | | | | • | | | • | | • |
| Brunner and Brunnett | [BB04] | • | | • | | | | | • | • | | | | • | | • |
| Mortara et al. | [MPS+04] | • | | • | | • | | | | • | | | | • | | • |
| Raab et al. | [RGS04] | • | | • | | | | | • | | | • | | • | | • |
| Werghi et al. | [WXS06] | • | • | • | | | • | | | | • | | | • | | • |
| Sharf et al. | [SLSK07] | • | • | • | | • | | | | | | • | | • | | • |
| Tierny et al. | [TVD07] | • | | • | • | | • | | | | | | • | • | • | • |
| Au et al. | [ATC+08] | • | | • | | | | • | | | | • | | • | | • |
| Lovato et al. | [LCG09] | • | • | • | | | | | • | | | • | | • | | • |
| Berretti et al. | [BDBP09] | • | | • | • | | • | | | | | | • | • | | • |
| Aleotti and Caselli | [AC12] | • | | • | • | | • | | | | | | • | • | | • |

Abbreviations:
**M**: Mesh; **PC**: Point Cloud;
**Ff**: Freeform; **NFf**: Non-Freeform or CAD;
**MA**: Medial Axis; **RG**: Reeb Graph; **MC**: Mesh Contraction; **VC**: Volumetric Contraction;
**Sk**: Skeleton; **DRG**: Discrete Reeb Graph; **CSK**: Curve-Skeleton;**RG**: Reeb Graph;
**Nt**: Naturalness; **OS**: Over-Segmentation; **A**: Automatic;

The main problem of medial axis-based methods lies in the difficulty in constructing of the medial axis itself. On the other hand, the main difficulty of Reeb graph-based methods is in the choice of the height function used to build the Reeb graph. Also, Reeb graph-based methods tend to produce less natural segmentations than other methods belonging to the category of skeleton-based methods. In respect to mesh contraction-based methods, the main issue is the production of non-connected skeleton, which needs some sort of reconstruction procedure, though the latest algorithms in this category have paved the way to overcome this problem. In regard to volumetric contraction-based methods, the main issue has to do with the resolution of the 3-dimensional lattice, i.e., the size of its building blocks (e.g., voxels), which has impact on whether or not the features of an object are well determined. As expected, and as shown in Table 2.3, skeleton-based methods do not suffer from over-segmentation, except the one due to Tierny et al. [TVD07], but even in this case the over-segmentation is not much. Finally, let us mention that these methods are in general automated because the number of segments can be determined from object skeleton.

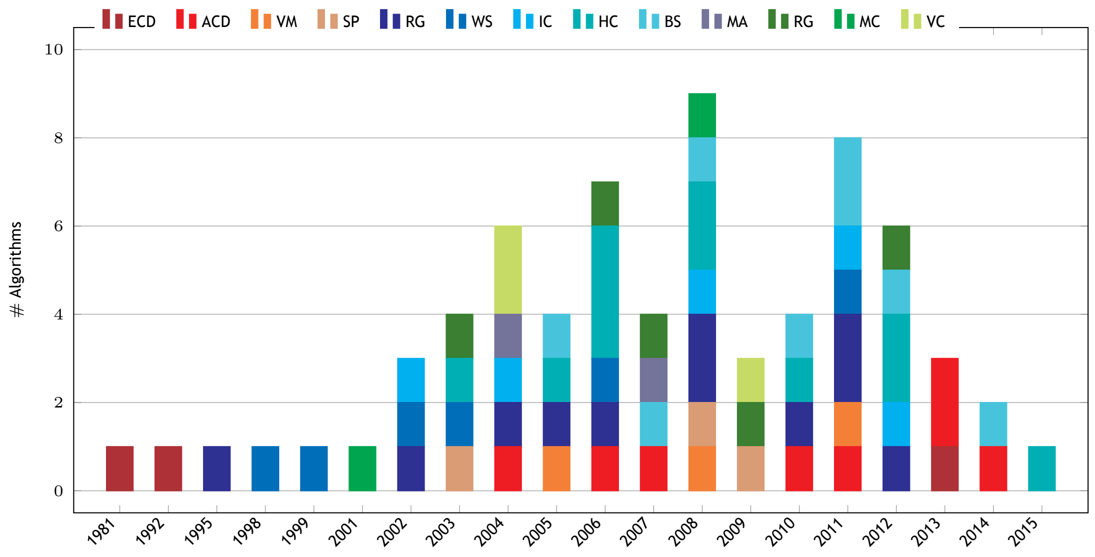


Figure 2.38: Time distribution of the meaningful segmentation algorithms, grouped by classes.

## 2.5 Evaluation, Discussion and Future trends

A brief glance at Figure 2.38 shows us that the field of meaningful mesh segmentation attained its peak in 2008 in terms of the number of algorithms, with the boundary-based and approximate convex decomposition-based techniques gaining a particular emphasis in recent years.

### 2.5.1 Quantitative Evaluation

Let us now compare some of the mesh segmentation methods described above in the context of the Princeton benchmark [CGF09]. This benchmarking dataset comprises 4,300 manually generated segmentations (created by people from around the world) for 380 surface meshes collected into 19 different object categories (Figure 2.39).

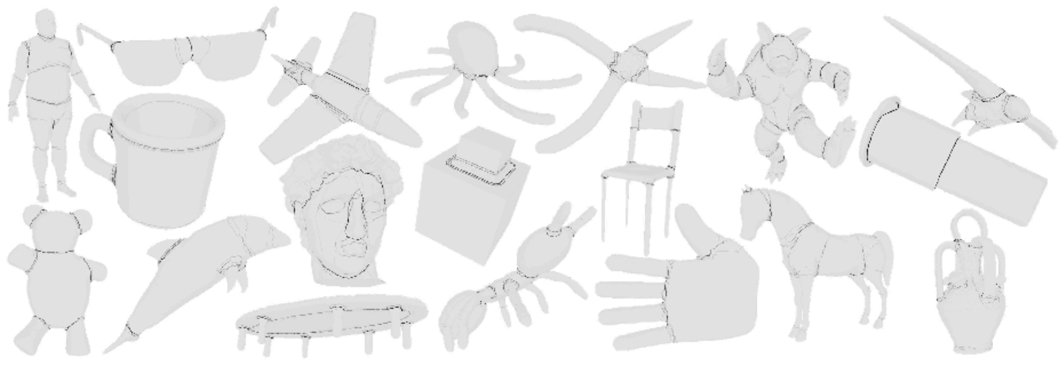


Figure 2.39: Some segment boundaries selected by different people, one example is shown for each of the 19 object categories [CGF09].

Additionally, the dataset includes a set of segmentations produced by seven algorithms published before 2009, namely: KMeans [LZHM06] (hierarchical clustering technique), random walks [LHMR08] (iterative clustering technique), fitting primitives [AFS06] (hierarchical clustering technique), randomized cuts [GF08] (boundary-based technique), core extraction [KLT05] (region growing technique), and also shape diameter function [SSCO08] (hierarchical clustering technique). Additionally, we have included in the dataset those segmentations generated by other five algorithms described in articles published after 2011, namely: isoline [AZC$^+$12] (boundary-based technique), M-S [ZZWC12] (boundary-based technique), WC-Seg [KFK$^+$14] (volume-based technique), SSFSeg [WLAT14] (boundary-based technique), and HSC [ZLG$^+$15] (hierarchical clus-

Table 2.4: Benchmark metrics [CGF09].

| Method | Class | Metrics | | | |
|---|---|---|---|---|---|
| | | RI | CD | HD | GCE |
| Benchmark[CGF09] | - | 0,101 | 0,171 | 0,098 | 0,067 |
| Rand Cuts [GF08] | BS | 0,156 | 0,275 | 0,136 | 0,122 |
| Shape Diam.[SSCO08] | HC | 0,174 | 0,274 | 0,165 | 0,129 |
| Core Extra.[KLT05] | RG | 0,208 | 0,374 | 0,169 | 0,135 |
| Rand Walks[LHMR08] | IC | 0,227 | 0,384 | 0,207 | 0,175 |
| Fit Prim. [AFS06] | HC | 0,214 | 0,348 | 0,240 | 0,217 |
| KMeans[LZHM06] | HC | 0,252 | 0,419 | 0,275 | 0,249 |
| Isoline Cut[AZC$^+$12]* | BS | 0,127 | 0,220 | 0,130 | 0,090 |
| M-S [ZZWC12]* | BS | 0,120 | - | - | 0,100 |
| WC-Seg [KFK$^+$14]* | ACD | 0,125 | 0,220 | 0,129 | 0,100 |
| SSF-Seg [WLAT14]* | BS | 0,110 | 0,200 | 0,120 | 0,090 |
| HSC[ZLG$^+$15]* | HC | 0,170 | 0,270 | 0,160 | 0,120 |

(*approximated values)

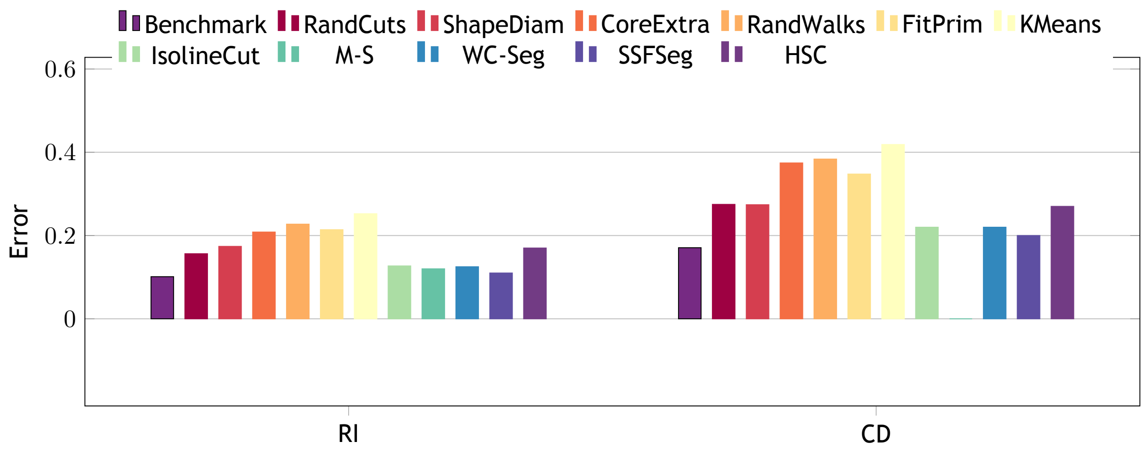


Figure 2.40: Comparison of the algorithms using the metrics RI and CD [CGF09].

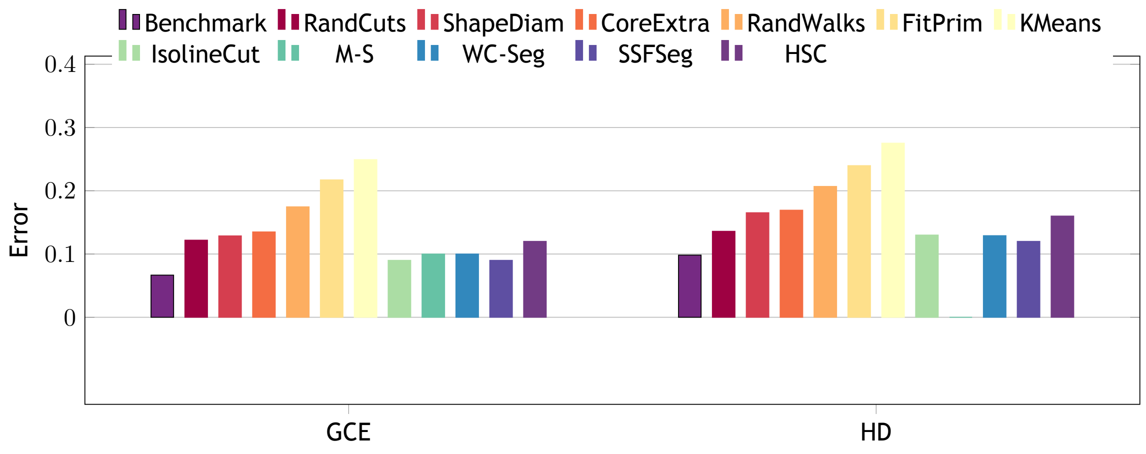


Figure 2.41: Comparison of the algorithms using the metrics GCE and HD [CGF09].

tering technique). The quantitative results of the segmentations generated from 12 algorithms, here expressed in terms of the four metrics, are listed in Table 2.4.

Such metrics are as follows:

- *Rand Index* (RI). It measures the likelihood that two triangles belong to either the same cluster (or region) or two different clusters (regions) when we are comparing two segmentations of the same mesh. In other words, it measures the similarity between two segmentations of the same mesh.

- *Cut Discrepancy* (CD). Intuitively, it is a boundary-oriented metric that measures the distances between cuts. The idea is to measure how well the region boundaries overlap with the ground truth.

- *Consistency Error* (CE). This metric measures hierarchical (or nested) similarities and differences between two distinct segmentations.

- *Hamming Distances* (HD). They measure the overall region-based differences between two segmentation results. The main advantage of these metrics is that they allow us to find correspondences between regions produced by a segmentation algorithm and those of human-generated segmentations.

In short, one of the metrics focuses on boundary errors (Cut Discrepancy), and the other three focus on region dissimilarities (Hamming Distance, Rand Index, and Consistency Error). Note that these metrics stand for errors that vary in the range $[0, 1]$, as partially shown in Figures 2.40 and 2.41.

A brief glance at the charts in Figures 2.40 and 2.41 shows us the following. First, the segmentations produced by methods described in articles published after 2011 seem to be more close to the ground-truth (or human) segmentations, i.e., they are more natural. Second, three out of five methods published after 2011 belong to the category of boundary-based methods, and seem to outperform other methods in terms of naturalness (or meaningfulness). In a way, this explains why this thesis focuses on boundary-based methods, in particular the next chapter approaches a new method belonging to this category.

### 2.5.2 Discussion and Future Trends

Looking back to the three main categories of meaningful mesh segmentation methods, we observe the following. The meaningfulness of a mesh segmentation depends on whether the decomposition of a mesh into meaningful parts satisfies Hoffman's assertion [HS97].

Volume-based methods are very time-consuming because of geometric decomposition of an object into sub-objects, which involves cutting procedures based on geometric intersections. They require the use of approximate convexity to succeed in reducing the over-segmentation phenomenon and in making segmentations more meaningful.

Surface-based methods have difficulties in dealing with over-segmentation, unless we use the merging step. Among these surface-based methods, the emerging boundary-methods seem to be very promising because they tend to rid off the over-segmentation.

Skeleton-based methods do not suffer from over-segmentation; in addition, they are inherently automatic since it is possible to determine the number of segments from the skeleton of a mesh. The downside of these method is that they are are mainly designed for articulated objects, i.e., they do not apply to objects in general (i.e., mechanical parts). Alias, mesh segmentation methods hardly are general.

Although, many methods have tried to cope with all sorts of objects, we noted that some are more suited for freeform objects, while others are more adequate for non-freeform objects, or even others fit in those specificities of articulated objects.

In the future, we envisage the following challenges in the design of new meaningful mesh segmentation algorithms:

***Naturalness***. This has to do with perceptual proximity to ground-truth segmentations.

As seen above, in recent years, we have assisted to the introduction of new algorithms that produce more and more natural segmentations. In particular, the have assisted to emergence of the boundary-based methods that well comply with the requirement of naturalness.

***Generality***. There is a long way to go in order to come up with general mesh segmentation methods, i.e., methods capable of dealing with objects of distinct categories in a similar degrees of naturalness given by metrics as those aforementioned.

***Automation***. The *a priori* computation of segments of a mesh is an important research topic for future. We have seen some hints for this computation as, for example, critical points, concave paths on the mesh, skeletons and the like. But, as we will see in the next chapter, it is also possible to bring techniques from image segmentation into mesh segmentation in order to find new ways to solve this issue.

### 2.5.3 Other Literature Reviews

In the last few years, two relevant surveys have been appeared in the literature. First, Shamir [Sha08] introduced a classification of mesh segmentation algorithms based on different segmentation goals, optimization criteria and features, as well as on their algorithmic techniques or methodological frameworks. Second, and following essentially the same line of thinking as Shamir [Sha08], Theologou el al. [TPT15] presented a survey on mesh segmentation algorithms (include chart segmentation, meaningful segmentation and co-segmentation) grouping them according to their methodological frameworks (e.g., clustering, region growing, surface fitting, and so forth)

On the contrary, we use the dimension of the building blocks of the segmentation to introduce a new taxonomy for meaningful mesh segmentation. Therefore, in volume-base methods, the 3D object volume is divided into 3D sub-volumes, even when volumes are represented by 2D shells (i.e., boundary representtaion); in surface-based methods, a 2D mesh in 3-dimensional space is divided into 2D sub-meshes with boundary; in skeleton-based methods, we first find a 1D skeleton which is divided into 1D skeletal branches.

## 2.6 Concluding Remarks

In this chapter, we have carried out a general literature review of the meaningful mesh segmentation algorithms. As aforementioned, in the end of the last decade, we have assisted to a seemingly decline of this knowledge field in terms of production of new solutions for the problem of mesh segmentation. However, this decline has given place to the rise of another category of algorithms, the now well-known co-segmentation algorithms that take advantage of machine learning techniques. We have not approached the chart segmentation methods either.

## Related Publications

The work described in this chapter originated a survey on meaningful segmentation methods, which is to be submitted for publication as indicated below:

> *Rui S. V. Rodrigues, José F. M. Morgado, and Abel J. P. Gomes. 2016. Mesh Segmentation: A Literature Review. ACM Computing Surveys (submitted), 2016.*

# Chapter 3

# A Contour-Based Segmentation for Triangle Meshes in 3D Space

This chapter introduces the first contour-based mesh segmentation algorithm that we may find in the literature, which is inspired in the edge-based segmentation techniques used in image analysis, as opposite to region-based segmentation techniques. Its leading idea is to firstly find the contour of each region, and then to identify and collect all of its inner triangles. The encountered mesh regions correspond to ups and downs, which do not need to be strictly convex nor strictly concave, respectively. These regions, called relaxedly convex regions (or saliences) and relaxedly concave regions (or recesses), produce segmentations that are less-sensitive to noise and, at the same time, are more intuitive from the human point of view; hence it is called human perception-oriented (HPO) segmentation. Besides, and unlike the current state-of-the-art in mesh segmentation, the existence of these relaxed regions makes the algorithm suited to both nonfreeform and freeform objects.

## 3.1 Introduction

Mesh segmentation is a fundamental procedure in areas as diverse as geometric modeling, computer-aided design, computer graphics, and so forth [Sha08]. This procedure consists in partitioning a mesh into a number of sub-meshes in conformity with some convexity criterion. Intuitively, this is equivalent to detect and delimit the ups (saliences) and downs (recesses) of the mesh, as illustrated in Figure 3.1. But, as argued by Attene et al. [AKM+06], mesh segmentation can be driven by either geometric criteria or semantic criteria or both.

In geometry-based segmentation techniques, the mesh is divided into a number of sub-meshes or regions that satisfy some geometric property (e.g., curvature, distance to a fitting plane, etc.). On the other hand, in semantics-based segmentation techniques, the division of a mesh into sub-meshes takes place when each sub-mesh delimits a perceptually meaningful region (e.g., an arm of the human body). In this respect, Biederman [Bie87] states that the people perceive objects as collections of parts, while Hoffman [HS97] refers that the human vision defines boundaries of parts along the negative minima of principal curvatures. Note that, in a way, Hoffman's assertion implies that the meaningful parts are essentially convex.

Hoffman's assertion has inspired the development of our algorithm, so we follow the leading idea of how human vision perceives the boundaries of parts. Let us say that

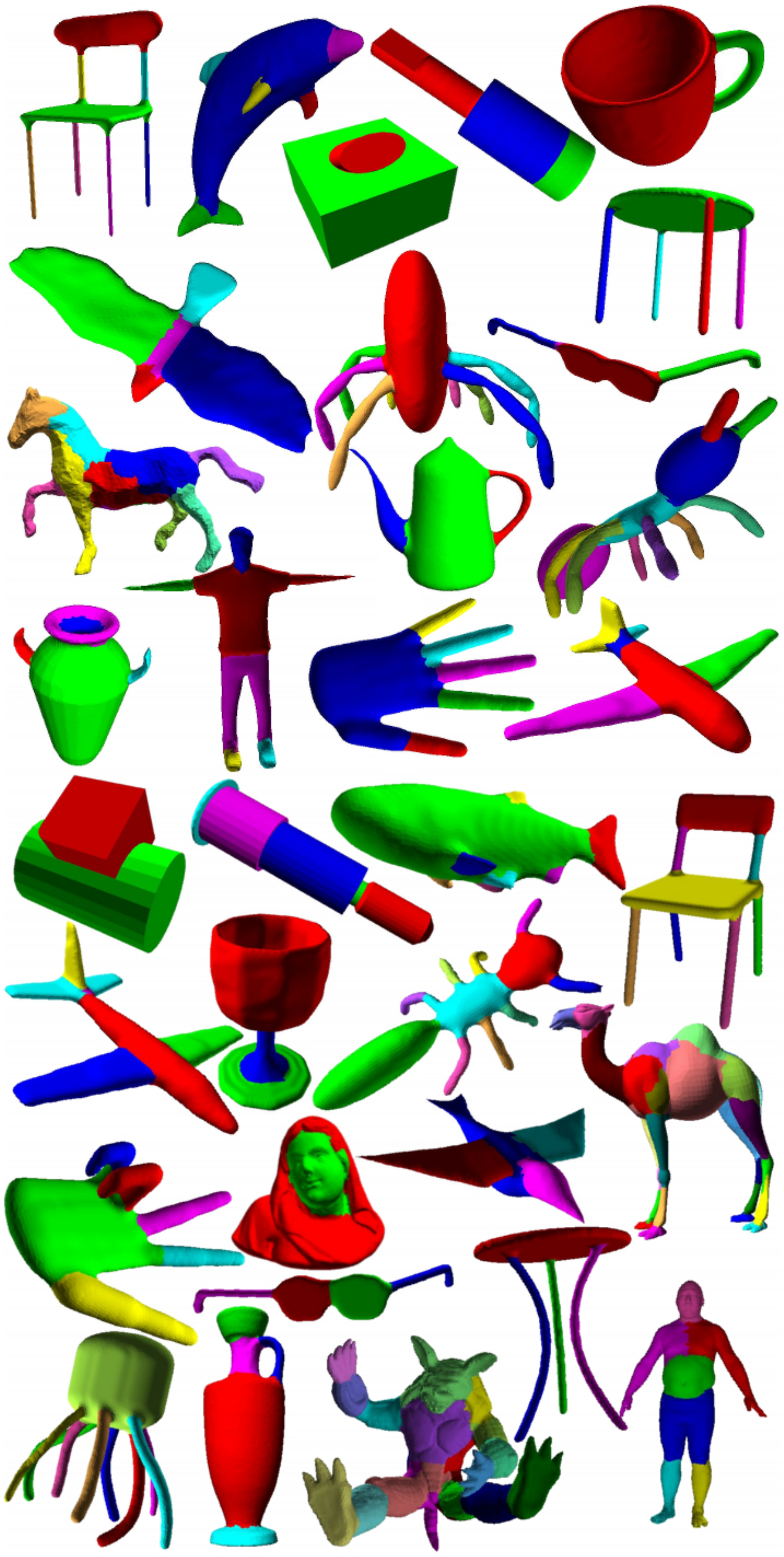


Figure 3.1: Segmented meshes (or models) using the human perception-oriented (HPO) segmentation.

our algorithm has been also inspired in the well-known family of edge-based segmentation algorithms found in image analysis and algorithms. With these edge-based algorithms, we first find the boundaries of each region, filling it afterwards; hence, the contour-based segmentation (region contour or boundary first, then its interior). On the contrary, the region-based algorithms work in the other way round: first the region is formed incrementally, being its frontier defined by some stopping condition.

As far as we know, we are here proposing the first contour-based segmentation in the domain of mesh segmentation in 3D. For this purpose, the point membership test (PMT) is here used as a convexity classifier, i.e., we use PMT to classify edges as convex, concave or flat. PMT is a particular case of the SMC (set membership classification) test, which is very popular in CSG (constructive solid geometry) modeling, and has been around for the last three to four decades [Til80]. Nevertheless, recesses and saliences on the mesh are not classified in a so strict manner in terms of convexity. We introduce the notion of *relaxed convexity* to classify those shape features of the mesh. A salience is a relaxedly convex region, while a recess is a relaxedly concave region. A relaxedly convex region is not strictly convex, i.e., it admits small concavities. On the other hand, a relaxedly concave region is not strictly concave, i.e., it admits small convexities. This shape relaxation allows us to minimize the effects of noise related to over-segmentation, and makes it suited to the segmentation of freeform objects such as, for example, the ant shown in Figure 3.1.

The remainder of the chapter is organized as follows. Section 3.2 briefly reviews the related work existing in the literature. Section 3.3 details the background that is on the basis of our algorithm, including PMT and the concept of relaxed convexity. Section 3.4 outlines our contour-based segmentation algorithm, called HPO segmentation. Section 3.5 details the region filling step of HPO segmentation algorithm. Section 3.6 introduces the mesh smoothing step (a Laplacian filter, in particular) for noisy meshes, which uses the cumulative area histogram analysis to identify whether a mesh is noisy or not. Section 3.7 describes the region merging step of HPO segmentation algorithm. Section 3.8 presents the most relevant experimental results produced by the HPO segmentation algorithm, in particular the benchmarking results produced by Princeton's benchmark software. Finally, Section 3.9 concludes the chapter.

## 3.2 Related Work

Intuitively, mesh segmentation consists in dividing a mesh into meaningful sub-meshes (or regions). With reference to the structural dimension of mesh regions, segmentation algorithms can be categorized as follows: *volume-based*, *surface-based*, and *skeleton-based*. These three segmentation categories are all based on geometry, including concepts of the convexity theory, topology, homology, etc. Independently of the nature of the segmentation technique, most algorithms make usage of some geometric or topological criteria that guide the segmentation of a given mesh [Sha08]. Examples of geometric criteria are the curvature [KK10], geodesic distances [ZMT05], dihedral angles [ZTS02], shape diameter function [SSCO08], planarity [KT96], symmetry [PSG+06], convexity [KJS06] [LLL10], concavity [AZC+12] [GALL13], volume [HC12], etc. Topological criteria include Reeb graphs [TVD07] and spectral analysis [ZvKD10].

*Volume-based* segmentation algorithms can be considered as the former category of segmentation algorithms. In this case, the segments are volumes. This family of algorithms follows the principle of decomposing a 3D volumetric object into a set of convex sub-objects (or convex volumes). The basic problem of convex decomposition of polyhedra was firstly addressed by Chazelle [Cha81] [BD92] [CP94], but such decompositions usually are costly in terms of time performance and memory space. More amenable decompositions, called approximate convex decompositions (ACD), have been proposed in the literature [LA04] [KJS06] [GALL13]. This sort of decomposition is based on a measure of concavity; for example, the ACD proposed in [LA06] is guided by the volume ratio between the actual object and its convex hull, Ghosh et al. [GALL13] propose a new strategy that improve the results of the ACD, while the one due to Kraevoy et al. [KJS06] is generated by measuring the average distance from all object's triangles to the object's convex hull.

In *surface-based* segmentation algorithms, the segments are regions of a 2D triangle mesh. Each region consists of a number of connected triangles that have similar geometric properties (e.g., convexity, curvature, etc.). In the literature, we find the following major sub-categories surface-based algorithms: region growing [CDST95] [KT96], watersheds [MW99] [ZTS02], hierarchical clustering [GWH01], iterative clustering [SWG+03], [KT03], spectral clustering [LZ07], [RBG+09], and fuzzy clustering [KT03].

In *skeleton-based* segmentation (also known as *skeletonization*) algorithms, the segments are line segments. The input is either a 3D volumetric mesh or a 2D surface mesh, but the output is a 1D skeleton that represents the structural shape of the mesh. A skeleton provides us with the one-dimensional topological shape of a given higher-dimensional object. Generating such a skeleton is a process known as skeleton extraction or skeletonization [CGC+02] [KT03]. Examples of automatic skeletonization techniques are found in the literature, including the Medial Axis Transform (MAT) [Blu67], Shock graph [SSDZ98] (i.e., another MAT representation), and Reeb graph extracted from various Morse functions [SKK01].

Note that our algorithm falls into the category of surface-based algorithms, but it does not fit any of its sub-categories. Making a parallel with the two more important families of segmentation algorithms we may find in image analysis and processing [GW08], a region-based segmentation algorithm starts somewhere in the interior of each region and stops on its frontier, while a contour-based segmentation algorithm starts on the frontier of each region and stops in its interior. This means that, we do not need to calculate neither maxima nor minima to start with the segmentation, nor to use other suited, sophisticated mechanisms to determine the curvature over the mesh as those we find commonly nowadays.

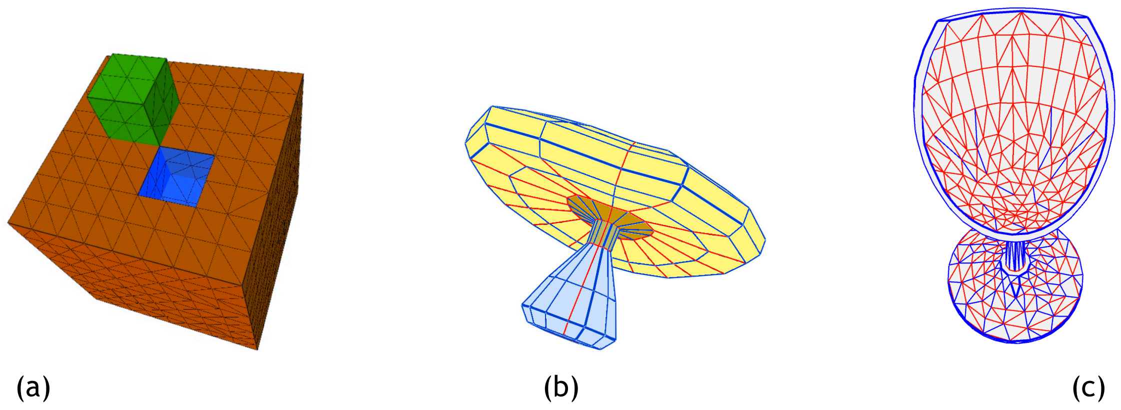


Figure 3.2: Three different meshes: (a) a mesh with two saliences (a brown salience and a green salience) and a blue recess; (b) a mushroom with four segments or regions (convex edges in blue and concave edges in red); (c) a calyx with a cut (convex edges in blue and concave edges in red).

## 3.3 Theoretical Background

According to Hoffman's assertion mentioned in Section 3.1, the human vision delineates the boundaries of object's regions along the negative minima of principal curvatures, what implies that meaningful regions of an object are convex. This agrees with the fact that there is only lack of matter (or existence of recesses) when the matter exists (or existence of saliences). But, when we talk about the convex regions, in fact, in most cases, we are talking about regions that we call *relaxedly convex* regions. This is illustrated in Figure 3.2(b), where the four meaningful parts of the mushroom are not strictly convex, i.e., we tend to observe the convexity in large. For example, the mushroom cap (i.e., the top region) is a relaxedly convex region since it mostly consists of convex edges in blue, although it also possesses concave edges in red.

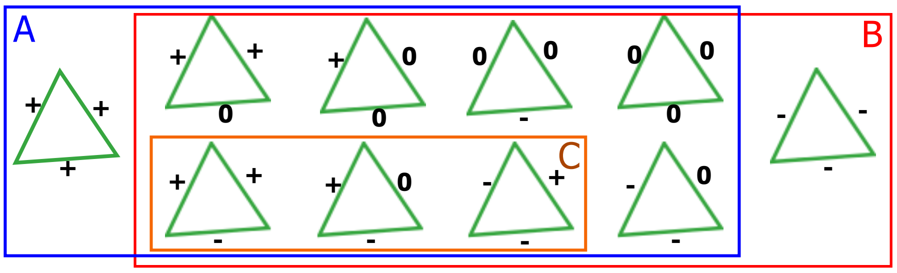


Figure 3.3: Edge types of a triangle: (+) for convex edges; (0) for flat edges; and (-) for concave edges. $A$ stands for the set of triangles used for build relaxedly convex regions; $B$ stands for the set of triangles used for build relaxedly concave regions; and $C$ stands for the set of boundary triangles.

### 3.3.1 Edge Classification

Evaluating the convexity of an edge bounding a mesh triangle can be reduced to the problem of classifying a point in relation to a plane. Let us then use the vectorial equation of the plane $\alpha$ given by the dot product $\overrightarrow{N} \cdot \overrightarrow{QP} = 0$, where $\overrightarrow{N}$ is the normal vector of the triangle at one of its points $Q$ and $\overrightarrow{QP}$ stands for a vector in $\alpha$ defined by $Q$ and any other point $P$ belonging to $\alpha$. Assuming that $N = (a, b, c)$, $Q = (x_0, y_0, z_0)$ and $P = (x, y, z)$, we easily obtain the Cartesian equation of the plane as $\alpha(x, y, z) : ax + by + cz + d = 0$, where $d = (-ax_0 - by_0 - cz_0)$.

It is clear that the plane $\overrightarrow{N} \cdot \overrightarrow{QP} = 0$ divides $\mathbb{R}^3$ in two half-spaces, i.e., the positive half-space $\overrightarrow{N} \cdot \overrightarrow{QP} > 0$ and the negative half-space $\overrightarrow{N} \cdot \overrightarrow{QP} < 0$. The normal $\overrightarrow{N}$ points to the positive half-space (i.e., outside), so that we can easily test whether an edge is convex or not using the plane of the first of its incident triangles and the opposite vertex (i.e., the third vertex) of the second triangle.

Let $\overrightarrow{N_1}$ and $\overrightarrow{N_2}$ be the normal vectors of the first and second triangles incident on a given edge, respectively. Also, let $P_1$ and $P_2$ be the opposite vertices of the first and second triangles. Thus, after calculating $\Delta = \overrightarrow{N_1} \cdot \overrightarrow{P_1P_2}$, we have:

- If $\Delta = 0$, then the triangles are coplanar, and the edge is said to be *flat*;
- If $\Delta < 0$, the second triangle lies in the negative side of the plane $\alpha$, that is, the edge shared by those two triangles is said to be *convex*;
- If $\Delta > 0$, the second triangle lies in the positive side of $\alpha$, that is, the edge shared by those two triangles is said to be *concave*.

Note that this criterion, here called PMT, can be used to know whether or not an edge is convex without computing the dihedral angle between its incident triangles. Besides, the PMT does not suffer from the ambiguity of the dihedral angle in evaluating the convexity of edges. During testing, we observed that PMT criterion is more than fifty percent faster than the dihedral angle criterion in the evaluation of the convexity of edges.

### 3.3.2 Triangle Classification

The PMT is a strict convexity criterion for edges. An edge shared by two triangles is flat if $\Delta = 0$, convex if $\Delta < 0$, and concave if $\Delta > 0$. Considering that we have three different types of edges and also that a triangle has tree bounding edges, we end up having $3^3 = 27$ types of triangles classified in terms of their edges. Nevertheless, after eliminating the redundant types of triangles, we end with 10 different types of triangles, which are shown in Figure 3.3.

**Définition 1.** *A triangle falls into one of the following categories:*

1. *A triangle is of the **type+3 convex** (or simply convex) if its edges are all convex (e.g., the triangle on the left hand side in Figure 3.3);*

2. *A triangle is of the* ***type+2 convex*** *if it has two convex edges and one flat edge;*
3. *A triangle is of the* ***type+1 convex*** *if it has one convex edge and two flat edges;*
4. *A triangle is* ***flat*** *if its edges are all flat;*
5. *A triangle is of the* ***type-3 concave*** *(or simply concave) if its edges are all concave (e.g., the triangle on the right hand side in Figure 3.3);*
6. *A triangle is of the* ***type-2 concave*** *if it has two concave edges and one flat edge;*
7. *A triangle is of the* ***type-1 concave*** *if it has one concave edge and two flat edges;*
8. *A triangle is of the* ***type+2-1 undefined*** *if it has two convex edges and one concave edge;*
9. *A triangle is of the* ***type+1-2 undefined*** *if it has one convex edge and two concave edges;*
10. *A triangle is of the* ***type+1-1 undefined*** *if it has one convex edge, one concave edge, and one flat edge.*

In short, we have tree types of convex triangles, one type of flat triangles, three types of concave triangles, and three types of undefined triangles in terms of convexity.

### 3.3.3 Region Classification

Let us now to come up with definitions about regions, which are sustained on the definitions introduced in previous sections.

**Définition 2.** *A* ***region*** *of a triangle mesh is a connected set of triangles (or sub-mesh) that locally has the topological type of a 2-dimensional disc.*

**Définition 3.** *A region is said to be* ***convex*** *if the following conditions are satisfied:*

(1) (Interior Condition). *Its triangles belong to the boundary of its convex hull;*

(2) (Boundary Condition). *It is delimited by at least one cycle of concave edges.*

A consequence of the first condition is that the inner triangles of a convex region are convex (any type) or flat. It also follows that triangles with concave edges are not admissible for inner triangles of a convex region. Additionally, those two conditions imply that each outer triangle of a convex region has a single concave edge, that is, each outer triangle belongs to one the following types: type-1 concave, type+2-1 undefined, or type+1-1 undefined. For example, the outer triangles of the salience (in green) shown in Figure 3.2(a) fall into two types: type-1 concave and type+1-1 undefined. Note that a convex region may be bounded by two cycles of concave edges, as it is the case of the smallest region (in green) of the five-region shaft depicted in Figure 3.1.

Analogously, a ***concave*** region also satisfies the interior condition (1) above. By contrast, the boundary of concave region consists of at least one cycle of convex edges. For example, the depression (in blue) shown in Figure 3.2(a) is a concave region that is bounded by a cycle of convex edges. Thus, the inner triangles of a concave region are concave (any type) or flat, while those of a convex region are convex (any type) or flat.

In respect to relaxedly convex regions, we can say the following:

**Définition 4.** *A region is said to be **relaxedly convex** if the following condition is satisfied:*

(1) (Relaxed Interior Condition). *Its triangles do not necessarily belong to the boundary of its convex hull;*

(2) (Boundary Condition). *It is delimited by at least one cycle of concave edges.*

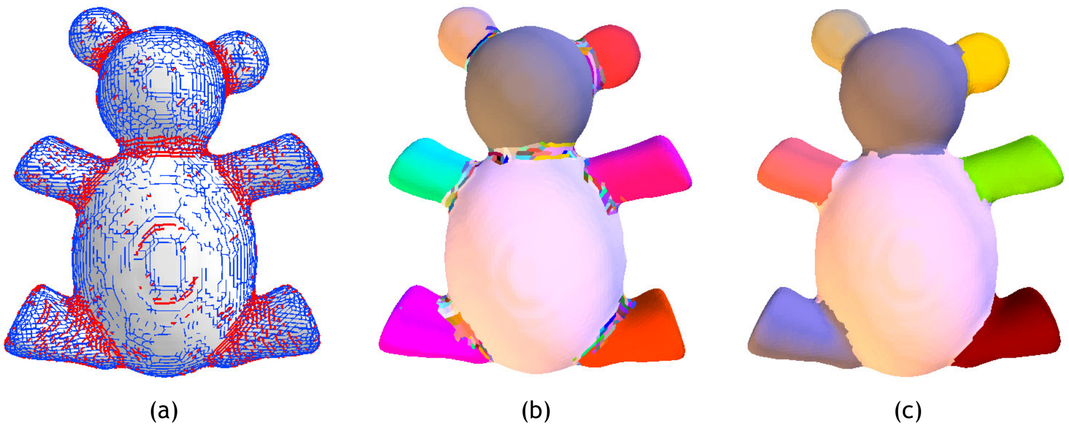


Figure 3.4: Algorithm steps for a teddy mesh: (a) after edge classification (convex edges in blue and concave edges in red) and triangle classification; (b) after region filling; (c) after region merging.

That is, a relaxedly convex region does not satisfy the interior condition of a convex region; however, convex regions and relaxedly convex region share the same boundary condition. This means that we may have concave edges in the interior of a relaxedly convex region since that they do not form a loop. For example, the mushroom cap shown in Figure 3.2(b) is a relaxedly convex region that has concave edges (in red) in its interior; in addition, the region adjacent to the mushroom cap shown in Figure 3.2(b) is a relaxedly convex region with two bounding cycles of concave edges.

Similar to relaxedly convex regions, a ***relaxedly concave*** region also satisfies the relaxed interior condition. That is, we may find convex edges in the interior of a relaxedly concave region; however, they cannot form a cycle. On the other hand, concave regions and relaxedly concave regions share the same boundary condition, that is, they are delimited by one or more cycles of convex edges. For example, the inside of the calyx shown in Figure 3.2(c) is a relaxedly concave region that has convex edges (in blue) in its interior.

It is worthy noting that a convex region is also relaxedly convex, but not vice-versa; similarly, a concave region is also relaxedly concave, but not vice-versa. Relaxed convexity is particularly important in the segmentation of freeform objects; for example, the legs of the octopus depicted in Figure 3.1 are examples of relaxedly convex regions. But, it is also suitable for feature recognition of artifacts as those produced by geometric kernels and CAD systems (e.g., the two mechanical shafts in the Figure 3.1). In short, unlike other segmentation techniques that divide a mesh into convex, concave and saddle regions, ours divides a mesh into relaxedly convex regions (saliences) and relaxedly concave regions (recesses), preventing so the over-segmentation of the mesh as much a possible.

## 3.4 Contour-Based Segmentation Algorithm: Overview

Considering the previous theoretical elements on convexity, the steps of our algorithm are as follows:

1. *Edge Classification*. Classify each edge of the mesh as either convex edge, concave edge or flat edge. This classification is performed using the point membership test (PMT) instead the dihedral angle. See Section 3.3.1 for further details.
2. *Triangle Classification*. This procedure builds on the edge classification above, as formalized by Definition 1 in Section 3.3.2.
3. *Region Filling*. Form mesh regions from boundary triangles, that is, triangles having at least one edge in the boundary of each region. However, the algorithm is flexible enough to start with any triangle to form a region. See Section 3.5 for more details.
4. *Mesh Smoothing*. This step starts by checking whether or not a given mesh is noisy (e.g., Armadillo's mesh shown in Figure 3.8(a)) or not. Any noisy mesh is subject to a smoothing procedure, after which such mesh is re-segmented repeatedly while the noise persists above a given threshold. Non-noisy meshes are not subject to any smoothing procedure, i.e., the mesh is immediately dispatched to the region merging step.
5. *Region Merging*. Merge adjacent regions if appropriate. See Section 3.7 for further details.

The main steps of the algorithm are illustrated in Figure 3.4. In Figure 3.4(a), we have the edge classification (convex edges in blue, concave edges in red, and flat edges in grey), while all the triangles are drawn in grey. Figure 3.4(b) illustrates the region filling step. The inwards expansion (i.e., from the boundary to interior) of each region stops when there is not more boundary and interior triangles to fill in. Finally, Figure 3.4(c) shows us the teddy after the region merging step.

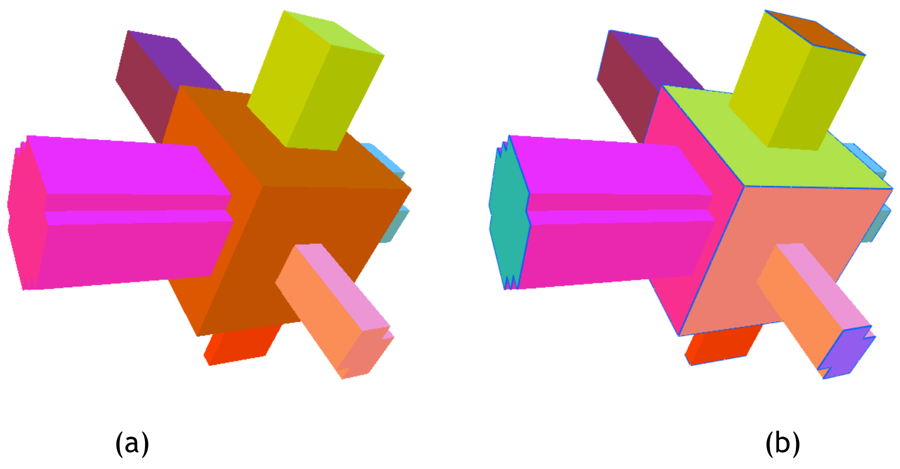


Figure 3.5: HPO segmentation: (a) first convex regions, then concave regions; (b) first concave regions, then convex regions.

Our algorithm produces a kind of human-meaningful segmentation (i.e., HPO segmentation) in a way that saliences (relaxedly convex regions) are formed before recesses (relaxedly concave regions). As illustrated in Figure 3.5(a), the saliences must be found before recesses because the resulting mesh segmentation approximately agrees with the way humans perceive the shape of 3D geometric objects. In fact, the ground-truth segmentations produced by the Princeton benchmark [CGF09] show us that humans essentially have an additive shape perception of 3D objects in the sense that each object is mainly seen as a union of saliences. To reinforce this idea, let us consider that the priority of segmentation goes to recesses at detriment of saliences. As shown in Figure 3.5(b), the resulting segmentation endows a subtractive perception of shape to 3D objects, which is not so natural from the human point of view.

## 3.5 Region Filling

Let us assume that all the mesh edges have been labeled in terms of their convexity: either convex, or concave, or flat. Also, the mesh triangles have been labelled in conformity with Definition 1. Then, we can proceed to filling in relaxedly convex regions in first place, and relaxedly concave regions afterwards.

### 3.5.1 Filling Relaxedly Convex Regions

Filling in a relaxedly convex region starts with one of its boundary triangles, i.e., with one undefined triangle (cf. undefined triangles of the set C in Figure 3.3). Recall that a boundary triangle has at least a convex edge and a concave edge. The filling procedure follows the *door-in-door-out* principle [AG03]. An edge that bounds a triangle acts as door to get in or get out such a triangle, or still to stop the local expansion of a region.

Given a relaxedly convex region, we consider that convex edges (and also flat edges) in the frontier of a triangle are open doors to neighbor triangles, so that such a region spreads inwards. In other words, if a door-in edge of a triangle is convex, its door-out edges must be convex or flat; in this case, a concave edge is a boundary edge that blocks the expansion of such a relaxedly convex region.

The procedure to form relaxedly convex regions (or saliences) of a mesh is described in Algorithm 1. As noted above, filling in a relaxedly convex region starts with a boundary triangle (cf. lines 4 to 10 in Algorithm 1). For this purpose, and according to the door-in-door-out principle, one examines whether or not the neighbor triangles may be added to the relaxedly convex region (cf. line 7), being then the growing process iterated on each previously added triangle. Recall that the filling of a relaxedly convex region is barred by concave edges (i.e., its frontier), as expressed by the condition of the while loop in line 7.

It is worthy noting that Algorithm 1 applies to all boundary triangles, i.e., no boundary triangle is left to the posterior formation of relaxedly concave regions. This means that Algorithm 1 may produce small relaxedly convex regions consisting of two triangles; for example, the isolated convex edge (in blue) in the interior of the calyx in Figure 3.2 may originate a relaxedly convex region consisting of its two incident triangles. As described in Section 3.7, these small regions will be later absorbed by larger regions in their vicinity.

**Algorithm 1** Filling in Relaxedly Convex Regions (RCR)

```
Input: $F$: array of boundary triangles
Input: $R$: array of filled RCRs
1: $R \leftarrow \varnothing$
2: $n \leftarrow$ size of $F$
3: for $i \leftarrow 0$ to $n-1$ do
4:   if $F[i] \notin$ any mesh region then
5:     Generate new RCR $r$
6:     $r \leftarrow r \cup F[i]$
7:     while $\exists$ a convex or flat edge $e$ bounding $r$ do
8:       $f \leftarrow$ triangle adjacent to $e$
9:       $r \leftarrow r \cup f$
10:    end while
11:    $R \leftarrow R \cup r$
12:  end if
13: end for
```

### 3.5.2 Filling Relaxedly Concave Regions

Likewise, the procedure to form relaxedly concave regions (or recesses) is done by exchanging the roles of convex and concave edges in Algorithm 1. Besides, the algorithm to filling in relaxedly concave regions does not iterate on the array of boundary triangles, but on the array of concave triangles (i.e., type-$k$ concave triangles, with $k = 1, 2, 3$).

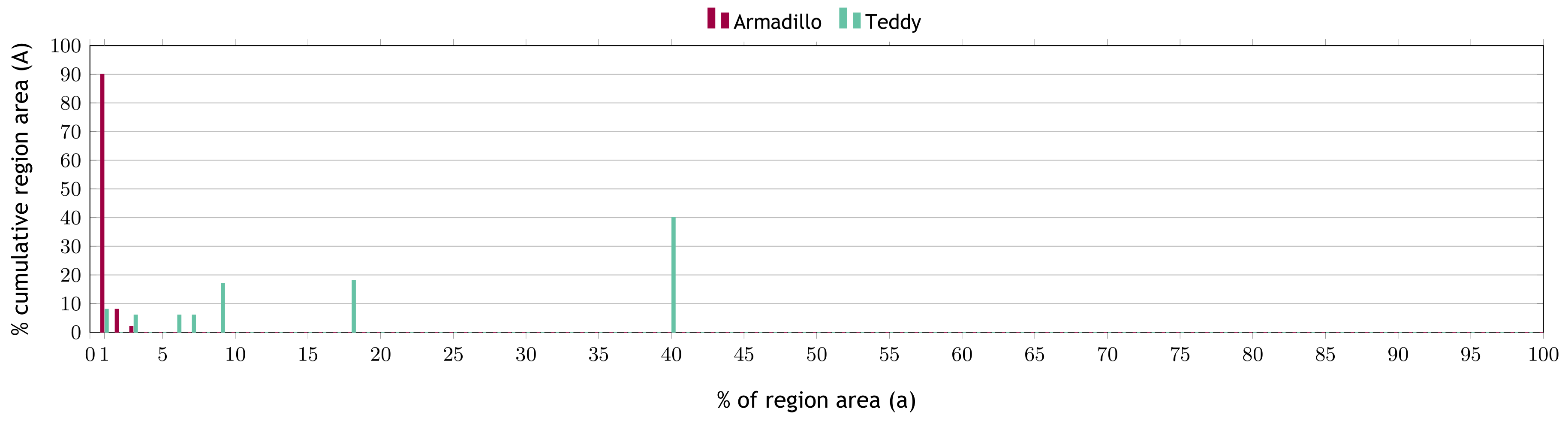


Figure 3.6: Cumulative area histogram of Armadillo and Teddy meshes (models 161 and 281 models of Princeton Benchmark).

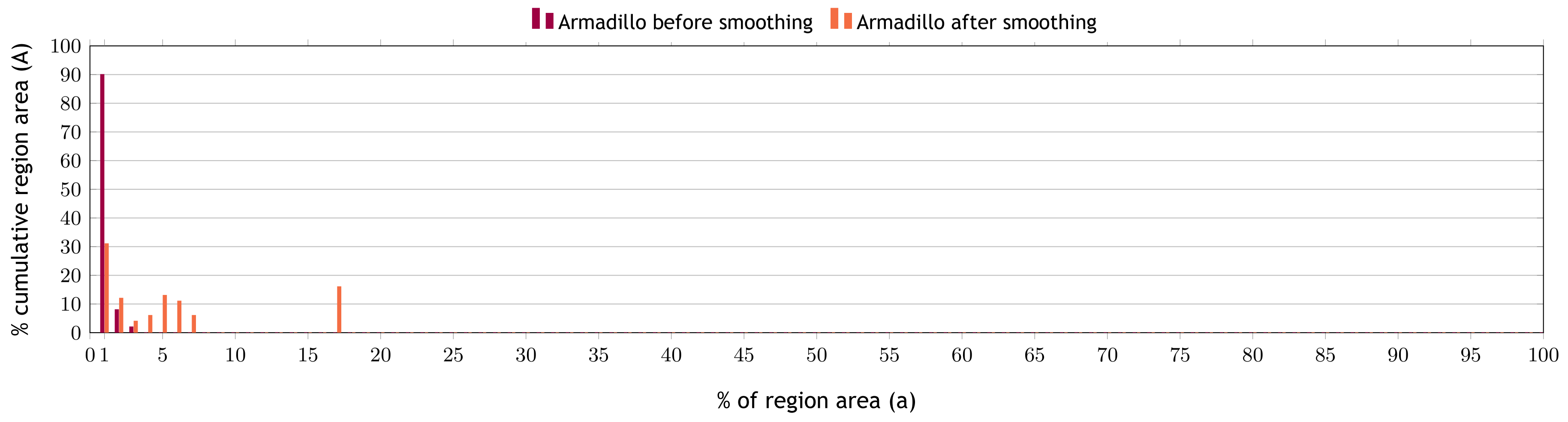


Figure 3.7: Cumulative area histogram of Armadillo before and after smoothing (model 281 of Princeton Benchmark).

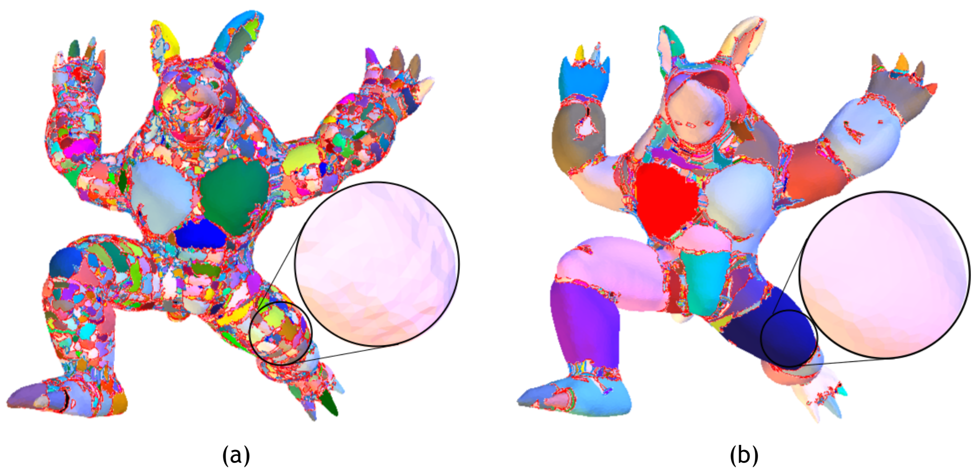


Figure 3.8: Armadillo's mesh after the region filling step (model 281 of Princeton benchmark): (a) original mesh; (b) smoothed mesh.

It is clear that if the priority of the segmentation had been given to recesses at detriment of saliences, the algorithm would iterate on the array of boundary triangles, while saliences would be formed from the array of convex triangles (i.e., type-$k$ convex triangles, with $k = 1, 2, 3$). But, as shown in Figure 3.5(b), the resulting segmentation in this case would not be so close to the human perception about shape of 3D objects.

Remarkably, the region filling step suffices to correctly segment meshes of *nonfreeform objects* (e.g., mechanical parts). Even some freeform objets do not need the supplementary step of region absorbing to attain their final segmentations, as it is the case of the mushroom shown in Figure 3.2. This makes our algorithm particularly adequate for feature recognition of artifacts as those produced by geometric kernels and CAD systems (e.g., the two mechanical shafts in Figure 3.1).

## 3.6 Mesh Smoothing

It is widely known that noise may undermine the segmentation of a given mesh. We follow the notion of noise in image processing and analysis [GW08], so that high frequencies of a signal (or image) correspond to rapid oscillations of a mesh. Intuitively, this means that a noisy mesh possesses a large number of small segments, i.e., it is over-segmented.

In order to cope with noisy meshes, we need to check beforehand whether a given segmented mesh is noisy or not; for this purpose, we use histogram analysis. Identified a noisy mesh, we apply a Laplacian filter to reduce its noise.

### 3.6.1 Area Histogram Analysis

By analyzing the cumulative relative area histogram of a given segmented mesh, we are able to decide whether such a mesh is noisy or not. For example, Figure 3.6 shows the cumulative histogram of relative region areas of two meshes, Teddy's mesh (Figure 3.4) and Armadilo's mesh (Figure 3.8), where $a$ stands for the region area percentage in relation to the total mesh area, also called *relative region area*, while $A$ denotes the cumulative region area percentage in relation to the total mesh area, here called *cumulative relative region area*. Each histogram bin $[a, a+1[$, where $0 \leq a \leq 99$, accumulates the relative areas of all regions such that the relative area of each region is greater or equal to $a$ and less than $a+1$.

Intuitively, we see that Teddy's mesh is not much noisy because the cumulative relative area of its smaller regions (i.e., regions with less than 1 percent of relative area) is small (i.e., less than 10 percent); also, we have 1 region (the body trunk) with $a = A = 40\%$, 1 region (the head) with $a = A = 18\%$, 2 regions (the legs) with $a = 9\%$ and $A = 18$, and so on. On the contrary, we observe that Armadillo's mesh is noisy because $90\%$ of the total area of the mesh is made up of the smaller regions, i.e., regions with less $1\%$ of relative area. Note that these smaller regions add up to the first bin (on the left hand side) of the histogram, here called *noise bin*.

More specifically, we assume that a mesh is noisy if the percentage of the cumulative region area ($A$) of the noise bin $[0, 1[$ is at least twice greater than the one of the second highest-ranked bin. This *noise-based criterion* is empirical and was obtained after an exhaustive number of experiments.

### 3.6.2 Laplacian Smoothing

We use Laplacian smoothing to reduce the noise of a given noisy mesh. The Laplacian operator replaces the position of each vertex by the arithmetic mean of the positions of its adjacent vertices [Her76] [ABE99] [VMM99]. Therefore, the Laplacian operator is a local shape operator. It works as a low-pass filter, i.e., it preserves the large mesh oscillations (corresponding to low frequencies of a signal) and throws away the small mesh oscillations (corresponding to high frequencies of a signal).

The Laplacian operator is applied to the mesh as many times as necessary, i.e., while the noise-based criterion is satisfied. For example, the Armadillo's histogram (in orange) in Figure 3.7 was obtained after four iterations of the Laplacian operator. It is worth noting that after applying the Laplacian operator to the mesh, the flow control of the segmentation algorithm goes back to its first step, in order to proceed to the re-segmentation of the mesh. For example, the Armadillo's segmentation shown in Figure 3.8(b) is the result of applying the Laplacian smoothing twice. Once the noise-based criterion is no longer satisfied, the mesh vertices recover their original positions before proceeding to the region merging step. It should be also noted that the smoothing step described in the present section does not apply to the majority of the meshes; it only applies to noisy meshes.

## 3.7 Region Merging

The region merging step is essentially necessary for freeform objects (e.g., human beings and other animals). These freeform objects are the most challenging objects in the segmentation process of our algorithm because they feature many variations in curvature, what provokes the *over-segmentation* of the mesh. In large extent, the ups and downs of the freeform objects led us to come up to the notion of relaxed geometry, which is at the core of our algorithm. Thus, the region merging aims at further reducing the over-segmentation of the mesh.

In the region merging step, we decided to use the algorithm proposed by Chen and Georganas [CG06]. The leading idea of this algorithm is to reduce the number of small regions (or segments) by merging of these small regions with adjacent larger regions. The main steps of the merging procedure are presented in Algorithm 2.

**Algorithm 2** Region Merging

```
Input: $L$: list of segments sorted by increasing area
Input: $n$ : # of segments given by Eq. (3.1) or ground truth
 1: $first \leftarrow$ first segment of $L$
 2: $N \leftarrow$ size of $L$
 3: while $N > n$ do
 4:     $neighbor \leftarrow$ find best segment neighboring $first$
 5:     $mergedsegment \leftarrow$ merge $first$ and $neighbor$ segments
 6:     add $mergedsegment$ to $L$
 7:     remove $first$ from $L$
 8:     remove $neigbor$ from $L$
 9:     $first \leftarrow$ first segment of $L$
10:     $N \leftarrow$ size of $L$
11: end while
```

The algorithm requires a preliminary sorting of the list of regions in terms of increasing areas. Basically, a small region is merged with its adjacent region with which it shares the longest path of its boundary (cf. line 4 of Algorithm 2). The stopping condition (cf. line 3 of Algorithm 2) is satisfied when the number of regions of the segmentation equals the value given by

$$n = \sum_{i=2}^{100} \frac{A_i}{a_i} \tag{3.1}$$

where $a_i$ stands for the relative region area associated to each bin of the histogram, and $A_i$ denotes the corresponding cumulative region area. Note that the index $i$ of the first bin of the histogram (i.e., $i = 1$ that corresponds to the interval $[0, 1[$ in the horizontal axis) is not considered in Eq. (3.1) because it accumulates areas of noisy regions.

Eq. (3.1) provides a number of segments that is similar to the one of the ground truth defined by the Princeton benchmarking [CGF09]. Nevertheless, as suggested by Chen et al. [CGF09], we also used the value of $n$ of the ground truth in order to not give advantage to any algorithm involved in the comparison undertaken in Section 3.8. Morever, only 2 out of 7 benchmark algorithms are able to calculate the number of segments automatically.

## 3.8 Experimental Results

### 3.8.1 Testing Setup

The HPO segmentation algorithm was designed and implemented on an Intel Core Duo 2.4 computer running a Mac OS X operating system, and using the OpenGL User Interface Library (GLUI), which is a C++ user interface library based on the OpenGL Utility Toolkit (GLUT).

### 3.8.2 Benchmarking

We have used the Princeton benchmarking to compare our algorithm to the state-of-the art of algorithms in mesh segmentation [CGF09]. This benchmarking tool provides 19 categories of models, with each category comprising 20 shapes (i.e. 380 models). This tool also supplies the ground truth of 4300 human-generated segmentations, being provided on average 11 human-generated segmentations concerning distinct poses of each shape.

The Princeton segmentation benchmarking provides quantitative comparisons of human-generated segmentations (i.e., ground truth) and computer-generated segmentations produced by the following seven algorithms: k-means [STK02], random walks [LHMR08], fitting primitives [AFS06], normalized cuts [GF08] randomized cuts [GF08], core extraction [KLT05] and shape diameter function [SSCO08]. It is worth noting that the benchmarking software does not include the codes of those seven algorithms, but only segmentations produced by them, which work as input data for the benchmarking software. Therefore, before running the benchmarking itself for the eight algorithms (including ours), we added the segmentations generated by our algorithm to the benchmark.

In order to compare the segmentations produced by those eight algorithms (including our algorithm), the Princeton benchmarking tool uses the following four metrics: *Rand Index* (RI), *Cut Discrepancy* (CD), *Consistency Error* (CE), and *Hamming Distances* (HD). The CD metric focuses on boundary errors, while the other three metrics focus on region dissimilarities. Figure 3.9 compares our algorithm to the seven segmentation algorithms

Table 3.1: Comparison of segmentation algorithms for each model category (19 categories with 20 models each).

| Object Category | Benchmark Metrics and Algorithms | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RI | | | | | | | | CD | | | | | | | |
| | HPO | NormCuts | RandCuts | ShapeDiam | CoreExtra | RandWalks | FitPrim | KMeans | HPO | NormCuts | RandCuts | ShapeDiam | CoreExtra | RandWalks | FitPrim | KMeans |
| Human | 1 | 5 | 2 | 4 | 7 | 8 | 3 | 6 | 3 | 5 | 4 | 2 | 6 | 8 | 1 | 7 |
| Cup | 1 | 3 | 2 | 6 | 4 | 5 | 7 | 8 | 1 | 2 | 5 | 3 | 4 | 8 | 6 | 7 |
| Glasses | 2 | 3 | 1 | 5 | 7 | 8 | 6 | 4 | 2 | 4 | 1 | 5 | 6 | 8 | 3 | 7 |
| Airplane | 2 | 5 | 3 | 1 | 7 | 8 | 4 | 6 | 1 | 4 | 3 | 2 | 8 | 7 | 5 | 6 |
| Ant | 3 | 4 | 2 | 1 | 5 | 6 | 7 | 8 | 3 | 4 | 1 | 2 | 5 | 6 | 7 | 8 |
| Chair | 3 | 1 | 6 | 2 | 5 | 4 | 8 | 7 | 2 | 3 | 4 | 1 | 8 | 6 | 5 | 7 |
| Octopus | 1 | 4 | 5 | 2 | 3 | 6 | 8 | 7 | 1 | 5 | 3 | 2 | 4 | 7 | 6 | 8 |
| Table | 1 | 2 | 7 | 4 | 6 | 3 | 5 | 8 | 1 | 2 | 7 | 5 | 6 | 3 | 4 | 8 |
| Teddy | 1 | 5 | 2 | 3 | 4 | 6 | 7 | 8 | 2 | 6 | 1 | 3 | 7 | 4 | 5 | 8 |
| Hand | 2 | 4 | 1 | 6 | 3 | 8 | 7 | 5 | 2 | 3 | 1 | 7 | 4 | 8 | 5 | 6 |
| Plier | 5 | 4 | 2 | 8 | 1 | 6 | 3 | 7 | 3 | 4 | 2 | 8 | 1 | 5 | 6 | 7 |
| Fish | 1 | 5 | 4 | 2 | 3 | 6 | 8 | 7 | 1 | 3 | 6 | 2 | 8 | 4 | 7 | 5 |
| Bird | 1 | 7 | 2 | 3 | 4 | 8 | 6 | 5 | 1 | 6 | 3 | 2 | 4 | 8 | 7 | 5 |
| Armadillo | 4 | 7 | 1 | 2 | 8 | 5 | 3 | 6 | 3 | 5 | 1 | 2 | 8 | 6 | 4 | 7 |
| Bust | 2 | 5 | 1 | 3 | 7 | 6 | 4 | 8 | 5 | 3 | 1 | 2 | 8 | 6 | 4 | 7 |
| Mech | 1 | 2 | 6 | 3 | 7 | 4 | 5 | 8 | 1 | 3 | 6 | 2 | 8 | 4 | 5 | 7 |
| Bearing | 3 | 4 | 2 | 1 | 8 | 7 | 5 | 6 | 3 | 4 | 2 | 1 | 8 | 6 | 7 | 5 |
| Vase | 1 | 5 | 2 | 4 | 3 | 6 | 7 | 8 | 1 | 4 | 2 | 3 | 7 | 5 | 6 | 8 |
| FourLeg | 2 | 5 | 3 | 1 | 6 | 8 | 4 | 7 | 3 | 6 | 4 | 1 | 8 | 7 | 2 | 5 |
| Average | 1 | 4 | 2 | 3 | 5 | 7 | 6 | 8 | 1 | 4 | 3 | 2 | 6 | 7 | 5 | 8 |
| | GCE | | | | | | | | HD | | | | | | | |
| | HPO | NormCuts | RandCuts | ShapeDiam | CoreExtra | RandWalks | FitPrim | KMeans | HPO | NormCuts | RandCuts | ShapeDiam | CoreExtra | RandWalks | FitPrim | KMeans |
| Human | 5 | 1 | 2 | 8 | 4 | 3 | 7 | 6 | 4 | 2 | 1 | 7 | 3 | 5 | 6 | 8 |
| Cup | 1 | 4 | 2 | 3 | 6 | 5 | 7 | 8 | 1 | 3 | 2 | 6 | 5 | 4 | 7 | 8 |
| Glasses | 2 | 3 | 1 | 4 | 5 | 6 | 8 | 7 | 2 | 3 | 1 | 5 | 6 | 7 | 8 | 4 |
| Airplane | 2 | 5 | 3 | 1 | 4 | 7 | 6 | 8 | 2 | 4 | 3 | 1 | 6 | 7 | 5 | 8 |
| Ant | 3 | 5 | 2 | 1 | 4 | 6 | 7 | 8 | 3 | 4 | 2 | 1 | 5 | 6 | 7 | 8 |
| Chair | 4 | 2 | 6 | 1 | 3 | 5 | 8 | 7 | 3 | 1 | 5 | 2 | 4 | 6 | 8 | 7 |
| Octopus | 2 | 5 | 3 | 1 | 4 | 6 | 7 | 8 | 1 | 5 | 3 | 2 | 4 | 6 | 8 | 7 |
| Table | 2 | 1 | 7 | 4 | 6 | 3 | 5 | 8 | 1 | 2 | 7 | 4 | 5 | 3 | 6 | 8 |
| Teddy | 2 | 7 | 1 | 3 | 6 | 4 | 5 | 8 | 1 | 5 | 2 | 3 | 4 | 6 | 7 | 8 |
| Hand | 5 | 2 | 1 | 6 | 4 | 3 | 8 | 7 | 4 | 2 | 1 | 7 | 3 | 5 | 8 | 6 |
| Plier | 3 | 6 | 2 | 5 | 1 | 4 | 7 | 8 | 3 | 4 | 2 | 7 | 1 | 5 | 6 | 8 |
| Fish | 1 | 5 | 3 | 2 | 4 | 7 | 8 | 6 | 1 | 5 | 4 | 2 | 3 | 6 | 8 | 7 |
| Bird | 3 | 6 | 1 | 2 | 4 | 5 | 8 | 7 | 2 | 5 | 1 | 4 | 3 | 6 | 8 | 7 |
| Armadillo | 5 | 4 | 2 | 3 | 1 | 7 | 6 | 8 | 3 | 6 | 1 | 2 | 4 | 7 | 5 | 8 |
| Bust | 3 | 7 | 1 | 6 | 2 | 5 | 4 | 8 | 3 | 7 | 1 | 4 | 2 | 5 | 6 | 8 |
| Mech | 1 | 2 | 6 | 3 | 4 | 5 | 7 | 8 | 1 | 2 | 4 | 6 | 7 | 3 | 5 | 8 |
| Bearing | 3 | 1 | 2 | 4 | 6 | 7 | 5 | 8 | 2 | 4 | 3 | 1 | 8 | 6 | 5 | 7 |
| Vase | 2 | 5 | 1 | 3 | 4 | 6 | 7 | 8 | 1 | 5 | 2 | 4 | 3 | 6 | 7 | 8 |
| FourLeg | 5 | 4 | 3 | 2 | 1 | 6 | 7 | 8 | 5 | 4 | 3 | 1 | 2 | 6 | 7 | 8 |
| Average | 1 | 5 | 2 | 3 | 4 | 6 | 7 | 8 | 1 | 5 | 2 | 3 | 4 | 6 | 7 | 8 |

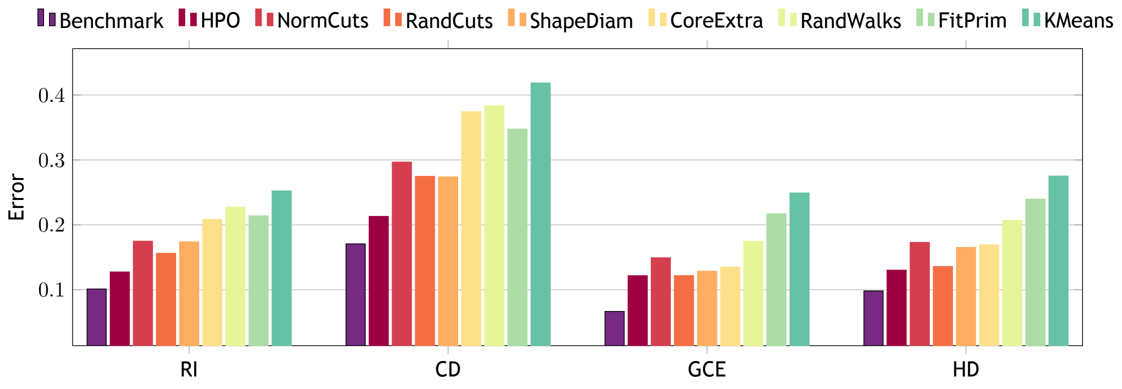


Figure 3.9: Comparison of segmentation algorithms with different metrics.

of the benchmark, with reference to the aforementioned four benchmark metrics. Note that the metrics are consistent with one another, in the sense that segmentation algorithms have the same relative performance with regard to such metrics. A more detailed comparison is presented in Table 3.1, whose results show that *on average* the HPO algorithm ranks first for the four benchmarking metrics.

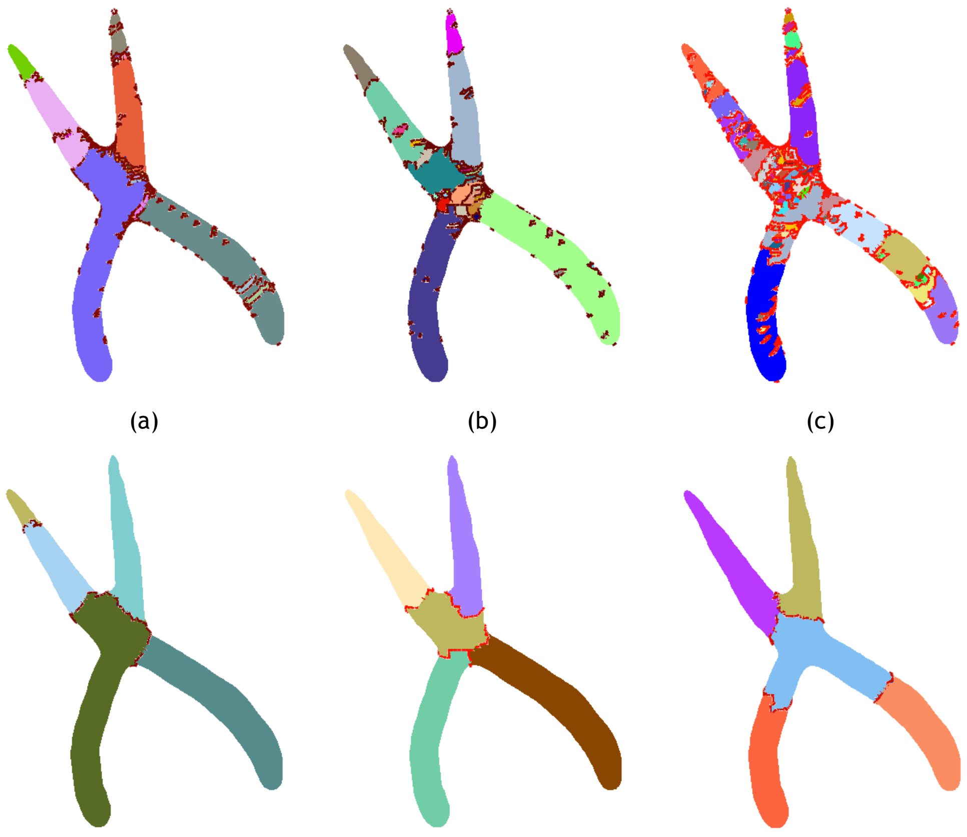


Figure 3.10: A plier (model 202 of Princeton benchmark) with different levels of noise, as segmented by the HPO algorithm: (a) original mesh after its filling (top) and merging (bottom) steps; (b) mesh with noise generated by vertex displacement of 0.005, after its filling (top) and merging (bottom) steps; (c) mesh with more noise generated by vertex displacement of 0.007, after its filling (top) and merging (bottom) steps.

### 3.8.3 Discussion and Limitations

In a previous version of the HPO algorithm, the smoothing step was not part of the algorithm. Even so, the HPO algorithm was ranked first for the RI, CD, and HD metrics, while it ranked second for the CE metrics. In this latter metric, the RandCuts algorithm performed better than our algorithm, but not that much. This happens because CE metrics provide better results when the number of regions is higher than the human-generated regions of the ground truth, as it is the case of the RandCuts algorithm. In fact, according to Chen et al. [CGF09], the CE metrics have the "problem that they normally provide better scores when two models have different numbers of segmentations". Consequently, we can fit small regions produced by RandCuts in bigger human-generated regions. By contrast, the HPO algorithm tends to produce the same number of regions as in the ground truth, so it becomes more difficult to fit a HPO region in a ground truth region. This drawback of HPO algorithm in respect to CE metrics was solved using the additional step of mesh smoothing for noisy meshes, as it is the case of Armadillo's mesh shown in Figure 3.8. In this way, the HPO algorithm ends up being more robust in respect to mesh noise.

Although the mesh noise is a problem for many algorithms, it is not less true that the absence of mesh noise may be also a problem for a proper segmentation of a given mesh. For example, the HPO algorithm cannot produce an adequate segmentation for a T-shape provided that its T-shaped back surface (and T-shaped front surface) is made of coplanar triangles. Consequently, there is no cycle of concave edges separating the horizontal part from the vertical part of a T-shape. This explains why the regions concerning the arms of the man placed in the lower right corner in Figure 3.1 do not end at the clavicles.

A possible solution to this problem would be to add noise to the mesh. For example, the plier depicted in Figure 3.10(a) essentially is a X-shaped model, which is not very well segmented because the central part of its X-shaped back surface (and X-shaped front surface) consists of coplanar triangles. By adding noise to the plier mesh (cf. Figure 3.10(b)), we end up obtaining a perceptually better segmentation of the plier. But, as shown in Figure 3.10(c), adding more and more noise to the mesh is not a good solution, because much likely this procedure undermines the segmentation. Possible, a better solution would be to add noise to the mesh locally (instead of globally) where needed, but this remains an open issue for future work because currently there is no way to know in advance when we need to do so.

Anyway, the important fact to retain is that the HPO segmentations agree largely with the human-generated regions of the ground truth. A number of HPO segmentations of the entire set of models (among those 380 models of the Princeton benchmark repository) are depicted in Figure 3.1; also the HPO segmentations of the entire family of 20 bears is shown in Figure 3.11. Moreover, the HPO algorithm compares to more recent algorithms proposed in the literature, namely those due to Au et al. [AZC+12] and Wang et al. [WLAT14].

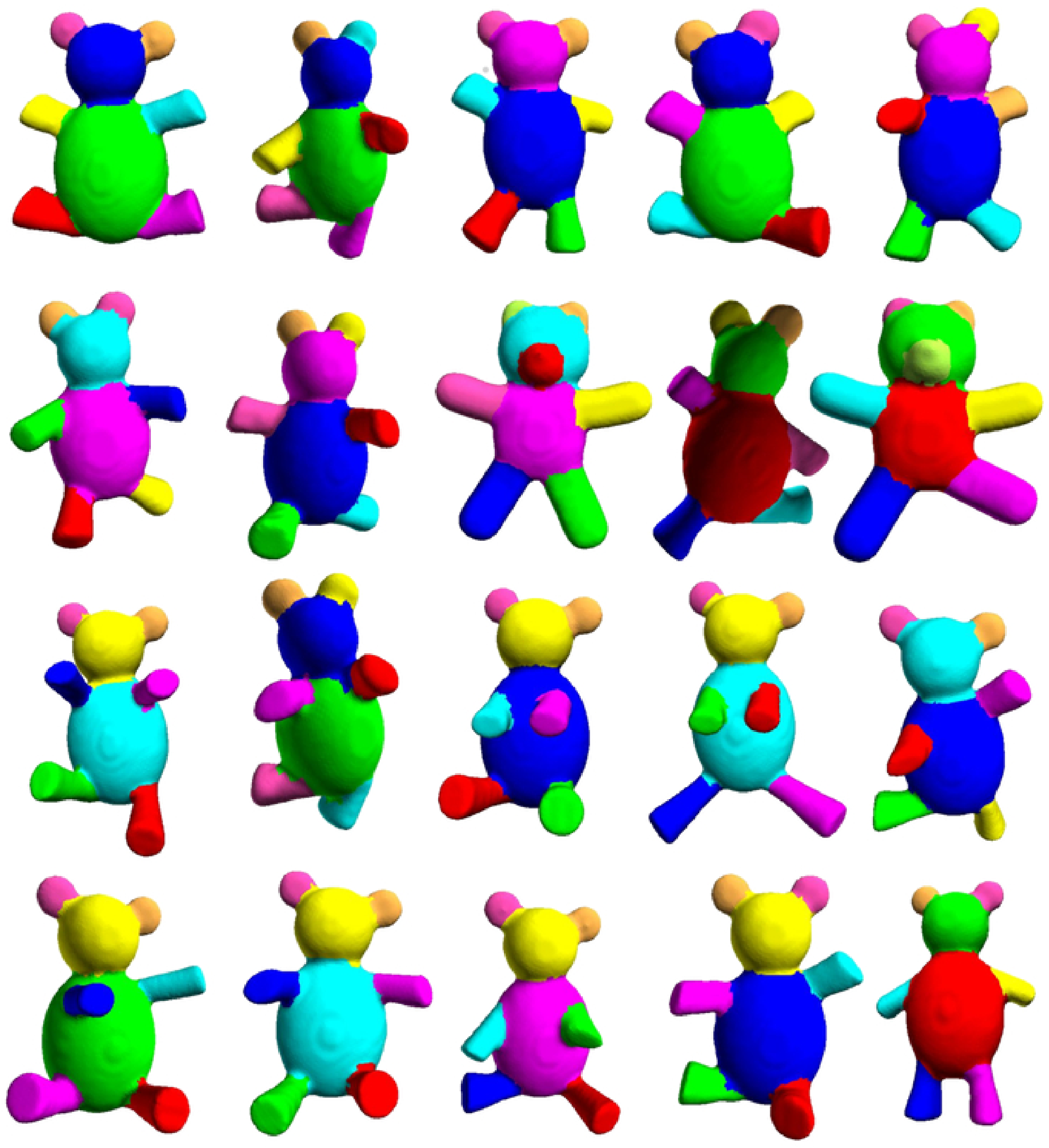


Figure 3.11: Models of the category of bears segmented by the HPO algorithm.

## 3.9 Conclusions

We have introduced here a new mesh segmentation algorithm, named HPO algorithm, that outperforms the state-of-the-art algorithms in mesh segmentation in the sense that it ranks first in Princeton benchmarking for all its four metrics. This means that the HPO segmentations largely match the human-generated segmentations of the ground truth, even using only geometric criteria (i.e., the PMT criterion, interior condition, boundary condition, and the door-in-door-out principle) and reasoning, without the need of computing convex hulls as usual in many convexity-based algorithms.

At our best knowledge, there is not any other contour-based mesh segmentation algorithm in the literature, other than the HPO algorithm itself. The idea of a contour-based segmentation taken from image analysis and processing has led us to the concept of relaxed convexity. This in some extent avoids the over-segmentation of freeform objects. For over-segmented or noisy meshes, we have also introduced a cumulative histogram that is capable of distinguishing a noisy mesh from a mesh without significant noise, as well as a mesh smoothing filter. Furthermore, in practice, it is enough to use the region filling step to segment non-freeform objects correctly.

## Related Publications

The work described in this chapter originated a publication as indicated below:

> *Rui S.V. Rodrigues, José F.M. Morgado, and Abel J.P. Gomes. 2015. A contour-based segmentation algorithm for triangle meshes in 3D space.* Computers & Graphics *49 (2015), pp. 24–35*

# Chapter 4

# A Shape-Preserving Multiresolution Segmentation Scheme for Triangle Meshes with Creases and Apices

A plethora of segmentation techniques, as well as a number of multiresolution techniques, for triangle meshes already exist in the literature. However, it is not so common to find algorithms and data structures that fuse these two concepts, multiresolution and segmentation, into a symbiotic multi-resolution scheme for both plain and segmented meshes, in which a plain mesh is understood as a mesh with a single segment. In this chapter, we introduce such a novel multiresolution segmentation scheme, called extended Ghost Cell (xGC) scheme. This scheme preserves the shape of the meshes in both global and local terms, i.e., mesh segments and their boundaries, as well as creases and apices are preserved, no matter the level of resolution we use for simplification/refinement of the mesh. Moreover, unlike other segmentation schemes, it was made possible to have adjacent segments with two or more resolution levels of difference. This is particularly useful in computer animation, mesh compression and transmission, geometric computing, scientific visualization, and computer graphics.

## 4.1 Introduction

With the advent of modern sampling tools and systems (e.g., laser scanners), the complexity of 3D models used in geometric computing and computer graphics has increased at least in an order of magnitude. Also, the triangle mesh is the standard representation for 3D objects, in largely because it easily converts into primitives of the 3D graphics pipeline for rendering purposes. Note that triangle meshes also include geometry and topology information, which —in case they possess a large number of triangles— spend a lot in computation, storage, transmission, and display tasks. Hence, multi-resolution techniques have emerged as a solution to represent data with multiple levels of detail (LOD) [GH97], or levels of resolution, mainly for progressive transmission and visualization purposes.

In 3D scenes, it is quite common to apply multiresolution techniques to the visible parts of meshes (i.e., those seen by the viewer), particularly in the context of compression and transmission of geometry [KL01] [YKK04] [KCL06] [MGH11] [MGH13]. In fact, it is also common to use the distance to the viewer as a multiresolution criterion [Hop97] [ESV99] [Paj01] [CKLL09], in particular in terrain modeling and rendering. Furthermore, we can say that there are parts of the mesh with more or less detail, which we might call *segments*, not it were the fact that these parts depend on the viewer's position. At best, these parts of the mesh that are not perceptually meaningful might be called *charts*. That is, *mesh segmentation* is a particular type of *mesh chartification* in the sense that a segment is a chart endowed with perceptual meaningfulness, but a chart is not necessarily a segment. Therefore, mesh segmentation consists in dividing a mesh into meaningful regions, also called segments (cf. Figure 4.1).

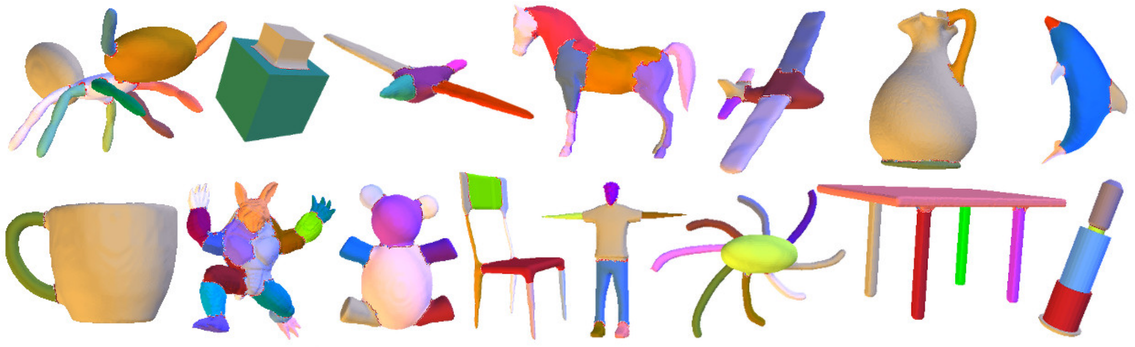


Figure 4.1: Segmentations of different meshes.

This procedure has been used in a number of tasks that take into account such meaningful components of objects, namely shape processing [KT03] [LKA06], shape matching [FKS+04] [MW99] [ZG06], 3D modeling, collision detection [LWTH01], skeletonization, metamorphosis, mesh deformation [HSL+06], animation [VF14], parameterization [ZMT05], mesh editing [STL06], and texturing of 3D shapes [SWG+03], just to mention a few [GF08] [CKGK11] [ZSSL15].

Both mesh segmentation and multi-resolution meshes have been widely discussed in the fields of applied geometry, computer aided design, and computer graphics. But, it is a bit surprising that a synthesis of these two techniques has never emerged in the literature in a convincing way, as shown in Table 4.1, where we can observe that only a few schemes have tried to combine chartification with multiresolution, and much less segmentation with multiresolution. In fact, mesh segmentation techniques have been mostly developed for plain meshes, but not for multi-resolution meshes. On the other hand, as far as we know, there is not any multi-resolution scheme for segmented meshes that copes with segments having a resolution differential greater or equal to two. The exception is the multiresolution scheme put forward by González et al. [GGC+09] (see Table 4.1), but even in this case there is not the desirable symbiosis between segmentation and multiresolution because the boundary preservation of material color-based facet clusters (or sub-ojects) develops from the idea of descontinuity preservation due to Hope [Hop99]. Unlike the current state-of-the-art algorithms, we here propose a multi-resolution scheme for segmented meshes that supports segments with distinct resolutions.

In addition to the lack of an integrated multiresolution segmentation scheme, another problem with the segmentation of a multiresolution mesh is that there is no guarantee that the segmentation holds after applying the multi-resolution operators, i.e., refinement and simplification operators. This is illustrated in Figure 4.2, where we can observe that after simplify twice the mesh, its number of segments does not longer hold. In short, the ordinary multi-resolution schemes do not preserve the meaningful shape of a mesh (or object). Besides, the simplification of a mesh tends to eliminate its concavities, creases, and apices.

Note that, with the exception of a few solutions (see, for example, Hoppe [Hop99] and Silva and Gomes [SG04] in Table 4.1), there is no guarantee that refinement and simplification operators preserve eventual creases and apices in the mesh. However, these solutions generally are averse to segmentation techniques, i.e., they are not approached in the context of mesh segmentation.

Table 4.1: Schemes that somehow combine segmentation, multiresolution, discontinuities (i.e., apices and creases), and preservation of the chart/segment/cluster boundaries.

| Method | Charts | Segments | Boundary Preservation | Apices | Creases | MR | PM | PM Cl. | Segments With Distinct Resolutions | Application |
|---|---|---|---|---|---|---|---|---|---|---|
| Garland and Heckbert [GH98] | | | • | | • | | • | • | | MRV |
| Hoppe [Hop99] | | | • | | • | • | • | • | | MRV |
| Kim and Lee [KL01] | • | | | | | • | | | | MRV |
| Zuckerberger et al. [ZTS02] | • | | • | | | | | | | MS |
| Silva and Gomes [SG04] | | | | • | • | • | • | | | MRV |
| Vivodtzev et al. [VBLT05] | • | | • | | | | • | • | | MS |
| Kim et al. [KYL06] | • | • | • | • | • | | | | | MS |
| Kim et al. [KCL06] | • | | | | | • | | | | MRC |
| Cheng et. al. [CLJ07] | | • | | | | • | | | | MRC |
| González et al. [GGC$^+$09](*) | | | • | | | • | • | • | • | MRV |
| Maglo et al. [MGH11] | • | | • | | | | | | | MRC |
| Thomas et al. [TNB11] | | | • | | | | | • | | MS |
| Maglo et al. [MGH13] (**) | • | | | | | • | • | | | MRC |
| xGC | • | • | • | • | • | • | • | | • | MRV |

* Object defined as set of sub-objects;
** Cluster defined by merged vertices;
MR: Multiresolution; PM: Plain Mesh; PM Cl: Plain Meshes with Clusters;
MRV: Multiresolution to Visualization; MS: Mesh Simplification; MRC: Multiresolution to Compression.

Summing up, preserving the shape of a multiresolution segmented mesh requires the following:

- *Segments and their boundaries*. First, a multi-resolution mesh preserves its shape if their segments are preserved, independently of the resolution of each segment. That is, the meaningful 2-dimensional shapes (i.e., regions) must be preserved.

- *Creases and apices*. Second, creases and apices (e.g., edges and vertices of a cube) must be also preserved. That is, the 1- and 0-dimensional local shapes of a mesh must be also preserved.

Thus, in this chapter, we engender a solution that solves these issues in a straightforward manner. Basically, the solution focuses on the boundaries (i.e., 1-cycles) of segments, creases, and apices. In the case of segments, there is no need to identify their boundaries because they are given beforehand by the segmentation itself. In regard to creases and apices, which consist of points of high curvature, they are easily filtered out using the norm of covariance matrix, as shown by Mangan et al. in [MW99].

The main contributions of this chapter are the following:

- At our best knowledge, here we propose the first data structure for multi-resolution segmented meshes in an integrated or symbiotic manner.
- We also introduce a shape-preserving simplification operator for segmented meshes. This requires that the boundaries that separate mesh segments from each other must hold after all.
- Creases and apices are also preserved after finding them through the covariance matrix norm in the context of multiresolution segmented meshes.
- Simplification and refinement operators can be applied to the entire mesh or to any number of segments in an independent manner. This means the segments may have distinct levels of detail.

So, the remainder of the chapter is organized as follows. Section 4.2 describes the xGC-based multi-resolution scheme as an extension of the GC-based scheme, as needed for mesh segmentation. Section 4.3 describes the techniques to preserve shape of segmented meshes and their distinctive geometric features. Section 4.4 describes the extended versions of mesh collapse and vertex split used in the context of segmented meshes. Section 4.5 details how to preserve segments of a mesh independently of the level of detail. Section 4.6 advances with the most important results obtained with xCG-based multi-resolution scheme for segmented meshes. Section 4.7 concludes the chapter with some hints for future work.

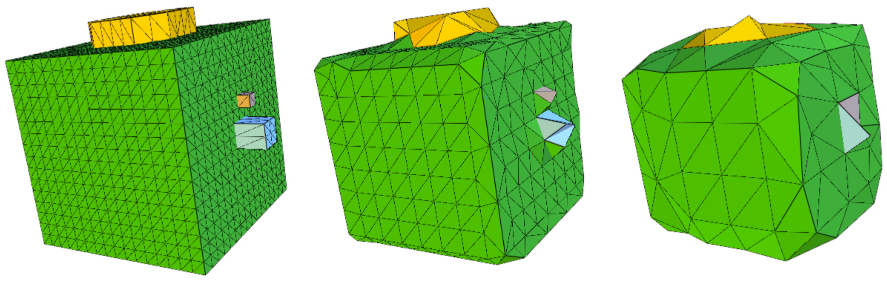


Figure 4.2: Multi-resolution schemes do not preserve the meaningful shape.

## 4.2 Multi-resolution Segmented Meshes

### 4.2.1 Multi-resolution schemes

In the literature, we find mainly two families of multi-resolution schemes for plain meshes, but there is none for segmented meshes. Anyway, let us briefly review the former schemes because the one here proposed for segmented meshes can be seen as an extended multi-resolution scheme. Those two families include plain subdivision schemes and selective subdivision schemes; note that subdivision is by definition a recursive operation.

A *plain subdivision scheme* subdivides all polygons in the same manner for each level of detail [SZD+00] [DFM02]; for example, by subdividing every single triangle into four smaller triangles, we end up getting a progressively smoother mesh. In a word, this scheme operates globally on the mesh in the sense that every single triangle is equally refined (or simplified) at each level of detail.

In a *selective subdivision scheme*, not all the triangles are necessarily subject to a refinement/simplification procedure at each level of detail [Gar99]. This is particularly useful in a shape-preserving scenario, as it the case of segmented meshes, as well when we intend to preserve creases and apices in the mesh.

In this chapter, we take advantage of the selective subdivision scheme proposed by Rodrigues et al. [RMSG07], called GC-based scheme, with GC standing for 'ghost cell'. GC-based scheme has been extended in order to cope with mesh segments, resulting in a new scheme called xGC ('extended ghost cell'). As shown further ahead, its selective nature enables us to preserve the shape of a mesh along the segment boundaries, as well as on creases and apices. The xGC data structure builds on the GC data structure [RMSG07], which in turn builds on the AIF data structure [SG03], also known as Woo's data structure [Woo85], as illustrated in Figure 4.3.

### 4.2.2 AIF data structure

The AIF (adjacency and incidence framework) data structure is an optimal $C_4^9$ data structure for plain polygonal meshes [SG03] (cf. Ni e Bloor [NB94] about the optimality of $C_4^9$), which encodes the following topological relationships: $V \prec E$ e $E \prec F$, and their inverses $E \succ V$, and $F \succ E$. Therefore, the edge is the central topological entity of the data structure, as shown in Figure 4.3 (code in black). Here, the incidence relationship $V \prec E$ denotes the edges incident at a given vertex, which is represented by `Array<Edge*> aoe;` in the class `V` for vertices, while relationship $E \prec F$ stands for the faces incident on a given edge, which is accounted for by `Array<Face*> aof;` in the class `E` for edges. Conversely, the adjacency relationship $E \succ V$ represents the vertices bounding a given edge, as encoded by the variables `v1` and `v2` in the class `E` for edges; also, $F \succ E$ denotes the edges bounding a given face, as encoded by the variable `Array<Edge*> aoe;` in the class `F` for faces. In addition to the classes `V` for vertices, `E` for edges, and `F` for faces, we also need the central repository for meshes, which is represented by the class `MESH`.

### 4.2.3 GC-based data structure

The GC-based data structure was designed for multi-resolution meshes by Rodrigues et al. [RMSG07], in conformity with a selective subdivision scheme. In Figure 4.3, GC-based data structure includes the AIF data structure code in black and the code in brown. For example, the brown code in the class `MESH` concern the arrays of vertices, edges,

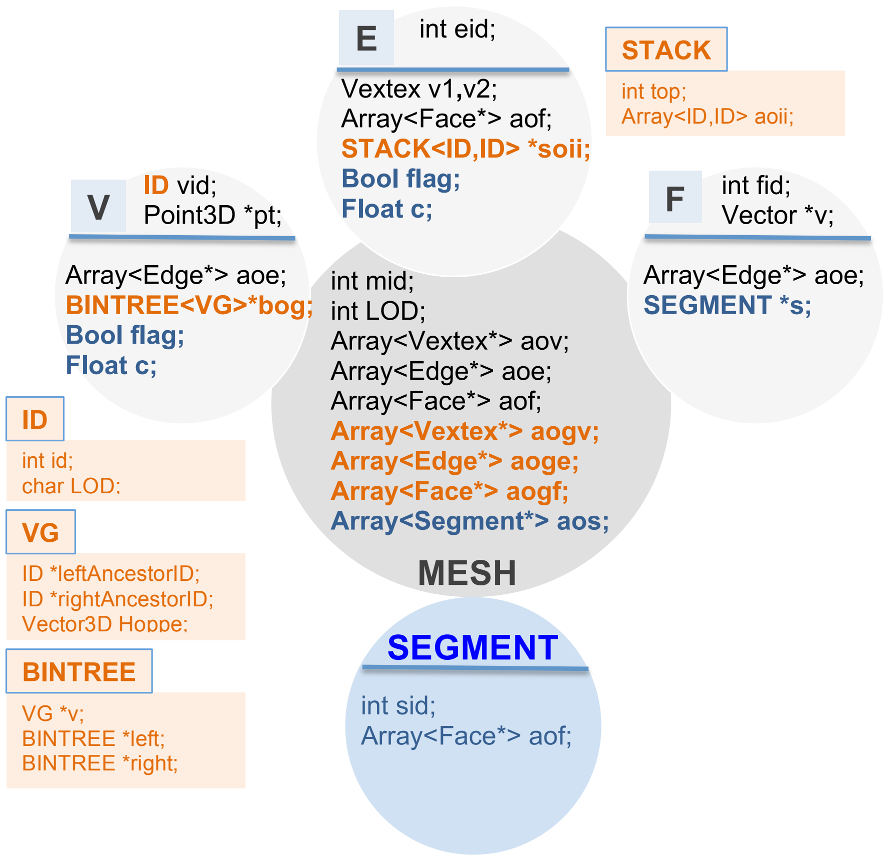
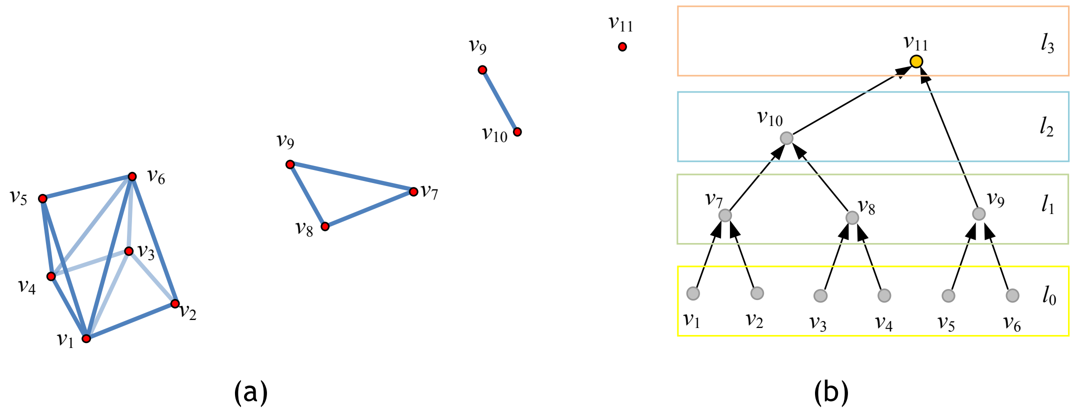


Figure 4.3: xGC-based data structure.

and faces that became invisible (or ghost) to the viewer as a consequence of simplification operations, i.e., edge collapsing; hence, the arrays of ghost simplices for vertices (`Array<Vertex*> aogv`), edges (`Array<Edge*> aoev`), and faces (`Array<Vertex*> aogf`) in the class `MESH`.

The two most important supplementary GC-based data structures that we need in multiresolution meshes are those associated to vertices and edges, i.e., `BINTREE<VG> *bog` and `STACK<ID,ID> *soii`. Note the presence of Hoppe's vector in the class `VG`, as necessary to refine the mesh later on. The bintree associated to each vertex represents its genealogy, that is, its vertex ancestors. The stack associated to each edge represents its state at different levels of detail (LODs).

As illustrated in Figure 4.4(b), the vertex bintree grows from the bottom to top during the process of mesh simplification, i.e., the vertices at the tree leaves (level $l_0$) work as ancestors for those vertices belonging to higher levels of simplification. In parallel, as a consequence of edge collapsing inherent to mesh simplification, edges end up turning into ghost edges as pictured by the stacking of each edge in Figure 4.4(c); for example, the edge $(v_1, v_2)$ turns into $(v_7, v_7)$ at the first level, i.e., it becomes a ghost edge because it collapses into a single vertex $v_7$.



| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $(v_{11},v_{11})$ | $(v_{11},v_{11})$ | $(v_{11},v_{11})$ | $(v_{11},v_{11})$ | $(v_{11},v_{11})$ | $(v_{11},v_{11})$ | $(v_{11},v_{11})$ | $(v_{11},v_{11})$ | $(v_{11},v_{11})$ | $(v_{11},v_{11})$ | $(v_{11},v_{11})$ | $(v_{11},v_{11})$ | $l_3$ |
| $(v_{10},v_{10})$ | $(v_{10},v_{10})$ | $(v_{10},v_{10})$ | $(v_{10},v_9)$ | $(v_{10},v_9)$ | $(v_{10},v_{10})$ | $(v_{10},v_9)$ | $(v_{10},v_{10})$ | $(v_{10},v_9)$ | $(v_{10},v_9)$ | $(v_{10},v_9)$ | $(v_9,v_9)$ | $l_2$ |
| $(v_7,v_7)$ | $(v_7,v_8)$ | $(v_7,v_8)$ | $(v_7,v_9)$ | $(v_7,v_9)$ | $(v_7,v_8)$ | $(v_7,v_9)$ | $(v_8,v_8)$ | $(v_8,v_9)$ | $(v_8,v_9)$ | $(v_8,v_9)$ | $(v_9,v_9)$ | $l_1$ |
| $(v_1,v_2)$ | $(v_1,v_3)$ | $(v_1,v_4)$ | $(v_1,v_5)$ | $(v_1,v_6)$ | $(v_2,v_3)$ | $(v_2,v_6)$ | $(v_3,v_4)$ | $(v_3,v_6)$ | $(v_4,v_5)$ | $(v_4,v_6)$ | $(v_5,v_6)$ | $l_0$ |

(c)

Figure 4.4: Illustrating GC-based mesh simplification of a wedge of 6 points via edge collapsing: (a) across four LODs, $l_0, \ldots, l_3$; (b) bintrees for vertice: $v_1$ and $v_2$ (level $l_0$) collapse into $v_7$ (level $l_1$), $v_3$ and $v_4$ (level $l_0$) collapse into $v_8$ (level $l_1$) , $v_5$ and $v_6$ (level $l_0$) collapse into $v_9$ (level $l_1$); $v_7$ and $v_8$ (level $l_1$) collapse into $v_{10}$ (level $l_2$); $v_9$ (level $l_1$) and $v_{10}$ (level $l_2$) collapse into $v_{11}$ (level $l_3$); (c) stacks for 12 edges from level $l_0$ to level $l_3$, i.e., each edge stack has 4 edges, except the stack of $(v_5, v_6)$ that possesses 3 edges.

### 4.2.4 xGC-based data structure

The xGC-based data structure extends the GC-based structure in order to represent multi-resolution segmented meshes; hence, the class `SEGMENT` for segments in Figure 4.3. The additional blue code in Figure 4.3 was exclusively written for segment meshes. It is clear that a segment is a connected set of faces, as indicated by the field `Array<Face*> aof;` in the class `SEGMENT`. Conversely, the class `F` has now the field `SEGMENT *s;` that stands the segment of each face. Additionally, the classes `V` and `E` include the field `Bool flag;` to indicate whether or not a vertex or an edge are amenable to simplification and refinement operations. In our case, vertices and edges belonging to the boundaries of segments, as well as those belonging to creases and apices, are not amenable to such multi-resolution operations. Obviously, the class `MESH` also includes an array of segments.

## 4.3 Shape-Preserving Multiresolution Meshes

Every multi-resolution scheme is centered at the sort of topological entity (i.e., either vertex or edge) to be modified [DFM02]. That is, there are vertex-based and edge-based multi-resolution schemes. Edge-based multi-resolution schemes make usage of edge collapse and vertex split operations [Hop96] [ESV99] [DFM$^+$05], while vertex-based multi-resolution schemes use vertex insertion and removal operations [SZL92] [CCMS97]) [RB93] [LE97].

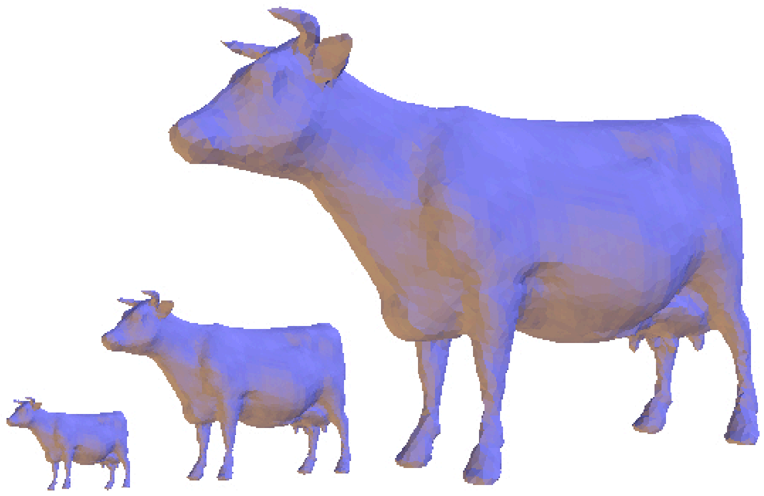


Figure 4.5: A non-segmented mesh with three levels of detail depending on the distance to the viewer. The smaller mesh is more far away from the viewer and has thus a less number of visible vertices, edges, and faces. We used the dihedral angle of each edge to decide on its collapse, so that the global shape of the mesh is preserved.

Our xGC-based data structure sustains on an edge-based multiresolution scheme. In ordinary edge-based multiresolution schemes (i.e., those for plain meshes), we follow a criterion (or criteria) to choose the next edge to collapse. Examples of criteria are the energy minimization [HDD$^+$93] [Hop96], vertex minimization error [GH97], and dihedral angle [SG04]. If we prevent edges with significant dihedral angles from collapsing, we end up preserving the global shape of a given mesh, as it is the case of the mesh of the cow pictured in Figure 4.5.

### 4.3.1 Preserving mesh segments

However, when using segmented meshes, the dihedral angle criterion does not preserve the zonal shape of a mesh, i.e., the perceptually significant regions (e.g., legs, hands, etc.) of a mesh are not preserved. This is so because the dihedral angle of some edges bounding a given region may be small; consequently, such edges are eligible for collapsing. In order to keep perceptually significant regions independently of the level of detail used in multiresolution meshes, boundary edges of regions cannot be subject to collapsing operations. In other words, we preserve the shape of a segmented

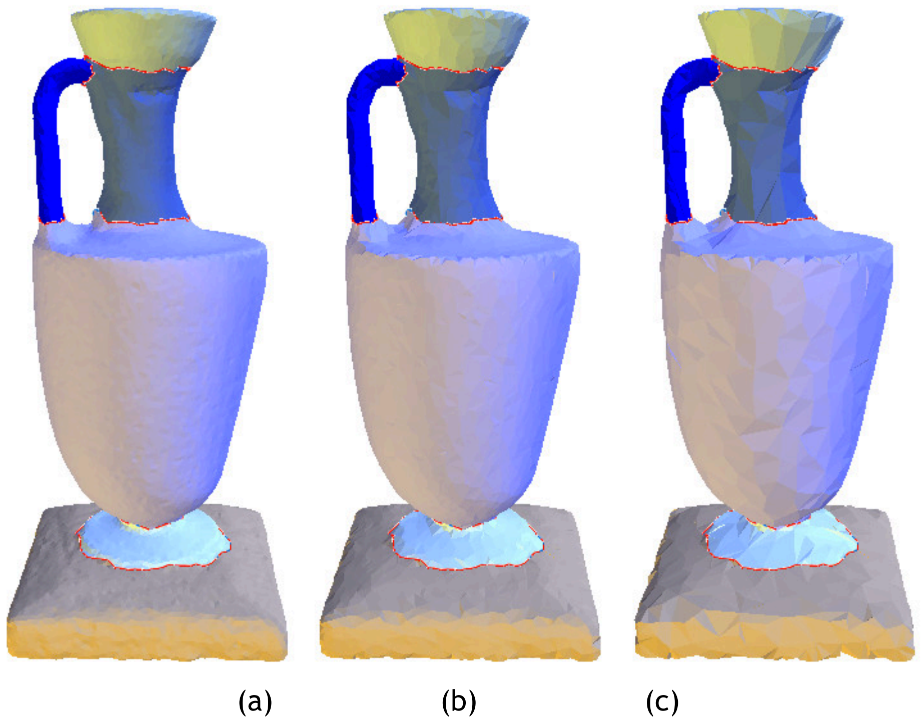


Figure 4.6: Preserving the segment boundaries independently of the level of detail (LOD): (a) the most detailed amphora; (b) the same amphora after simplifiying it in 2 LODs; (c) the same amphora after simplifiying it in 4 LODs;

mesh by preserving the boundary of each region, and this is done independently of the levels of detail of adjacent regions. For example, in Figure 4.6, the boundaries (in red) of the mesh segments remain unchanged, yet the their levels of detail are distinct in Figure 4.6(a), (b), and (c).

### 4.3.2 Preserving mesh creases and apices

Mesh creases and apices are edges and vertices, respectively, that are characterized by high values of curvature [MW99]. This combinatorial curvature at a vertex can be calculated through the norm of covariance matrix of the normals $\{\mathbf{N}_i\}$ of its incident faces, with $\mathbf{N}_i = (x_i, y_i, z_i)$ e $i = 1, \ldots, n$. For that purpose, we need to determine the mean vector of normals given by:

$$\bar{\mathbf{N}} = (\bar{x}, \bar{y}, \bar{z}), \text{ com } \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \; \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i, \text{e } \bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i. \tag{4.1}$$

Recall that the general formula to calculate the covariance of two variables $u \in \{x, y, z\}$

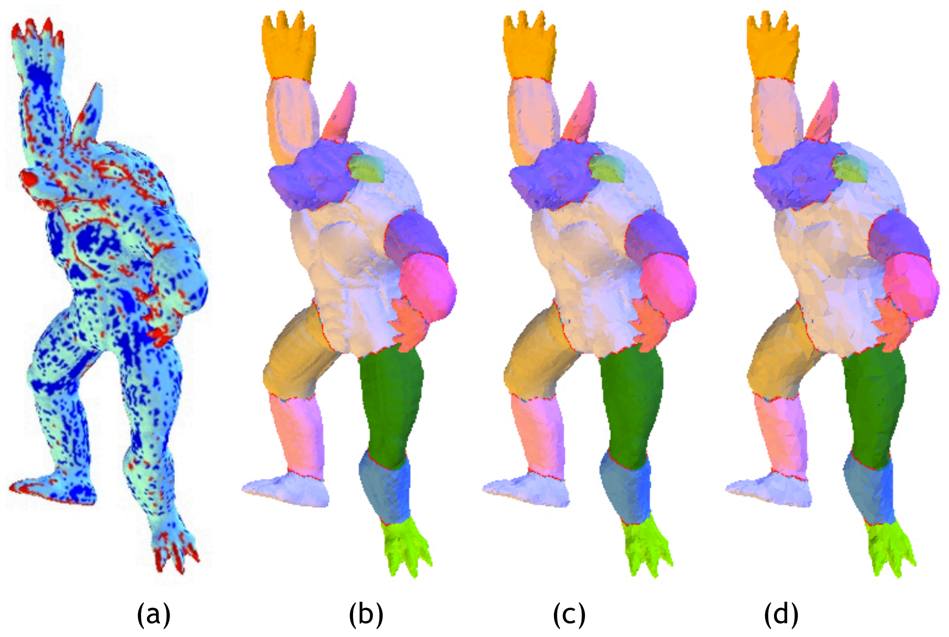


Figure 4.7: Preserving segments, creases and apices: (a) Armadillo's curvature map with high-curvature vertices and edges in red; (b) Armadillo with 50,542 faces after segmentation; (c) Armadillo with 17,440 faces after its simplification in 2 LODs; (d) Armadillo with 9,512 faces after its simplification in 4 LODs.

and $v \in \{x, y, z\}$ is as follows:

$$\sigma_{uv}^2 = \frac{1}{n}\sum_{i=1}^{n}(u_i - \bar{u})(v_i - \bar{v}) \tag{4.2}$$

so that the covariance matrix is given by:

$$C = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \tag{4.3}$$

Consequently, the norm of the covariance matrix, also known as Frobenius matrix, is the following:

$$K = \sqrt{\sum_{\substack{u\in\{x,y,z\}\\ v\in\{x,y,z\}}} \sigma_{uv}^2} \tag{4.4}$$

which represents the combinatorial curvature of a given vertex surrounded by $n$ faces. Note that, in conformity with Eq. (4.4), the value of $K$ is always greater or equal to zero. In Figure 4.7, the high-curvature vertices (including those that are apices or corners) in red are those for which $K \geq 0.5$; those in blue correspond to vertices belonging to nearly planar zones ($K \approx 0$), while those in cyan correspond to vertices in slightly non-planar zones $K < 0.5$. In our xGC-based scheme, high-curvature vertices are not allowed to change at all. Also, an edge with at least a high-curvature vertex bounding it is not allowed to collapse, i.e., they are not eligible for collapsing in the process of simplifying a mesh. These edges bounded by at least a high-curvature vertex are called high-curvature edges.

## 4.4 Multiresolution Operations

In the presence of segmented meshes and meshes with creases and apices, the multiresolution operators must necessarily work in a different manner.

### 4.4.1 Edge collapse for segmented meshes $\odot$

Before proceeding any further, let us recall that the primary simplification rule holds: an edge cannot be collapsed more than once at the same level of detail. For segmented meshes, the secondary simplification rule is as follows: no edge bounding a mesh segment is eligible to collapse. However, these two rules are not enough to preserve the boundaries of mesh segments, i.e., their shapes, because the vertices bounding edges of segment boundaries also bound edges that are eligible for collapsing. These collapse-eligible edges incident at vertices belonging to the boundaries of segments are here called wing edges. Summing up, we have three types of segmentation edges: boundary edges (with disabled collapsing), interior edges (with enabled collapsing), and wing edges (with enabled, but different, collapsing). The two sorts of edge collapse are shown in Figure 4.8: interior edge collapse and wing edge collapse.

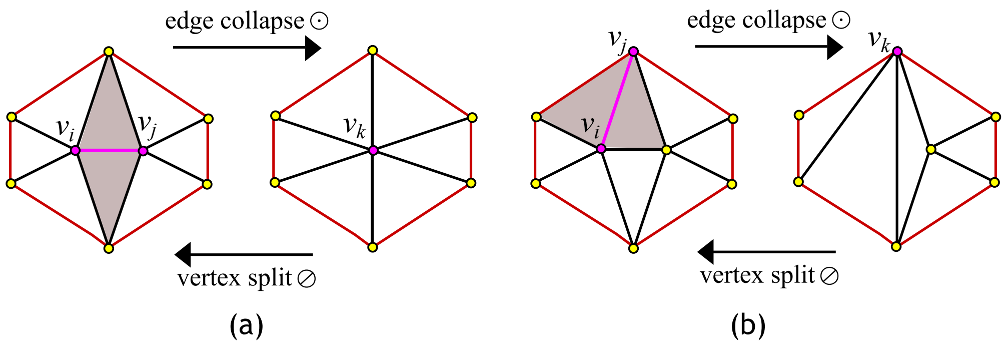


Figure 4.8: Two types of edge collapse (and vertex split).

The *interior edge collapse* (or type-1 edge collapse) is just the standard edge collapse proposed by Hoppe et al. [HDD+93], with the restriction that it solely applies to interior edges of each segment. This topological operator is illustrated in Figure 4.8(a), where the magenta edge bounded by two vertices, $v_i$ and $v_j$, is replaced by a single vertex $v_k$; as a consequence, the two triangles incident on the collapsing edge are also discarded during this operation. In xGC-based multi-resolution scheme, the edge collapse is accomplished according to Hoppe [Hop98], i.e., new vertex $v_k$ is given by the midpoint of the edge $(v_i, v_j)$, so that the Hoppe vector $v_H = v_i - v_k$ allows us to recover the original edge in refinement operations.

The *wing edge collapse* (or type-2 edge collapse) is illustrated in Figure 4.8(b) in magenta. Note that one of the vertices bounding the collapsing edge belongs to the boundary of the mesh segment. In this case, the edge collapses into such vertex belonging to the segment boundary, not into its midpoint; hence, and assuming that $v_j$ is the vertex in the segment boundary, the new vertex is $v_k = v_j$, so that the Hoppe vector is now $v_H = v_i - v_k$.

### 4.4.2 Vertex split for segmented meshes $\oslash$

As illustrated in Figure 4.8, we also have two sorts of vertex split, *interior vertex split* (or type-1 vertex split) and *wing vertex split* (or type-2 vertex split), which are the inverse operations to the interior edge collapse and wing edge collapse, respectively. Each vertex split operation divides the vertex into two new vertices. This means that a new edge and two new triangles end up by making their appearance as a result of either of those two refinement operations.

## 4.5 Segment-Preserving Multiresolution

As for plain multi-resolution meshes, simplification and refinement operations may be applied to a segmented mesh as a whole. In addition, such multiresolution operations can be applied to mesh segments in a separate manner, so that the segments may end up possessing distinct levels of detail. In fact, with the xGC-based scheme, each mesh segment can be simplified and refined independently of any other segment of the mesh. Clearly, this is a contribution in comparison to the scheme proposed by Maglo et. al. [MGH13], which only admits a difference of one level of detail between adjacent segments in the mesh.

### 4.5.1 Segment-Preserving Simplification

The simplification algorithm is based on the edge collapse operators described above. Hereupon, we need to iterate on the edges belonging to the array (`aov`) of visible edges, as defined in the class `MESH`, in order to proceed to the collapse of each eligible edge. Note that a visible edge is an edge that has not been collapsed yet. The eligibility of a visible edge for collapsing is determined according to the following criteria:

- It cannot be a wing edge of a collapsed edge at the current level of detail. This ensures that the mesh is uniformly simplified (cf. [RMSG07] for further details).

- It cannot be an edge belonging to the boundary of a mesh segment. This ensures the mesh segmentation holds.
- It cannot be a high-curvature edge belonging to a crease or being delimited by an apex. This ensures that the mesh holds its creased and pointed shape.

In general, given an eligible edge $(v_i, v_j)$ for collapsing, we proceed as follows:

1. Determine the sort of edge collapse for the edge $(v_i, v_j)$.
2. Determine the new vertex $v_k$ that results from collapsing the edge $(v_i, v_j)$.
3. Calculate the Hoppe vector $\vec{v}_H$.
4. Set the edge collapse type for $v_k$.
5. Set $\vec{v}_H$ as the Hope vector for $v_k$.
6. Create the genealogical bintree for $v_k$ with $v_i$ and $v_j$ as its ancestors.
7. Update de array of visible mesh edges.
8. Update de array of ghost mesh edges.
9. Update the set of edges of $v_k$.

### 4.5.2 Segment-Preserving Refinement

This refinement operation extends the one described by Rodrigues et al. [RMSG07], in the sense that we have now to take into account two sorts of edge collapse, and not a single one, before splitting a vertex that resulted from a edge collapse in the past. That is, before splitting a vertex $v_k$ into a former edge, we need to know which sort of edge collapse has originated it. Recall that all the data (i.e., Hoppe vector and sort of edge collapse) necessary to restore the mesh to a higher level of detail is all maintained in the genealogical tree of each vertex, as well as in the stack of edges associated to each edge. The edge restoration using the vertex split operation is accomplished as follows:

1. *Type-1 vertex split*. This is the inverse of type-1 collapse of the edge $(v_i, v_j)$. The restoration of the edge $(v_i, v_j)$ from the vertex $v_k$ placed at the midpoint of $(v_i, v_j)$ is carried out using the Hoppe vector (hold along $v_k$) as follows: $v_i = v_k - \vec{v}_H$ e $v_j = v_k + \vec{v}_H$.
2. *Type-2 vertex split*. This is the inverse of type-2 collapse of the edge $(v_i, v_j)$. In this case, the restoration of $(v_i, v_j)$ is done from $v_k = v_i$ also with the help of the Hoppe vector as follows: $v_i = v_k$ e $v_j = v_k + \vec{v}_H$.

## 4.6 Results

The xGC-based multi-resolution scheme was designed and implemented on an Intel Core Duo 2.4 computer running a Mac OS X operating system. We also took advantage of the OpenGL User Interface Library (GLUI), which is a C++ user interface library based on the OpenGL Utility Toolkit (GLUT), to build up a graphics interface to render and interactively handle meshes.

Let us mention that we used the mesh models of those 19 families made available by the Princeton Benchmarking [CGF09], but the mesh segmentation of each mesh was generated by the HPO algorithm due to Rodrigues et al. [RMG15]. Testing took place considering the following guidelines: shape preservation across uniformly detailed meshes, shape preservation across non-uniformly detailed meshes, simplification rate, memory space occupancy, and time performance.

### 4.6.1 Preserving shape across uniform levels of detail

As shown in Figs. 4.9 and 4.10, by simplifying a mesh in a uniform manner across successive levels of detail the mesh shape is preserved, as well its segmentation and geometric features (i.e., creases and apices), and this happens in spite of the significant reduction of triangles. This is so because, as mentioned above, the segment boundaries and those geometric features remain geometrically unchanged.

### 4.6.2 Preserving shape across non-uniform levels of detail

Sometimes, we need to simplify the entire mesh with the same level of detail, sometimes it is required to have segments with distinct levels of detail. In the latter case, we need to preserve the boundaries of segments. Unlike other multi-resolution schemes for meshes, the xGC-based scheme is capable of dealing with segments with rather distinct levels of detail. For example, the segmented meshes shown in Figure 4.11 exhibit different levels of detail from a segment to another. Note that segment boundaries, creases, and apices (i.e., corners) are preserved independently of the detail of detail used for each segment. As a consequence, the mesh shape is also preserved.

### 4.6.3 Simplification rate

The simplification rate was evaluated for objects of the 19 classes of the Princeton benchmark, as indicated in Figure 4.12(a). As shown in Figure 4.12(a), the simplification rate (i.e., number of faces) inversely varies with the level of detail. This rate is about 50% for the first level of detail, and from there on it monotonically decreases from one level of detail to another. Nevertheless, the simplification stops after five to ten simplification steps, and this depends on the number of faces of the mesh and its shape.

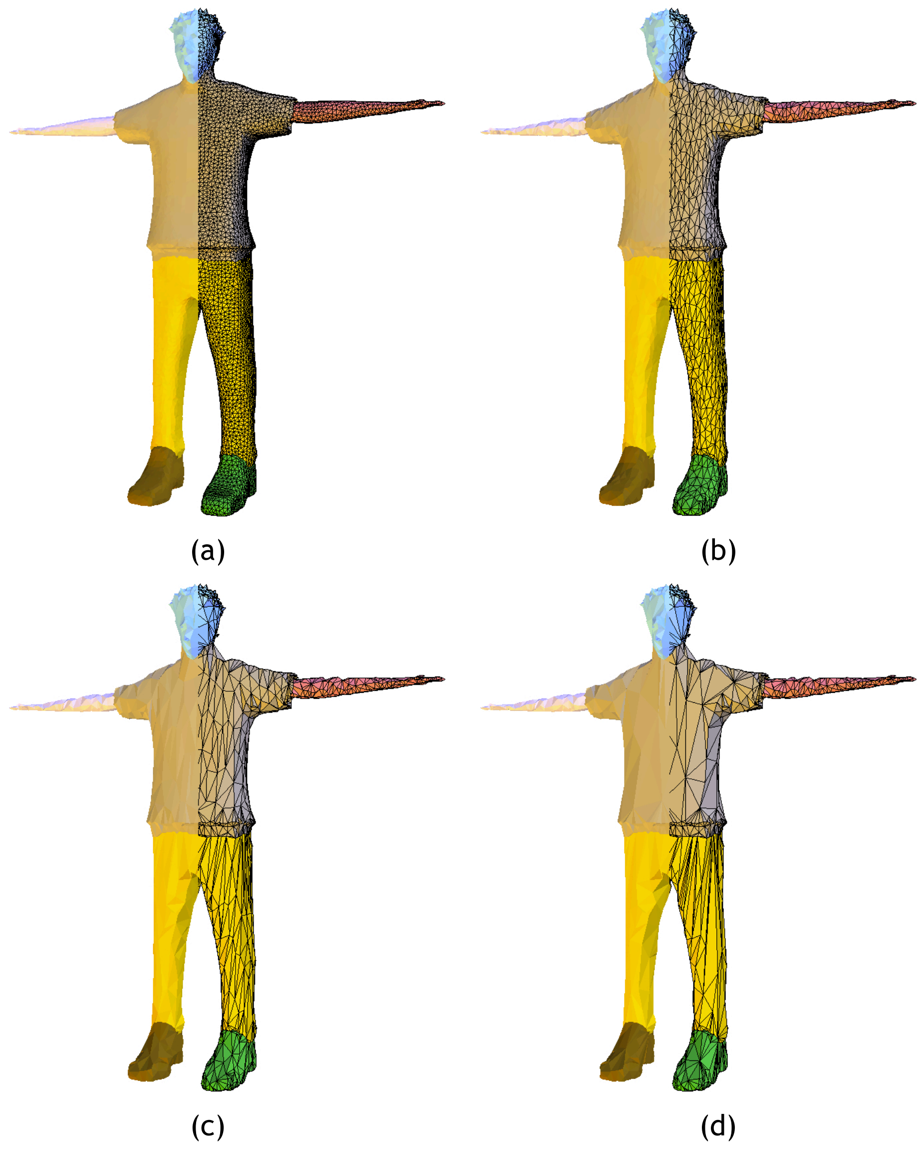


Figure 4.9: A human mesh simplified in a uniform manner after: (a) 0 simplifications (21994 faces, 100%); (b) 2 simplifications (7364 faces, 33%); (c) 4 simplifications (4192 faces, 19%); (d) 6 simplifications (3624 faces, 16%).

It is clear the simplified mesh comes to a point at which it cannot be further simplified due to the geometric criteria imposed to the simplification algorithm. Moreover, we observed that the simplification rate is higher for the meshes with larger segments and small variations of curvature.

### 4.6.4 Memory space occupancy

The memory space occupancy was also measured for objects of the 19 Princeton benchmark classes of objects, as shown in Figure 4.12(b). For that purpose, we measured the memory space occupied by each mesh with 0 to 7 levels of detail, i.e., considering up 7 levels of simplification. As illustrated in Figure 4.12(b), we noted that the average

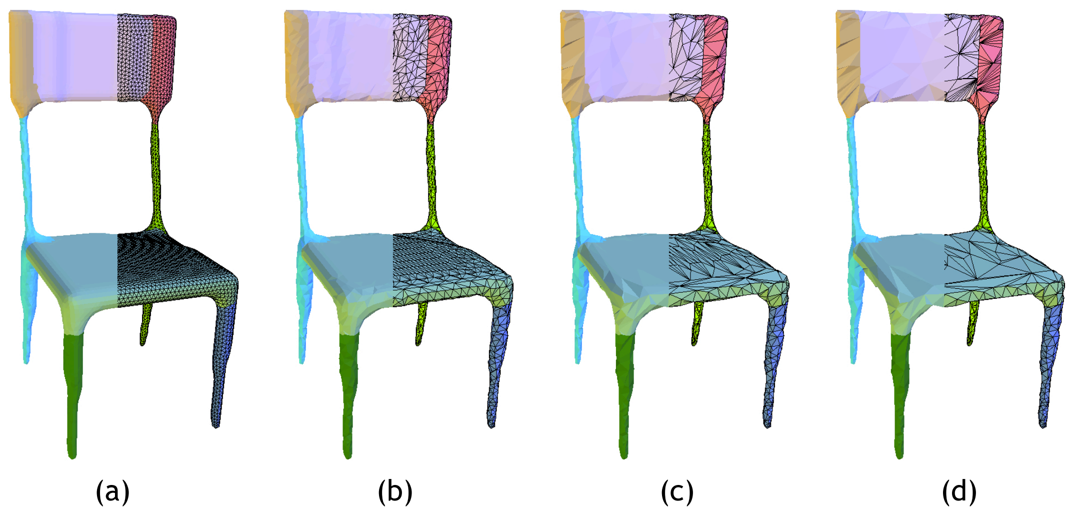


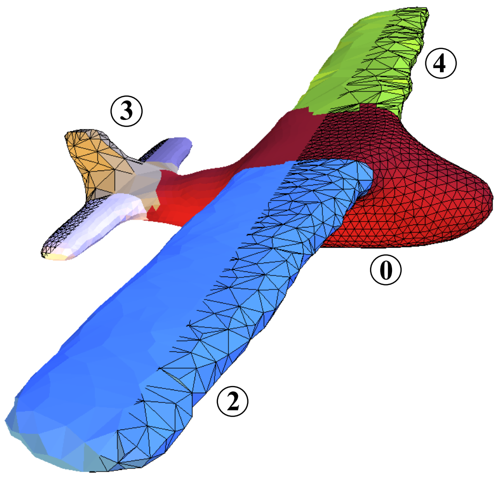
Figure 4.10: A chair mesh simplified in a uniform manner after: (a) 0 simplifications (16998 faces, 100%); (b) 2 simplifications (5846 faces, 34%); (c) 4 simplifications (3674 faces, 22%); (d) 6 simplifications (3238 faces, 19%).

memory utilized by xGC-based mesh across those 7 levels of detail slightly increases from level to level, attaining a maximum of 133% at the seventh level. This increase in occupied memory space is due to the supplementary vertex bintree and the stack of edges of the xGC data structure in Figure 4.3. Nevertheless, this xGC memory space occupancy is much less that those more than 300% of memory space occupied by the 7 traditional LOD meshes of the same mesh.

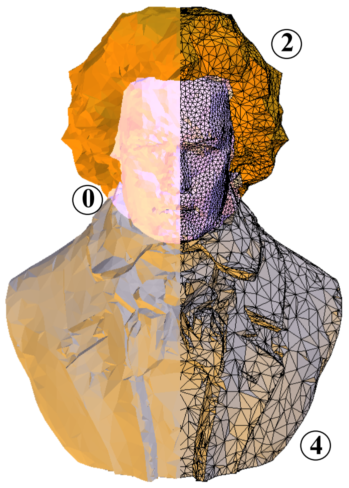
In more formal terms, and taking the Euler formula for 2-dimensional triangular meshes into consideration, the xGC data structure holds $v = n$ vertices, $e = 3n$ edges, and $f = 2n$ faces (triangles). If we assume that 1 float and 1 pointer are 4 bytes long each, 1 integer is 2 bytes long, and a char is 1 byte long, we come to the conclusion that the overall memory space occupancy is $(88 + 2k)f$, where $k$ denotes the vertex degree, i.e. the number of edges incident at each vertex. This memory space occupancy is the result from summing up the following contributions:

- *Vertices*. The storage cost for each vertex amounts to $4$ bytes for its identifier, $1$ byte for the level of detail, $3\times4$ bytes for its three float coordinates, $4\times k$ bytes for its $k$ incident edges, $4$ bytes for a pointer to the array of incident edges, $4$ bytes for its floating-point combinatorial curvature, $1$ char for its type of concavity, and $4$ bytes for a pointer to its genealogical tree (or bintree), i.e. $(30+4k)n = (15+2k)f$ bytes for all vertices.

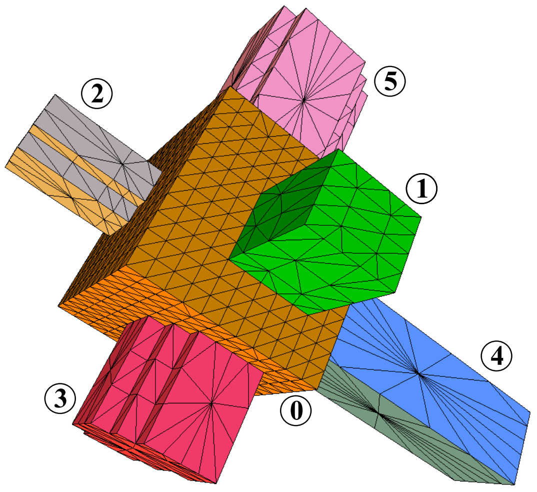
- *Edges*. The storage cost for each edge amounts to $4$ bytes for its identifier, $1$ byte for the level of detail, $2\times4$ bytes for pointers to its bounding vertices, $2\times4$ bytes for its incident faces, $4$ bytes for its floating-point combinatorial curvature, $1$ char for its type of concavity, and $4$ bytes for a pointer to its edge stack, i.e. $30e = 45f$ bytes for all edges.



(a)



(b)



(c)

Figure 4.11: Three meshes simplified in a non-uniform manner: (a) an airplane with segments with 0, 2, 3 and 4 levels of detail; (b) a bust with segments with 0, 2, and 4 levels of detail; (c) a box with segments with 0, 1, 2, 3, 4 and 5 levels of detail.

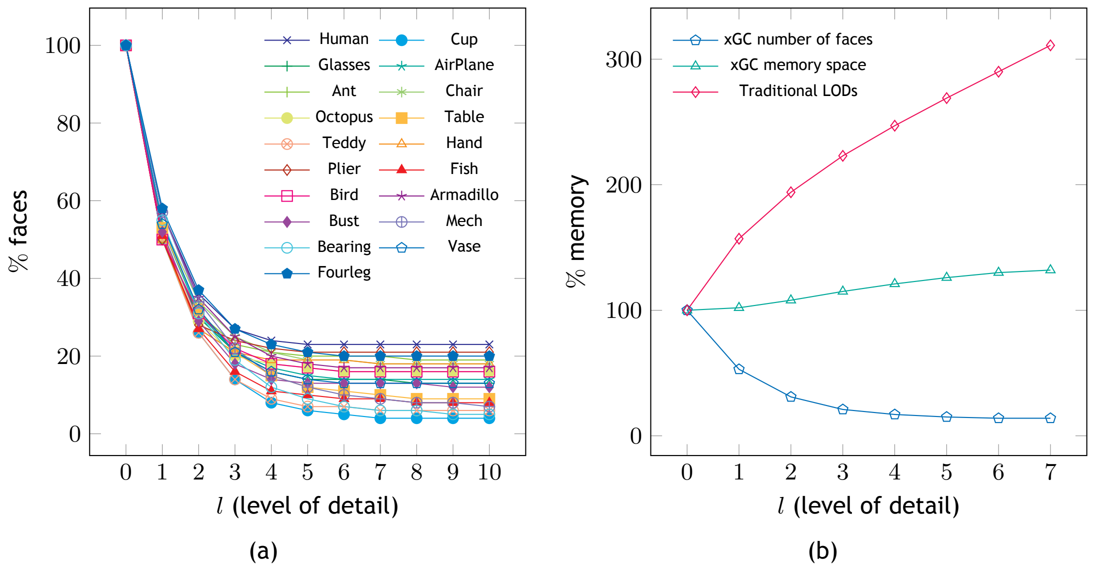


Figure 4.12: Results of Simplification and memory of our algorithm.

- *Faces*. The storage cost for each face amounts to $4$ bytes for its cluster identifier, $3 \times 4$ bytes for its 3 adjacent edges, and $3 \times 4$ bytes for the three floating-point components of its normal vector, i.e. $28f$ bytes for all triangles of the mesh.

However, if the mesh has been simplified, each node of the vertex bintree costs more 24 bytes, but it should be said that each vertex removed from the array of vertices during a simpliifcation step originates only one bintree node. But the cost of adding information to the edge stack amounts to either 11 or 6 bytes for each new stack element, depending on whether the edge is collapsed or not.

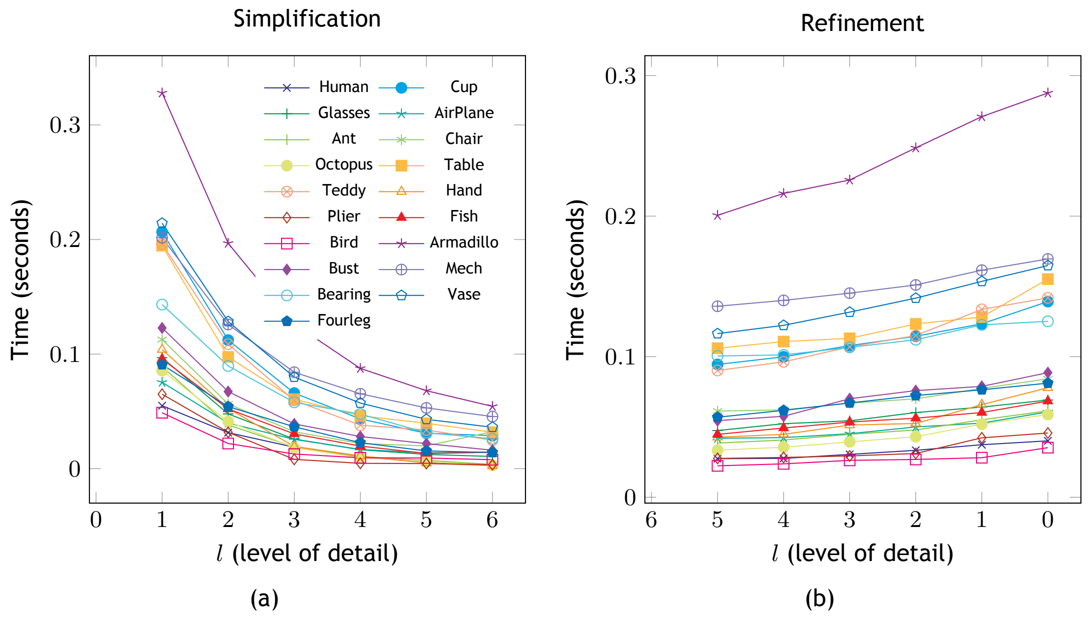


Figure 4.13: Time performance: (a) simplification algorithm; (b) refinement algorithm.

### 4.6.5 Time performance

As shown in Figure 4.13(a), the time performance of the xGC-based simplification algorithm up to 6 simplification levels is quite reasonable even for models possessing more 50,000 triangles; for example, the Armadillo mesh has 50,542 triangles and takes about 0.3 seconds to simplify from the level-0 mesh to level-1 mesh. Conversely, and as illustrated in Figure 4.13(b), the time performance of the xGC-based refinement algorithm is obviously faster than its simplification counterpart because it essentially involves 'undoing' operations, which restores the geometry of the previous level of detail.

## 4.7 Conclusions

We have introduced an integrated multiresolution scheme for segmented meshes that preserves their shape. That is, it preserves the boundaries of mesh segments —and, in turn, the segments themselves—, as well as creases and apices (or corners). The segmentation of a mesh allows for applying simplification and refinement operations to the mesh as a whole or, alternatively, to a segment in a separate manner. Also, segments may possess distinct levels of detail. Summing up, the mesh may possess non-uniform multi-resolution across the segments.

## Related Publications

The work described in this chapter originated a publications about segmented meshes and a Multi-resolution scheme, which is to be submitted for publication as indicated below:

> *Rui S.V. Rodrigues, José F.M. Morgado, and Abel J.P. Gomes. 2016. A Shape-Preserving Multiresolution Segmentation Scheme for Triangle Meshes with Creases and Apices.* Computers & Graphics *(submitted), 2016.*

# Chapter 5

# Conclusions

This thesis sustains on the research work carried out in mesh segmentation and multiresolution meshes, which are two important topics in geometric computing and computer graphics. Briefly speaking, we can say that its main contribution lies in the human perception-oriented mesh segmentation (also called meaningful or perceptive segmentation) technique advanced in the third chapter. But, the thesis also puts forward a solution that combines mesh segmentation and multiresolution meshes in the fourth chapter, which preserves the shape of mesh segments, creases, and apices, i.e., the shape of the a given mesh is preserved in both global and local terms.

## 5.1 Context of the Research Work

It is important to recall that the work developed in this thesis started after getting the understanding that the mesh segmentation algorithms and techniques lacked *generality*, *automation*, and *perceptual proximity*. In fact, we noted that most segmentation algorithms were limited to specific categories of geometric objects; for example, some techniques were more adequate for CAD (computer aided design) mechanical parts, which are usually defined by quadric algebraic geometry, while others were designed for specific families of freeform objects like animals and handicraft. Only recently, we noted a trend to comply with the requirement of generality in mesh segmentation algorithms.

In regard to fully automatic mesh segmentation, we also noted that most algorithms needed some sort of user assistance to set the final number of segments of the mesh. Therefore, we easily concluded that the mesh segmentation techniques in computer graphics were fairly behind the homologous image segmentation techniques in image analysis and processing. Hence, we had investigated automated ways of determining the number of segments beforehand, i.e., before terminating the mesh segmentation procedure.

Another important issue of our work was related with the perceptual proximity of the mesh segmentation. In other words, we were interested in obtaining mesh segmentations close to human-perceptually segmentations as much as possible. Hence, we had used the Princeton benchmark for comparison sake, because it includes human-perceptually segmentations as the ground truth.

## 5.2 Research Questions, Results, and Contributions

At this point, we are able to respond to the research questions put forward in Chapter 1:

**Is it possible to segment a given object into meaningful parts as they are perceived by the human being?**

In Chapter 3, we have introduced the concept of relaxed convexity in order to succeed in obtaining meaningful or perceptive mesh segmentations. The perceptive mesh segmentation is carried out from the region contours or boundaries to their interiors according to the door-in-door-out principle. Besides, perceptive mesh segmentation follows Hoffman's principle so that relaxedly convex regions are built before relaxedly concave regions because humans normally perceives a given object as a set of its convex regions separated from each other each other along concave boundaries.

**Is it possible to segment both freeform and non-freeform objects into meaningful parts through a single segmentation algorithm?**

Likewise, we have also shown that it is possible to design and implement mesh segmentation algorithms since one uses human perception principles. The difficulty was to translate human perception principles into mathematical abstractions and rules. We succeeded at this point because we were able to come up with the concept of relaxed convexity.

**Is it possible to design and implement an algorithm to automatically extract segments without requiring user interaction?**

This is feasible provided that we succeed in finding the number of segments beforehand. Despite the difficulties in computing such number of segments in advance, we did that using histogram-based techniques inspired in image analysis and processing.

**Is it possible to use mesh segmentation within a multiresolution scheme that preserves the segments?**

Preserving mesh segments means that such segments must be preserved no matter the level of detail of the mesh as a whole, as well as the level of detail of its segments separately. We have thus demonstrated that it is possible to integrate the concepts of segmentation and multiresolution seamlessly.

Summing up, and recalling the thesis statement:

> *Is it feasible to segment different categories of meshes in a single, automated manner as the humans perceptually do, regardless of their level of detail?*

we can say the research work described in the present document responds positively to and validates the thesis statement above, in respect to the three open issues mentioned in Chapter 1, namely: perpetual proximity, generality, and automation.

## 5.3 Research Limitations and Future Work

During the research work carried out in the doctoral programme, we have identified a number of research limitations that open a window for future work, namely:

- *Incomplete contours.* Incomplete region contours or boundaries may occur in static and dynamic objects. For example, a plier has a X-shape but may not be divided into four parts because its front and posterior surfaces are planar, i.e., it is not possible to find four complete contours bounding four stem regions. The same applies to L-shapes because there is no complete contour separating the vertical stem from horizontal stem. This problem also occurs in presence of dynamic objects or objects with a number of poses. How to delineate the fingers of an hand when they are leaning against each other?
- *Contour smoothing.* There is still room for improvement the perception of our segmentation algorithm in case we succeed in smoothing the jagged region contours. This would allow us to improve the benchmark results in relation to the CD metric.
- *Number of segments.* We succeeded in calculating the number of segments in an automated manner. However, that required a preliminary mesh segmentation. A possible challenge for the future is to find such number before making any segmentation. In particular, it is certainly necessary to further investigate how this problem has been overcome in the area of image analysis and processing, apart the histogram-based techniques.
- *Small features.* Small features tend to be eliminated because the algorithm prioritizes large regions in detriment of small regions. In fact, small regions usually merge with large regions. This means that small features as those concerning the human eyes end up being absorbed by others neighboring them.
- *Triangle regularity.* After successive mesh simplifications in the multiresolution scheme, triangles tend to get less and less regular. An idea for future work is to introduce a complementary criterion in the multiresolution operators in order to mitigate this issue.

As a final note, let us to say that remains the general impression for future work that a more integrated research work in visual computing, which combines image analysis and synthesis, i.e., image analysis and computer graphics, may originate a catalytic chunk of new knowledge in mesh segmentation and multiresolution in 3D, particularly in human-perceptually segmentations.

# Bibliography

[ABE99] Nina Amenta, Marshall Bern, and David Eppstein. Optimal point placement for mesh smoothing. *Journal of Algorithms*, 30(2):302–322, 1999. 66

[ABK98] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 415–421. ACM, 1998. 38

[ABS01] Marco Attene, Silvia Biasotti, and Michela Spagnuolo. Re-meshing techniques for topological analysis. In *Proceedings of the International Conference on Shape Modeling and Applications*, pages 142–151. IEEE Computer Society, 2001. 39

[AC12] Jacopo Aleotti and Stefano Caselli. A 3D shape segmentation approach for robot grasping by parts. *Robotics and Autonomous Systems*, 60(3):358–366, 2012. 42, 45

[ACK01] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry*, 19(2):127–153, 2001. 39

[AFS06] Marco Attene, Bianca Falcidieno, and Michela Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22:181-193, 2006. 22, 28, 37, 47, 68

[AG03] Eugene L. Algower and Kurt Georg. *Introduction to Numerical Continuation Methods*. Society for Industrial and Applied Mathematics, Philadelphia, USA, 2003. 62

[AGCO13] Shmuel Asafi, Avi Goren, and Daniel Cohen-Or. Weak convex decomposition by lines-of-sight. *Computer Graphics Forum*, 32(5):23–31, 2013. 16, 20

[AHLD07] Grégoire Aujay, Franck Hétroy, Francis Lazarus, and Christine Depraz. Harmonic skeleton for realistic character animation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 151–160. Eurographics Association, 2007. 39

[AKM$^+$06] 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 (SMI'06)*, pages 7–19. IEEE Computer Society, 2006. 53

[AMSF08] Marco Attene, Michela Mortara, Michela Spagnuolo, and Bianca Falcidieno. Hierarchical convex approximation of 3D shapes for fast region selection.

In *Proceedings of the Symposium on Geometry Processing* (SGP'08), Copenhagen, Denmark, pages 1323–1332. Eurographics Association, 2008. 17, 20

[APPS10] Alexander Agathos, Ioannis Pratikakis, Stavros Perantonis, and Nickolas S Sapidis. Protrusion-oriented 3D mesh segmentation. *The Visual Computer*, 26(1):63–81, 2010. 22, 37

[ATC$^+$08] Oscar Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. Skeleton extraction by mesh contraction. *ACM Transactions on Graphics*, 27(3):44, 2008. 39, 43, 44, 45

[AZC$^+$12] Oscar Au, Youyi Zheng, Menglin Chen, Pengfei Xu, and Chiew-Lan Tai. Mesh segmentation with concavity-aware fields. *IEEE Transactions on Visualization and Computer Graphics*, 18(7):1125–1134, 2012. 34, 35, 37, 38, 47, 56, 71

[BAT11] Filippo Bergamasco, Andrea Albarelli, and Andrea Torsello. Semi-supervised segmentation of 3D surfaces using a weighted graph representation. In *Graph-Based Representations in Pattern Recognition*, pages 225–234. Springer-Verlag, 2011. 23

[BAT12] Filippo Bergamasco, Andrea Albarelli, and Andrea Torsello. A graph-based technique for semi-supervised segmentation of 3D surfaces. *Pattern Recognition Letters*, 33(15):2057–2064, 2012. 23, 37

[BB04] David Brunner and Guido Brunnett. Mesh segmentation using the object skeleton graph. In *Proceedings of the 7th IASTED International Conference on Computer Graphics and Imaging,* Kauai, Hawaii, USA, August 16-18, pages 48–55, 2004. 44, 45

[BD92] C. Bajaj and T. Dey. Convex decomposition of polyhedra and robustness. *SIAM Journal on Computing*, 21(2):339–364, 1992. 13, 20, 56

[BDBP09] Stefano Berretti, Alberto Del Bimbo, and Pietro Pala. 3D mesh decomposition using Reeb graphs. *Image and Vision Computing*, 27(10):1540–1554, 2009. 42, 45

[BHP01] Gill Barequet and Sariel Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *Journal of Algorithms*, 38(1):91–109, 2001. 18

[Bie87] Irving Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987. 3, 10, 53

[Blu67] H. Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, 1967. 38, 56

[BLVD11] Halim Benhabiles, Guillaume Lavoué, Jean-Philippe Vandeborre, and Mohamed Daoudi. Learning boundary edges for 3D-mesh segmentation. *Computer Graphics Forum*, 30(8):2170–2182, 2011. 33, 34, 37

[BPVR11] William Benjamin, Andrew Wood Polk, S.V.N. Vishwanathan, and Karthik Ramani. Heat walk: Robust salient segmentation of non-rigid shapes. *Computer Graphics Forum*, 30(7):2097–2106, 2011. 25, 37

[CCMS97] Andrea Ciampalini, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. Multiresolution decimation based on global error. *The visual computer*, 13(5):228–246, 1997. 82

[CDST95] Bernard Chazelle, David P. Dobkin, Nadia Shouraboura, and Ayellet Tal. Strategies for polyhedral surface decomposition: an experimental study. In *Proceedings of the 11th Annual Symposium on Computational Geometry* (SCG'95), Vancouver, British Columbia, Canada, pages 297–305. ACM Press, 1995. 21, 22, 37, 56

[CG06] Lijun Chen and NicolasD. Georganas. An efficient and robust algorithm for 3D mesh segmentation. *Multimedia Tools and Applications*, 29(2):109-125, 2006. 25, 37, 67

[CGC$^+$02] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popovic. Interactive skeleton-driven dynamic deformations. *ACM Transactions on Graphics*, 21(3):586–593, 2002. 38, 56

[CGF09] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics*, 28(3):73:1–73:12, August 2009. 5, 7, 9, 47, 48, 62, 68, 71, 88

[Cha81] Bernard M. Chazelle. Convex decompositions of polyhedra. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing* (STOC'81), Milwaukee, Wisconsin, USA, pages 70–79. ACM Press, 1981. 13, 20, 56

[Cha84] Bernard Chazelle. Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. *SIAM Journal on Computing*, 13:488–507, July 1984. 12

[CK01] C. M. Cyr and B. B. Kimia. 3D object recognition using shape similarity-based aspect graph. In *Proceedings of the 8th IEEE International Conference on Computer Visio*, Vancouver, BC, Canada, pages 254–261. IEEE Press, 2001. 39

[CKGK11] Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. Probabilistic reasoning for assembly-based 3D modeling. *ACM Transactions on Graphics*, 30(4):35:1–35:10, July 2011. 76

[CKLL09] Sungyul Choe, Junho Kim, Haeyoung Lee, and Seungyong Lee. Random accessible mesh compression using mesh chartification. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):160–173, January 2009. 75

[CLJ07] Zhi-Quan Cheng, Hua-Feng Liu, and Shi-Yao Jin. The progressive mesh compression based on meaningful segmentation. *The Visual Computer*, 23(9-11):651–660, 2007. 77

[CP90] Bernard Chazelle and Leonidas Palios. Triangulating a nonconvex polytope. *Discrete & Computational Geometry*, 5(1):505–526, 1990. 13

[CP94] B. Chazelle and L. Palios. Decomposition algorithms in geometry. In C. Bajaj, editor, *Algebraic Geometry and its Applications*, pages 419–447. Springer-Verlag, Berlin, 1994. 13, 56

[CSAD04] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23:905–914, August 2004. 31

[CSM07] Nicu D Cornea, Deborah Silver, and Patrick Min. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization & Computer Graphics*, 13(3):530–548, 2007. 38

[CTO$^+$10] Junjie Cao, Andrea Tagliasacchi, Matt Olson, Hao Zhang, and Zhixun Su. Point cloud skeletons via Laplacian based contraction. In *Shape Modeling International Conference* (SMI'10), Aix-en-Provence, France, June 21-23, pages 187–197. IEEE Press, 2010. 39

[DFM02] Leila De Floriani and Paola Magillo. Multiresolution mesh representation: Models and data structures. In A Ilske, E Quak, and MS Floater, editors, *Tutorials on Multiresolution in Geometric Modelling*, pages 363–417. Springer, 2002. 79, 82

[DFM$^+$05] E. Danovaro, L. Floriani, P. Magillo, E. Puppo, D. Sobrero, and N. Sokolovsky. The half-edge tree: A compact data structure for level-of-detail tetrahedral meshes. In *SMI'05: Proceedings of the International Conference on Shape Modeling and Applications 2005 (SMI'05)*, pages 334–339, Washington, DC, USA, 2005. IEEE Computer Society. 82

[dGGV08] Fernando de Goes, Siome Goldenstein, and Luiz Velho. A hierarchical segmentation of articulated bodies. *Computer Graphics Forum*, 27(5):1349–1356, 2008. 11, 12, 30, 37

[DMSB99] M. Desbrun, M. Meyer, P. Schröder, and A.H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *ACM SIGGRAPH Computer Graphics*, pages 317–324, 1999. 34

[ESV99] J. El-Sana and A. Varshney. Generalized view-dependent simplification. *Computer Graphics Forum*, 18(3):83-94, 1999. 75, 82

[FKS+04] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. Modeling by example. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH'04, pages 652-663, New York, NY, USA, 2004. ACM. 76

[FL+11] Lubin Fan, Kun Liu, et al. Paint mesh cutting. *Computer Graphics Forum*, 30(2):603-612, 2011. 24, 37

[FMA+10] Alfredo Ferreira, Simone Marini, Marco Attene, Manuel J Fonseca, Michela Spagnuolo, Joaquim A Jorge, and Bianca Falcidieno. Thesaurus-based 3D object retrieval with part-in-whole matching. *International Journal of Computer Vision*, 89(2-3):327-347, 2010. 9, 11

[FSKR11] Yi Fang, Mengtian Sun, Minhyong Kim, and Karthik Ramani. Heat-mapping: a robust approach toward perceptually consistent mesh segmentation. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR'11), Colorado Springs, Colorado, USA, June 21-23, pages 2145-2152. IEEE Press, 2011. 26, 27, 37, 38

[GALL13] Mukulika Ghosh, Nancy M. Amato, Yanyan Lu, and Jyh-Ming Lien. Fast approximate convex decomposition using relative concavity. *Computer-Aided Design*, 45(2):494 - 504, 2013. 15, 20, 56

[Gar99] Michael Garland. Multiresolution modeling: Survey & future opportunities. *State of the Art Reports of Eurographics'99, Eurographics Association, Switzerland*, pages 111-131, 1999. 79

[GF08] Aleksey Golovinskiy and Thomas Funkhouser. Randomized cuts for 3D mesh analysis. *ACM Transactions on Graphics*, 27(5):145:1-145:12, December 2008. 32, 33, 37, 47, 68, 76

[GGC+09] Carlos González, Jesús Gumbau, Miguel Chover, Francisco Ramos, and Ricardo Quirós. User-assisted simplification method for triangle meshes preserving boundaries. *Computer-Aided Design*, 41(12):1095-1106, 2009. 76, 77

[GH97] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 209-216. ACM Press/Addison-Wesley Publishing Co., 1997. 75, 82

[GH98] Michael Garland and Paul S Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings of the IEEE Conference on Visualization* (VIS'98), pages 263-269. IEEE Computer Society Press, 1998. 77

[Gra06] Leo Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768-1783, 2006. 26, 34

[GW08] Rafael Gonzalez and Richard Woods. *Digital Image Processing*. Prentice Hall, Inc., New Jersey, USA, 2008. 56, 65

[GWH01] Michael Garland, Andrew Willmott, and Paul S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*(I3D'01), Research Triangle Pk, North Carolina, USA, March 19-21, pages 49–58. ACM Press, 2001. 22, 28, 29, 56

[HC12] Tan-Chi Ho and Jung-Hong Chuang. Volume based mesh segmentation. *Journal of Information Science and Engineering*, 28(4):705–722, July 2012. 31, 37, 56

[HDD$^+$93] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *SIGGRAPH'93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, pages 19–26, New York, USA, 1993. ACM Press. 82, 85

[Her76] Leonard R. Herrmann. Laplacian-isoparametric grid generation scheme. *Journal of the Engineering Mechanics Division*, 102(5):749–756, 1976. 66

[HK92] L. W. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, 1992. 27

[Hop96] Hugues Hoppe. Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (SIGGRAPH'96), New Orleans, LA, USA, August 4-9, pages 99–108. ACM Press, 1996. 43, 82

[Hop97] Hugues Hoppe. View-dependent refinement of progressive meshes. view-dependent refinement of progressive meshes. In *Proceedings of the 24rd Annual Conference on Computer Graphics and Interactive Techniques*, volume 31, pages 189–198. ACM, 1997. 75

[Hop98] H. Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *VIS'98: Proceedings of the Conference on Visualization'98*, pages 35–42, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press. 85

[Hop99] Hugues Hoppe. New quadric metric for simplifiying meshes with appearance attributes. In *Proceedings of the conference on Visualization'99: celebrating ten years*, pages 59-66. IEEE Computer Society Press, 1999. 76, 77

[HR84] D.D. Hoffman and W.A. Richards. Parts of recognition. *Cognition*, 18:65-96, 1984. 10

[HRK08] Kai Huebner, Steffen Ruthotto, and Danica Kragic. Minimum volume bounding box decomposition for shape approximation in robot grasping. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automa-*

*tion* (ICRA'08), Pasadena, CA, USA, May 19-23, pages 1628–1633. IEEE Press, 2008. 18, 20

[HS97] Donald D. Hoffman and Manish Singh. Salience of visual parts. *Cognition*, 63(1):29 - 78, 1997. 2, 3, 10, 15, 22, 49, 53

[HS98] J.E. Hershberger and J.S. Snoeyink. Erased arrangements of lines and convex decompositions of polyhedra. *Computational Geometry*, 9(3):129 - 143, 1998. 13

[HSKK01] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (SIGGRAPH'01), Los Angeles, USA, August 12-17, pages 203–212. ACM Press, 2001. 39, 42

[HSL$^+$06] Jin Huang, Xiaohan Shi, Xinguo Liu, Kun Zhou, Li-Yi Wei, Shang-Hua Teng, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Subspace gradient domain mesh deformation. *ACM Transactions on Graphics*, 25(3):1126–1134, July 2006. 76

[JKP13] B Jüttler, M Kapl, and Qing Pan. Isogeometric segmentation. Part I: Decomposing contractible solids without non-convex edges. Technical Report G+S No. 7, Johannes Kepler University, Linz, Austria, October 2013. 13, 20

[JLCW06] Zhongping Ji, Ligang Liu, Zhonggui Chen, and Guojin Wang. Easy mesh cutting. *Computer Graphics Forum*, 25(3):283–291, 2006. 11, 34, 37

[KCL06] Junho Kim, Sungyul Choe, and Seungyong Lee. Multiresolution random accessible mesh compression. *Computer Graphics Forum*, 25(3):323–331, 2006. 75, 77

[KFK$^+$14] Oliver Van Kaick, Noa Fish, Yanir Kleiman, Shmuel Asafi, and Daniel Cohen-Or. Shape segmentation by approximate convexity analysis. *ACM Transactions on Graphics*, 34(1):4, 2014. 11, 16, 20, 47

[KJS06] V. Kraevoy, D. Julius, and A. Sheffer. Shuffler: Modeling with interchangeable parts. Technical Report TR-2006-09, Department of Computer Science, University of British Columbia, Vancouver, Canada, 2006. 14, 20, 56

[KJS07] Vladislav Kreavoy, Dan Julius, and Alla Sheffer. Model composition from interchangeable components. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (PG'07), Maui, HI, USA, October 20 - November 2, pages 129–138. IEEE Press, 2007. 14

[KK10] Hyoungseok B. Kim and Hosook Kim. Mesh segmentation based on local geometric properties. *International Journal of Computer Science and Network Security*, 10(1), 2010. 56

[KL01] Junho Kim and Seungyong Lee. Truly selective refinement of progressive meshes. In *Graphics Interface*, volume 1, pages 101–110, 2001. 75, 77

[KLT05] Sagi Katz, George Leifman, and Ayellet Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21:649-658, 2005. 22, 37, 47, 68

[KT96] Alan D. Kalvin and Russell H. Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics & Applications*, 16:64–77, May 1996. 21, 56

[KT03] Sagi Katz and Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22:954–961, July 2003. 26, 27, 28, 29, 31, 37, 38, 56, 76

[KYL05] Duck Hoon Kim, Il Dong Yun, and Sang Uk Lee. A new shape decomposition scheme for graph-based representation. *Pattern Recognition*, 38(5):673-689, 2005. 16, 17, 20

[KYL06] Dong Hwan Kim, Il Dong Yun, and Sang Uk Lee. Boundary-trimmed 3D triangular mesh segmentation based on iterative merging strategy. *Pattern Recognition*, 39(5):827–838, 2006. 77

[LA04] Jyh-Ming Lien and Nancy M. Amato. Approximate convex decomposition of polyhedra. In *Posters of the ACM International Conference on Computer Graphics and Interactive Techniques* (SIGGRAPH'04), Los Angeles, CA, USA, August 8-12. ACM Press, 2004. 14, 20, 38, 56

[LA06] Jyh-Ming Lien and Nancy M. Amato. Approximate convex decomposition of polygons. *Computational Geometry: Theory and Applications*, 35:100–123, August 2006. 12, 14, 15, 20, 56

[LCG09] Christian Lovato, Umberto Castellani, and Andrea Giachetti. Automatic segmentation of scanned human body using curve skeleton analysis. In *Computer Vision/Computer Graphics CollaborationTechniques*, volume 5496 of *Lecture Notes in Computer Science*, pages 34–45. Springer-verlag, 2009. 44, 45

[LE97] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. In *SIGGRAPH'97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 199–208, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. 82

[LHMR08] Yu-Kun Lai, Shi-Min Hu, Ralph R. Martin, and Paul L. Rosin. Fast mesh segmentation using random walks. In *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling*, (SPM'08), Stony Brook, New York, USA, June 2-4, pages 183–191. ACM Press, 2008. 26, 37, 47, 68

[LKA06] Jyh-Ming Lien, John Keyser, and Nancy M. Amato. Simultaneous shape decomposition and skeletonization. In *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling* (SPM'06), Cardiff, Wales, United Kingdom, June 6-8, pages 219–228. ACM Press, 2006. 39, 76

[LLL10] Hairong Liu, Wenyu Liu, and L.J. Latecki. Convex shape decomposition. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR'10), San Francisco, CA, USA, June 13-18, pages 97–104. IEEE Press, 2010. 14, 15, 20, 56

[Llo82] S. Lloyd. Least square quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. 21

[LLS+04] Yunjin Lee, Seungyong Lee, Ariel Shamir, Daniel Cohen-Or, and Hans-Peter Seidel. Intelligent mesh scissoring using 3D snakes. In *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications* (PG'04), Seoul, Korea, October 6-8, pages 279–287. IEEE Press, 2004. 32, 38

[LLS+05] Yunjin Lee, Seungyong Lee, Ariel Shamir, Daniel Cohen-Or, and Hans-Peter Seidel. Mesh scissoring with minima rule and part salience. *Computer Aided Geometric Design*, 22(5):444–465, 2005. 32, 37

[LW08] Guillaume Lavoué and Christian Wolf. Markov random fields for improving 3D mesh analysis and segmentation. In *Proceedings of the 1st Eurographics Workshop on 3D Object Retrieval* (3DOR'08), Crete, Greece, April 15, pages 25–32. Eurographics Association, 2008. 23, 37

[LWTH01] Xuetao Li, Tong Wing Woon, Tiow Seng Tan, and Zhiyong Huang. Decomposing polygon meshes for interactive applications. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics* (I3D'01), Research Triangle Pk, North Carolina, USA, March 19-21, pages 35–42. ACM Press, 2001. 11, 38, 43, 45, 76

[LZ04] Rong Liu and Hao Zhang. Segmentation of 3D meshes through spectral clustering. In *Proceedings of the 12th Pacific Conference Computer Graphics and Applications* (PG'04), Seoul, Korea, October 6-8, pages 298–305. IEEE Computer Society, 2004. 26, 37

[LZ07] Rong Liu and Hao Zhang. Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum*, 26(3):385–394, 2007. 28, 37, 56

[LZHM06] Yu-Kun Lai, Qian-Yi Zhou, Shi-Min Hu, and Ralph R. Martin. Feature sensitive mesh segmentation. In *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling* (SPM'06), Cardiff, Wales, United Kingdom, June 6-8, pages 17–25. ACM Press, 2006. 28, 29, 37, 47

[MGH11] Adrien Maglo, Ian Grimstead, and Céline Hudelot. Cluster-based random accessible and progressive lossless compression of colored triangular meshes for interactive visualization. In *Proceedings of the Computer Graphics International* (CGI'11), Ottawa, Ontario, Canada, June 12-15, volume 1, 2011. 11, 75, 77

[MGH13] Adrien Maglo, Ian Grimstead, and Céline Hudelot. Pomar: Compression of progressive oriented meshes accessible randomly. *Computers and Graphics*, 37(6):743 - 752, 2013. Shape Modeling International (SMI) Conference 2013. 75, 77, 86

[MPS$^+$04] M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies. In *Proceedings of the 9th ACM Symposium on Solid Modeling and Applications* (SPM'04), Genova, Italy, June 9-11, pages 339–344. ACM Press, 2004. 30, 39, 40, 45

[MPS06] M. Mortara, G. Patanè, and M. Spagnuolo. From geometric to semantic human body models. *Computers & Graphics*, 30(2):185 - 196, 2006. 40

[MW98] Alan P. Mangan and Ross T. Whitaker. Surface segmentation using morphological watersheds. In *Proceedings of the IEEE Visualization* (VIS'98), Sheraton Imperial Hotel, Research Triangle Park, Durham, USA, October 18-23. IEEE Press, 1998. 24, 37

[MW99] Alan Mangan and Ross Whitaker. Partitioning 3D surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, October 1999. 4, 24, 37, 56, 76, 77, 83

[NB94] Xiujun Ni and M Susan Bloor. Performance evaluation of boundary data structures. *Computer Graphics and Applications, IEEE*, 14(6):66–77, 1994. 79

[NPJ14] Dang-Manh Nguyen, Michael Pauley, and Bert Jüttler. Isogeometric segmentation. Part II: On the segmentability of contractible solids with non-convex edges. *Graphical Models*, 76(5):426–439, 2014. 13

[Paj01] R. Pajarola. Fastmesh: Efficient view-dependent meshing. In *PG'01: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*, page 22, Washington, DC, USA, 2001. IEEE Computer Society. 75

[PKA03] DL Page, AF Koschan, and MA Abidi. Perception-based 3D triangle mesh segmentation using fast marching watersheds. In *Proceedings Intl. Conference on Computer Vision and Pattern Recognition*, volume 2, pages 27–32, 2003. 25, 37

[PRF01] Sandeep Pulla, Anshuman Razdan, and Gerald Farin. Improved curvature estimation for watershed segmentation of 3-dimensional meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 2001. 21

[PSG$^+$06] Joshua Podolak, Philip Shilane, Aleksey Golovinskiy, Szymon Rusinkiewicz, and Thomas Funkhouser. A planar-reflective symmetry transform for 3D shapes. *ACM Transactions on Graphics*, 25:549–559, July 2006. 28, 37, 56

[RB93] J. Rossignac and P. Borrel. Multi-resolution 3D approximations for rendering complex scenes. In B. Falcidieno and T. Kunii, editors, *Modeling in Computer Graphics: Methods and Applications*, Berlin, 1993. Springer-Verlag. 82

[RBG$^+$09] Martin Reuter, Silvia Biasotti, Daniela Giorgi, Giuseppe Patanè, and Michela Spagnuolo. Discrete laplace-beltrami operators for shape analysis and segmentation. *Computers and Graphics*, 33:381–390, June 2009. 56

[Reu10] Martin Reuter. Hierarchical shape segmentation and registration via topological features of laplace-beltrami eigenfunctions. *International Journal of Computer Vision*, 89(2):287–308, 2010. 30, 37

[RGS04] R Raab, C Gotsman, and A Sheffer. Virtual woodwork: Generating bead figures from 3D models. *International Journal on Shape Modeling*, 10(1):1–30, 2004. 44, 45

[RMG15] Rui S.V. Rodrigues, José F.M. Morgado, and Abel J.P. Gomes. A contour-based segmentation algorithm for triangle meshes in 3D space. *Computers & Graphics*, 49(0):24 - 35, 2015. 88

[RMSG07] Rui Rodrigues, José Morgado, Frutuoso Silva, and Abel Gomes. A ghost cell-based data structure for multiresolution meshes. In *Proceedings of the International Conference on Computational Science and Its Applications* (ICCSA'07), Kuala Lumpur, Malaysia, August 26-29, volume 4706 of *Lecture Notes in Computer Science*, pages 666–679. Springer-Verlag, 2007. 79, 86, 87

[RYLL11] Zhou Ren, Junsong Yuan, Chunyuan Li, and Wenyu Liu. Minimum near-convex decomposition for robust shape representation. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 303–310. IEEE, 2011. 15, 20

[SBTZ02] Kaleem Siddiqi, Sylvain Bouix, Allen Tannenbaum, and Steven W. Zucker. Hamilton-Jacobi skeletons. *International Journal of Computer Vision*, 48(3):215–231, 2002. 39

[Ser82] J. Serra. *Image analysis and mathematical morphology*. Academic Press, London, 1982. 21

[SG03] Frutuoso G.M. Silva and Abel J.P. Gomes. Adjacency and incidence framework: a data structure for efficient and fast management of multiresolution meshes. In *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia* (GRAPHITE'03), pages 159–166. ACM Press, 2003. 79

[SG04] Frutuoso G.M. Silva and Abel J.P. Gomes. Normal-based simplification algorithm for meshes. In *Proceedings of the Conference on Theory and Practice of Computer Graphics* (TPCG'04), University of Bournemouth, England, June 8-10, pages 211–218. IEEE Press, 2004. 77, 82

[Sha08] Ariel Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539-1556, September 2008. 50, 53, 55

[SKK01] Y. Shinagawa, T. L. Kunii, and Y. L. Kergosien. Surface coding based on morse theory. *IEEE Computer Graphics and Applications*, 11(5):66–78, 2001. 39, 56

[SLSK07] Andrei Sharf, Thomas Lewiner, Ariel Shamir, and Leif Kobbelt. On-the-fly curve-skeleton computation for 3D shapes. *Computer Graphics Forum*, 26(3):323-328, 2007. 40, 41, 45

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000. 28

[SNKS09] Patricio Simari, Derek Nowrouzezahrai, Evangelos Kalogerakis, and Karan Singh. Multi-objective shape segmentation and labeling. *Computer Graphics Forum*, 28(5):1415–1425, 2009. 18, 19, 20

[SP08] Kaleem Siddiqi and Stephen Pizer. *Medial representations: mathematics, algorithms and applications*, volume 37. Springer Science & Business Media, 2008. 38

[SSCO08] L. Shapira, A. Shamir, and D. Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249–259, 2008. 29, 37, 47, 56, 68

[SSDZ98] Kaleem Siddiqi, Ali Shokoufandeh, Sven J. Dickinson, and Steven W. Zucker. Shock graphs and shape matching. In *Proceedings of the 6th IEEE International Conference on Computer Vision*, pages 222–229, Bombay, India, 1998. IEEE Press. 39, 56

[SSGH01] Pedro V Sander, John Snyder, Steven J Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 409–416. ACM, 2001. 10

[SSH99] M. Singh, G. Seyranian, and D. D. Hoffman. Parsing silhouettes: The short-cut rule. *Percept and Psychophys*, 61(4):636-660, 1999. 15

[STK02] Shymon Shlafman, Ayellet Tal, and Sagi Katz. Metamorphosis of polyhedral surfaces using decomposition. *Computer Graphics Forum*, 21(3):219–228, 2002. 11, 26, 27, 28, 37, 68

[STL06] Idan Shatz, Ayellet Tal, and George Leifman. Paper craft models from meshes. *The Visual Computer*, 22(9):825–834, September 2006. 76

[SvKK$^+$11] Oana Sidi, Oliver van Kaick, Yanir Kleiman, Hao Zhang, and Daniel Cohen-Or. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Transactions on Graphics*, 30(6), 2011. 10

[SWG$^+$03] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (SGP'03), Aachen, Germany, June 23-25, pages 146–155. Eurographics Association, 2003. 11, 56, 76

[SZD$^+$00] Peter Schröder, Denis Zorin, T DeRose, DR Forsey, L Kobbelt, M Lounsbery, and J Peters. Subdivision for modeling and animation. *SIGGRAPH 2000 Course Notes*, 2000. 79

[SZL92] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. In *SIGGRAPH'92: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, pages 65–70, New York, NY, USA, 1992. ACM Press. 82

[TAOZ12] Andrea Tagliasacchi, Ibraheem Alhashim, Matt Olson, and Hao Zhang. Mean curvature skeletons. *Computer Graphics Forum*, 31(5):1735–1744, 2012. 39

[TCL99] Tiow-Seng Tan, Ket-Fah Chong, and Kok-Lim Low. Computing bounding volume hierarchies using model simplification. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 63–69. ACM, 1999. 43

[Til80] R.B. Tilove. Set membership classification: a unified approach to geometric intersection problems. *IEEE Transactions on Computers*, C-29(10):874-883, October 1980. 5, 55

[TNB11] Dilip Mathew Thomas, Vijay Natarajan, and Georges-Pierre Bonneau. Link conditions for simplifying meshes with embedded structures. *Visualization and Computer Graphics, IEEE Transactions on*, 17(7):1007–1019, 2011. 77

[TPT15] Panagiotis Theologou, Ioannis Pratikakis, and Theoharis Theoharis. A comprehensive overview of methodologies and performance evaluation frameworks in 3D mesh segmentation. *Computer Vision and Image Understanding*, 135:49–82, 2015. 50

[TVD06] Julien Tierny, Jean-Philippe Vandeborre, and Mohamed Daoudi. 3D mesh skeleton extraction using topological and geometrical analyses. In *Proceedings of the 14th Pacific Conference on Computer Graphics and Applications*, (PG'2006), Taipei, Taiwan, October 11-13, pages 85-94, 2006. 41

[TVD07] Julien Tierny, Jean-Philippe Vandeborre, and Mohamed Daoudi. Topology driven 3D mesh hierarchical segmentation. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications* (SMI'07), Lyon, France, June 13-15, pages 215–220. IEEE Press, 2007. 12, 41, 42, 45, 46, 56

[VBLT05] Fabien Vivodtzev, Georges-Pierre Bonneau, and Paul Le Texier. Topology-preserving simplification of 2D nonmanifold meshes with embedded structures. *The Visual Computer*, 21(8-10):679–688, 2005. 77

[VF14] Andreas A. Vasilakis and Ioannis Fudos. Pose partitioning for multi-resolution segmentation of arbitrary mesh animations. *Computer Graphics Forum*, 33(2):293–302, 2014. 76

[VL00] Anne Verroust and Francis Lazarus. Extracting skeletal curves from 3D scattered data. *The Visual Computer*, 16(1):15-25, 2000. 43

[VMM99] J. Vollmer, R. Mencl, and H. Müller. Improved laplacian smoothing of noisy surface meshes. *Computer Graphics Forum*, 18(3):131–138, 1999. 66

[WDK01] Ming Wan, Frank Dachille, and Arie Kaufman. Distance-field based skeletons for virtual navigation. In *Proceedings of the IEEE Conference on Visualization* (VIS'01), San Diego, CA, USA, October 21-26, pages 239–246. IEEE Computer Society, 2001. 38

[WLAT14] Hao Wang, Tong Lu, Oscar Kin-Chung Au, and Chiew-Lan Tai. Spectral 3D mesh segmentation with a novel single segmentation field. *Graphical Models*, 76(5):440–456, 2014. 10, 35, 37, 38, 47, 72

[Woo85] T.C. Woo. A combinatorial analysis of boundary data structure schemata. *IEEE Computer Graphics and Applications*, 5(3):19–27, 1985. 79

[WXS06] Naoufel Werghi, Yijun Xiao, and Jan Paul Siebert. A functional-based segmentation of human body scans in arbitrary postures. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(1):153–165, 2006. 41, 45

[XGZ11] Chuhua Xian, Shuming Gao, and Tianming Zhang. An approach to automated decomposition of volumetric mesh. *Computers & Graphics*, 35(3):461–470, 2011. 17, 20

[XLXG11] Dong Xiao, Hongwei Lin, Chuhua Xian, and Shuming Gao. CAD mesh model segmentation by clustering. *Computers & Graphics*, 35(3):685–691, 2011. 24, 37

[XSW03] Yijun Xiao, Paul Siebert, and Naoufel Werghi. A discrete Reeb graph approach for the segmentation of human body scans. In *Proceedings of the 4th International Conference on 3-D Digital Imaging and Modeling* (3DIM'03), Banff, Alberta, Canada, October 6-10, pages 378–385. IEEE Press, 2003. 40, 45

[Xu96] Jianning Xu. Morphological decomposition of 2-D binary shapes into conditionally maximal convex polygons. *Pattern Recognition*, 29(7):1075–1104, 1996. 17

[YHN14] Liyang Yu, Qi Han, and Xiamu Niu. An improved contraction-based method for mesh skeleton extraction. *Multimedia Tools and Applications*, 73(3):1709–1722, 2014. 39

[YKK04] Sheng Yang, Chang-Su Kim, and CC Jay Kuo. A progressive view-dependent technique for interactive 3-d mesh transmission. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(11):1249–1264, 2004. 75

[YWLY12] Dong-Ming Yan, Wenping Wang, Yang Liu, and Zhouwang Yang. Variational mesh segmentation via quadric surface fitting. *Computer-Aided Design*, 44(11):1072–1082, 2012. 31, 37

[ZG06] Steve Zelinka and Michael Garland. Surfacing by numbers. In *Proceedings of Graphics Interface* (GI'06), Quebec, Canada, June 7-9, pages 107–113. Canadian Information Processing Society, 2006. 76

[ZH04] Yinan Zhou and Zhiyong Huang. Decomposing polygon meshes by means of critical points. In *Proceedings of the 10th IEEE International Multimedia Modelling Conference* (MMM'04), Brisbane, Australia, January 5-7, pages 187–195. IEEE Press, 2004. 22, 23, 37

[ZL05] Hao Zhang and Rong Liu. Mesh segmentation via recursive and visually salient spectral cuts. In *Proceedings of Vision, Modeling, and Visualization* (VMV'05), Erlangen, Germany, November 16-18, pages 429–436. AKA, Akademische Verlagsgesellschaft, 2005. 27, 28, 37

[ZLG+15] Huijuan Zhang, Chong Li, Leilei Gao, Sheng Li, and Guoping Wang. Shape segmentation by hierarchical splat clustering. *Computers & Graphics*, 51:136–145, October 2015. 31, 37, 47

[ZLXH08] Xi Zhang, Guiqing Li, Yunhui Xiong, and Fenghua He. 3d mesh segmentation using mean-shifted curvature. In *Advances in Geometric Modeling and Processing*, pages 465–474. Springer, 2008. 23, 24, 37

[ZMT05] Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics*, 24:1–27, 2005. 56, 76

[ZSSL15] Feiqian Zhang, Zhengxing Sun, Mofei Song, and Xufeng Lang. Progressive 3D shape segmentation using online learning. *Computer Aided Design*, 58(4):2–12, January 2015. 76

[ZT98] Y. Zhou and A. Toga. Efficient skeletonization of volumetric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):196–209, 1998. 44

[ZT10] Youyi Zheng and Chiew-Lan Tai. Mesh decomposition with cross-boundary brushes. *Computer Graphics Forum*, 29(2):527–535, 2010. 32, 33, 34, 37

[ZTA12] Youyi Zheng, Chiew-Lan Tai, and Oscar Kin-Chung Au. Dot scissor: a single-click interface for mesh segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1304–1312, 2012. 34

[ZTS02] Emanoil Zuckerberger, Ayellet Tal, and Shymon Shlafman. Polyhedral surface decomposition with applications. *Computers & Graphics*, 26(5):733–743, 2002. 4, 22, 24, 37, 56, 77

[ZvKD10] H. Zhang, O. van Kaick, and R. Dyer. Spectral mesh processing. *Computer Graphics Forum*, 29(6):1865–1894, 2010. 56

[ZZC11] Juyong Zhang, Jianmin Zheng, and Jianfei Cai. Interactive mesh cutting using constrained random walks. *IEEE Transactions on Visualization and Computer Graphics*, 17(3):357–367, 2011. 34, 37

[ZZWC12] Juyong Zhang, Jianmin Zheng, Chunlin Wu, and Jianfei Cai. Variational mesh decomposition. *ACM Transactions on Graphics*, 31(3), 2012. 27, 37, 38, 47