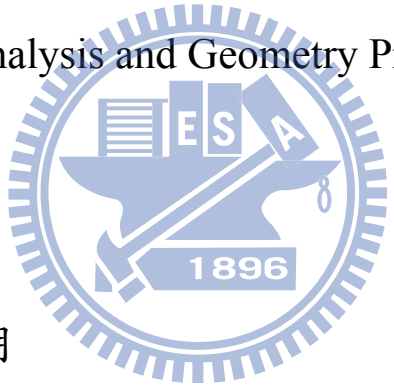
# 國立交通大學

## 資訊工程學系

## 博 士 論 文

以切面為基礎的三維形體分析與幾何處理

Slice-Driven Shape Analysis and Geometry Processing of 3D Models



研　究　生：何丹期

指導教授：莊榮宏　教授

中 華 民 國 一 百 年 七 月

# 以切面為基礎的三維形體分析與幾何處理
# Slice-Driven Shape Analysis and Geometry Processing of 3D Models

研 究 生 ：何丹期　　Student：Tan-Chi Ho

指導教授：莊榮宏　　Advisor：Jung-Hong Chuang

國 立 交 通 大 學

資 訊 工 程 系

博 士 論 文

A Thesis

Submitted to Department of Computer Science

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

July 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年七月

# 以切面為基礎的三維形體分析與幾何處理

學生：何丹期　　指導教授：莊榮宏教授

國立交通大學資訊工程學系博士班

## 摘　　要

中間階層的物體表面函數可以有效的表現物體部位的形體特徵與結構，然而要定義與導出中間階層的表面函數是非常困難的。在本論文中，我們提出了一個新的中間階層物體表面函數，稱為最小切面周長函數(MSP)，來表示物體的區域體積資訊並探討其在幾何處理上的應用。對於一個物體表面上的點，其最小切面周長為是定義在通過該點的切面上，目標是用來表示該點周圍的區域體積。相對於其他中間階層函數如形體對角函數(SDF)[73]，最小切面周長函數可以更正確的反應出物體的區域體積資訊。我們也將此一以切面為基礎的函數應用於網格切割與骨架產生上。所提出之網格切割方法完整的利用到物體區域體積資訊並能夠產生階層式的部位切割將重要部位首先切出並確保每一階層中所切出的部位具有類似的重要度。而所提出之網格骨架產生方式是將三維表面透過一連串的邊交換(edge-swap)操作直接擷取出曲線骨架。所產生之曲線骨架即使在物體的核心區域或骨架接合處都可以保有高密度的節點，此一特性可以幫助我們建立網格表面與骨架之間的對應。此外，所提出之方法只仰賴一個重要度參數來操作，透過調整重要度參數可以得到不同精細度的骨架。最後，我們也探討目前模型簡化機制只參考幾何資訊而未考量觀察者主觀定義所導致主觀認定之重要區域會被過度簡化的問題。為了克服此問題，我們提出了一套使用者可控制之網格簡化機制來讓使用者對網格上的主觀認定重要區域給定權重，並確保該區域在簡化後的模型上具有與給定權重值相似的解析度提升效果。

# Slice-Driven Shape Analysis and Geometry Processing of 3D Models

Student: Tan-Chi Ho

Advisor: Prof. Jung-Hong Chuang

Department of Computer Science
National Chiao Tung University

## ABSTRACT

Intermediate-level surface functions of 3D objects are useful for representing the object's part-level shape information and structure. In this thesis, we propose an intermediate-level surface function and explore its applications to geometry processing. The proposed surface function, called minimum slice perimeter function (MSP), is defined in terms of the slices that pass through the surface point and aims to represent the local volume around the surface point. This slice-based MSP represents more accurate local volume information than previous intermediate-level surface functions, such as Shape Diameter Function (SDF) [73] and ia immediately beneficial to applications such as mesh segmentation and skeletonization. Our proposed mesh segmentation algorithm, which takes advantage of local volume information around the surface point, is able to generate hierarchical segmentation where parts on the same level of the hierarchy share similar salience significance, while parts on a level are less significant than parts on their parental level. The proposed mesh skeletonization scheme employs a greedy edge-swap process that extracts the curve skeleton directly from the 3D surface. The resulting skeleton inherently possesses a dense node distribution at the core part and around the junctions which helps to derive a dense skeleton-surface mapping. Moreover, the single salience parameter for branch removal works well and provides a flexible control for deriving skeleton of varying detail. Finally,

existing level-of-detail modeling techniques consider only geometric other than semantic information, and hence areas of semantic importance are often oversimplified. To ameliorate the problem, we propose a user-controllable mesh simplification framework that allows users to assign weights on selected regions and obtain a predictable improvement of the resolution over the regions.

# ACKNOWLEDGEMENT

I wish to thank my wife, Szu-Wei Lee, for her love and tolerance, without her support this thesis would not have been possible. I would also like to thank my parents for their kindness and encouragement, and gave me fully support during my student life.

I will always be thankful to my advisor, Prof. Jung-Hong Chuang, for his guidance on research. He has led me to the field of compute graphics and supervised me to finish this thesis. I am also grateful for his advisements for the philosophy of life. I am indebted to Prof. Wen-Chieh Lin and Prof. Wingo Sai-Keung Wong for their valuable suggestions on my research. I would like to thank all the labmates over the past decade who fulled my life in NCTU with happiness.

*This thesis is dedicated to my wife, my son, and my family.*

# Contents

# List of Figures

# List of Tables

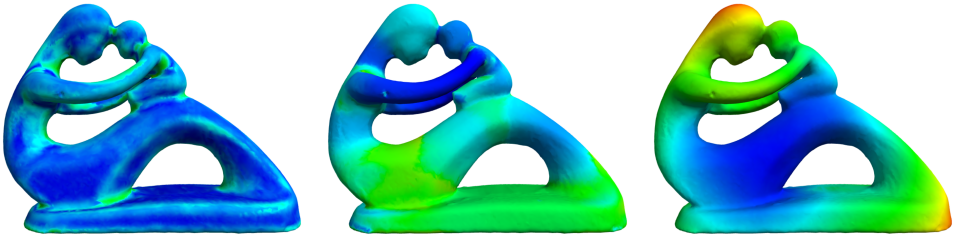
CHAPTER 1

# Introduction

The geometric modeling deals with the mathematical foundation of how the shape of an object is described and manipulated. Among the various object representations, the polygonal mesh, which is capable of representing arbitrary boundary shapes and is efficient for rendering, is most commonly used in the computer graphics field. Many geometric operations are developed for handling polygonal meshes, such as mesh simplification, mesh parameterization, remeshing, and mesh segmentation.

The shape properties (called surface functions) associated with the surface play a crucial role in the manipulation of the polygonal mesh. In the past two decades, various surface functions have been developed for different purposes. Most of them are derived directly on the 2-manifold of the surface. For example, the local geometric features such as curvature [82, 57], and planarity [21] or the global core-salient feature such as averaged geodesic distance [27]. These surface functions, however, are at the two ends of the scaling spectrum. For geometric operations that require the knowledge of part information, these surface functions may be either too local or too global.

Recently, the intermediate-level surface functions have drawn attention in geometric modeling field. Aiming to fill the gap between local and global surface functions, functions defined in intermediate scale can reveal the information about the object's part structures and their significance. Various intermediate-level shape properties have been proposed such as the part salience [41, 19], symmetry [63, 58], visibility inside the object [53], and the shape diameter function (SDF) [73]. In Fig. 1.1, three surface functions, namely mean curvature, our proposed minimum slice perimeter function (MSP), and averaged geodesic



(a) Local (Mean Curvature) (b) Intermediate (MSP) (c) Global (AGD)

Figure 1.1: Surface functions of different scaling on the fertility model.

distance function, that represent surface functions in different scales are visualized on the fertility model with color ranging from blue to red for increasing value. We can see that the minimum slice perimeter function (MSP) describes the part information better than others; the distribution of function values better reveals the surface regions representing the parts. The major difficulty for deriving the intermediate-level surface function is the definition of the affected region for gathering shape information. The local shape properties are usually derived using the theorems in differential geometry by gathering the information in the neighborhood of the surface point. On the contrary, the global shape properties are derived by searching over the entire model. The intermediate-level properties, however, require a reasonable definition about the region for gathering information around the surface point, and may have different region sizes for different surface points. Moreover, unlike the local shape properties, most intermediate-level properties gather shape information in 3D Euclidean space, which leads high computational complexity.

In this thesis, we propose a slice-based method for deriving the intermediate-level surface function. For a surface point $p$, we aim to find a good slice to describe the shape around $p$ so that the perimeter of the slice can be an approximation to the internal volume around the surface point $p$. Our motivation comes from the short-cut rule [79] which states that human vision prefers to use the shortest possible cuts to parse silhouettes. For a surface point $p$, the slice of mesh encompassed by the short-cut passing through $p$ should well represent the object shape around $p$. However, deriving short-cuts on the 3D object is difficult for 3D objects of complex shape and may be not always valid. Instead of finding the exact curved short-cuts on the object, we approximate the short-cut for a surface point $p$ by a planar slice that is the intersection of the mesh and the plane passing through $p$ and perpendicular to the surface at $p$, and has the minimum slice perimeter. We call this planar slice the *minimum perimeter slice* of $p$. Also, we denote *minimum slice perimeter* (MSP)

as the perimeter of the minimum perimeter slice for the surface point.

The minimum perimeter slice has several properties for describing the object shape. First, MSP function can successfully describe the relative internal volume of the local region due to the close relation between the perimeter and the interior area of the slice. Also, the normal of the minimum perimeter slice depicts the orientation of the object part, which might be useful for deriving the vector field on the surface by taking into account the orientation of object parts. Moreover, the fact that a short-cut always crosses a local symmetric axis implies that the intersection point of the minimum perimeter slice and the local symmetric axis can be approximated by the centroid of the slice. Hence the union of those centroid forms a shrunk mesh that approximates the skeleton of the object. Compared to another intermediate-level surface function *Shape Diameter Function* (SDF) [73] that measures the diameter of the object's volume around a surface point, MSP possesses a better measure about the local volume information since SDF reveals only portion of the internal volume for object parts that do not resemble cylinder.

The minimum perimeter slice can be regarded as a good representative planar slice for describing the object shape at the surface point. Geometric modeling applications that require the information about the local shape of object can be benefited. In this thesis, we apply the MSP function to the mesh segmentation and skeletonization processes and show that with the aid of MSP function, the results can have great improvement comparing to previous works.

## 1.1 Volume Based Mesh Segmentation

Most existing mesh segmentation methods employ local geometric properties or global core-salient features. While local shape properties often results in a over-segmented segmentation, the global shape properties may handle only models with core-salience feature.

The volume based surface function can reveal the information about the object's parts. We propose a new part-based hierarchical mesh segmentation scheme that utilizes the local volume information. Based on the observation that surface regions having similar local volume tend to be grouped in the same part, the candidate part boundary regions on the surface can be easily located by applying threshold to the magnitude of MSP gradient. Moreover, the significance of object part pairs separated by the boundary region can be measured using their relative volumes. Based on the significance of object part pairs, for each level of the

hierarchical segmentation, the proposed scheme decomposes the object into several object part pairs that have the most significance. Boundary of the object parts are extracted from the boundary regions using the graph-cut algorithm that takes both the local curvature and the magnitude of MSP gradient into account, resulting in smooth boundaries. Compared to the segmentation methods provided by the segmentation benchmark [10], our segmentation scheme is able to decompose the object into parts in a more visually convinced way for objects of different topological types.

## 1.2 Mesh Skeletonization using Volume Based Surface Function

The skeleton is an important shape descriptor for representing the topology of a 3D object. Most existing mesh skeletonization methods derive the skeleton by either quantizing the 3D object into volume representative or resampling processes. The quantization and resampling of object usually lead to missing topological features; for example, small handles on the original object may be missed in the resulting skeleton. There exists several skeletonization methods that extract the skeleton directly from the 3D mesh. Although the topology of the skeleton can be homeomorphic to the original mesh, such methods usually lead to over-simplified skeleton results.

Based on the proposed slice-based scheme, we introduce a new mesh skeletonization method to extract curved skeleton directly from the surface. Since the short-cut rule [79] implies that a valid short-cut should be across a local symmetry axis, the minimum perimeter slice, as an approximation to the short-cut, inherits the same property. We first approximate the skeleton by the geometric centers of the minimum perimeter slices for all surface points. Then a greedy edge-swap framework is invoked to degenerate the manifold mesh topology to an 1D skeleton. In the process of edge-swapping, path smoothing and branch removal are applied to smooth the skeleton and remove branches due to small surface features and noise. As a result, the derived curved skeleton has a dense skeleton node distribution even at the core or junction parts. Moreover, the skeleton is extracted directly from the original mesh without resampling or quantization of the original mesh. So it is homeomorphic to the original mesh. A skeleton-surface mapping is inherently established that may be useful for embedding the surface information into skeleton. The threshold for

branch removal requires only one parameter and provides a flexible control for deriving skeleton of varying details.

## 1.3 User-Controllable Mesh Simplification

At the end of this thesis, we discuss the issue about a user-controllable strategy for level-of-detail modeling process. The semantic meaning of an object is important in describing the visual significance of some object regions. For example, the eyes are important cues when looking at a human model. However, such a semantic meaning can hardly be described by the existing shape properties. Moreover, different applications may give different semantic meanings about the same object. To introduce the semantic meaning into the modeling simplification process, the user-assisted strategy is often required. The most important aspect in designing a user-assisted interface is the quantization of the semantic meaning and its adaptation to applications. We introduce a user-controllable scheme for the general mesh simplification application, in which the weighting provided by users is specified as the multiple of the resolution improvement in the selected surface regions. The previous weight-based user-assisted simplification methods [38, 64] take the weighting values as the multiplication of simplification cost. As a consequence, the resolution improvement in selected regions are unpredictable. In our user-controllable simplification scheme, the weighting values are used to delay the order of the simplification operators in the simplification sequence. With a carefully designed order delaying scheme, we can achieve predictable resolution improvement for different simplification targets. With our proposed scheme, the same amount of resolution improvement can be obtained for models that are simplified using different simplification metrics.

## 1.4 Contributions

We summarize the contributions of this research as follows :

- Propose a new surface function, called minimum perimeter slice function (MSP), for representing the local volume of an object.
- Propose a new part-based mesh segmentation method that well utilizes the volume based surface function.

  - Regions of similar volume are grouped according to the MSP function.
  - Can construct a hierarchical segmentation on which not only components on a higher level reveal higher degree of salience than their descendant parts but also the components on each level of hierarchy have similar degree of salience significance.

- Propose a novel greedy framework for extracting a curved skeleton directly from the 3D model.
  - The resulting skeleton possesses a dense node distribution at the core parts and around the junctions, and the skeleton-surface mapping and MSP value.
  - A single salience parameter is required for controlling branch removal so as to compute skeletons with varying details.
  - The proposed method is able to generate consistent skeletons for models in different resolutions and poses.
- Propose a user-controllable simplification framework that allows users to obtain a predictable resolution improvement over the simplified mesh deriving by using any existing error metrics.
  - The resolution improvement for a given weighting value in a selected region is predictable.
  - The weighting schemes are completely independent of the error metric used, that is, same resolution improvement for a weighting value is obtained no matter what error metric is used.
  - A weighting value will result in the same resolution improvement when it is applied to simplified meshes in different resolutions.

## 1.5 Thesis Organization

The remainder of the thesis is organized as follow.

In Chapter 2, a survey on the shape properties, mesh segmentation, skeletonization, and user-assisted simplification is given. We introduce a new surface function representing the local volume information around the surface point in Chapter 3. In Chapter 4, a part-based

mesh segmentation algorithm is proposed that can take complete advantage of the volume based surface function. Chapter 5 describes a new mesh skeletonization method that derived from the minimum perimeter slices. In Chapter 6, we propose a user-controllable mesh simplification that allows users to achieve a predictable resolution improvement by using weighting schemes. Finally, we conclude our works and address possible future works in Chapter 7.

CHAPTER 2

# Related Works

In this chapter, we discuss the works related to our research. The review of related work begins by a survey of shape properties used in geometric modeling. We categorize the shape properties according to the spectrum of scaling. Then, we review two geometric modeling applications that benefit from the intermediate-level shape properties, the mesh segmentation and skeletonization. Finally, user-assisted mesh simplification is reviewed.

## 2.1 Shape Property in Geometric Modeling

Most of the geometric operations rely on the analysis of shape properties of the object. Shape property can be classified depending on its coverage on the object. Earlier researches focused on the analysis of local geometric features which mainly derived based on the theories in differential geometry, such as the curvature [82, 57] and planarity [21]. However, the local geometric features are sensitive to the detail signal on the surface. The global shape properties, which are classified at another end of the scaling spectrum, are derived based on the analysis that covers the entire object. The distance between two surface point $p_1$ and $p_2$ can be a metric for describing the relation between these two points. Various distance measurement method have been proposed for different goals. The Euclidean distance is perhaps the easiest way to measure the distance between two points. However, the Euclidean distance can not reveal the change of shape. The geodesic distance [80] is the most widely used distance measurement in shape analysis which measures the shortest length between two surface points along the surface. Additionally, Hilaga et al. used the

average geodesic distance (AGD) to represent the protrusive of the object [27]. The major problem of AGD is its inability to capture the change of shape due to the nature of its global measurement. Moreover, the AGD is not applicable to objects without obvious core-salient features.

To fill the gap between the local and global shape properties, intermediate-scale surface attributes have been proposed for describing the localized regions. The salience is a perception-based measurement describing the visual importance of a region relative to others. On the surface of 3D object, the salience is usually defined as the combination of local shape properties such as the weighted averaging of Gaussian curvatures in different scales by Lee et al. [41] or the combination of curvature and area by Gal and Cohen-Or [19]. However, without the information from more global shape properties, the salience defined based only on the local shape properties may not be able to reveal the significance of the object region. Symmetry of an object is another important characteristic for analyzing the object. Kazhdan et al. defined the symmetry distance of an object using all planes and rotations through its center of mass, and used it as a global shape descriptor for performing shape matching in database retrieval [37]. Podolak et al. extended the work of Kazhdan et al. to the continuous measurement of global reflective symmetry of an object with respect to all planes pass through the surface [63]. Mitra et al. identified the object parts that have partial symmetries within or between them by using the voting process [58]. The reflective symmetry defined by the plane restricts the representative of symmetry for non-rigid shapes. Xu et al. extended the planar reflective symmetry proposed by Podolak et al. [63] to identify the partial intrinsic reflective symmetry of object [87]. They assume that surface points geodesically equidistant to both of the points in the reflective sample point pairs are the candidate position for the intrinsic reflective symmetry. A scalar surface function for describing the intrinsic reflective symmetry can be established by voting the surface points geodiscally equidistant to the sample pairs. Lipman et al. generalized the symmetry correspondences between points by finding orbits between them [50]. By grouping a set of points that have the same orbit in a correspondence graph, problems of finding approximate and partial symmetries can be reduced to the measurement of connectedness in the correspondence graph.

The volume inside the object is another intuitive and useful intermediate-scale shape property. Shapira et al. proposed a surface attribute that describes the local volume as-

sociated with the surface point, called *shape diameter function* (SDF) [73]. The SDF is an approximation of *medial axis transform* (MAT) [11] that uses local shape diameter to approximate the radius of the maximal inscribed ball. Both MAT and SDF have the same problem that the volume information at surface points is quite local. For example, the volume information associated with a point refers only to the maximal inscribed ball associated with that point in the MAT approach. For the slab of a non-cylindrical part, the union of several balls is required for evaluating the entire volume. The SDF has the same limitation since it is conceptually derived from MAT. There are other shape properties defined in intermediate-scale. Liu et al. measured the possibility of surface region to be within the same object part by the visibility of surface region inside the object [53].

## 2.2 Mesh Segmentation

In the past decade, many mesh segmentation methods have been proposed. Based on the objective, mesh segmentation methods generally fall into two categories: patch-type and part-type [5, 71, 10]. Patch-type segmentation usually decomposes the mesh into several patches by analyzing the surface properties such as dihedral angles [46, 75], curvature [56, 60, 75], geodesic distance [88], and planarity [21]. Part-type segmentation tends to segment a complex object into several meaningful components, usually based on the concepts from cognition theory [28, 29]. For example, the minima rule states that human perception tends to break an object into parts along the region of minimum negative curvature [28]. Moreover, the salience of parts determined by relative volume, boundary strength, and degree of protrusion is important for human perception [29]. Our method is a part-type segmentation, and we will mainly review this type of segmentation and refer readers to the excellent survey paper by Shamir [71] for the patch-type segmentation.

Locating the boundary between parts can be done by either boundary-based or region-based approach. Mangan and Whitaker used the watershed to segment the object into several parts according to the curvature on the surface [56]. Lee et al. [45] cut the object into parts by first finding the loop along the minimum negative curvature, and then test the salience of the divided parts based on the part salience theory [29]. However, the surface curvature is too local for describing the shape of object and locating cut boundaries based on the curvature cannot always result in a meaningful part segmentation. Moreover, the iterative segmentation process proposed leads to a binary hierarchical segmentation on

which each level does not provide an intuitive meaning for object parts.

The geodesic distance is another attribute widely used in mesh segmentation [77, 36, 51, 52]. The averaged geodesic distance (AGD) derived as the average of geodesic distance from a surface point to all other points can be used to represent the degree of protrusion of a part. However, such attributes are useful only for models that have an obvious core part and feature parts.

The Medial Axis Transform (MAT) is a global shape descriptor of the object [11]. MAT or skeleton can be used as a guideline for segmentation. For example, Li et al. segmented the object by moving a sweep plane along the skeleton of object [48]. Since the size of the cutting section of the sweeping plane can be regarded as local volumetric information of the object, the cut boundaries are usually at the regions where the size of cutting section varies rapidly. Oscar et al. segmented the skeletal mesh by measuring thickness corresponding to the skeleton nodes derived from the skeletonization process [6]. The most concave region for the cutting is searched for each skeleton branch by comparing the thickness of the skeleton nodes with their neighbors [6]. Reniers and Telea observed that the junction between two skeleton paths has high potential to be a good place for separating two parts [66, 67]. Shapira et al. proposed a hierarchical segmentation method by fitting $k$ Gaussian functions to the histogram of SDF values, and clustering the mesh faces according to their corresponding Gaussian functions [73]. However, the fitting of the global histogram can not reflect the difference between the object parts and some small parts with no salient feature may be segmented. The segmentations generated by using different number of Gaussian functions do not have consistent part correspondence and the part boundaries may not always lie on meaningful regions.

An iterative approach for decomposing the object into several parts is based on the $k$-means clustering [54]. Shlafman et al. used $k$-means clustering to segment the object into a user-specified number of components [77]. This work was later refined to achieve hierarchical segmentation [36]. However, the geodesic distance used in [36] describes only the protrusion of object parts and hence the resultant hierarchical segmentation tends to cut the object along the longest parts at the higher level of the hierarchy, which may not meet the concept of perception. Moreover, the method is usually suitable only for objects having obvious core-salient features. The challenge to the $k$-means clustering methods is that the value of $k$ needs to be given a priori. Liu and Zhang overcame such problems by applying

the spectral analysis on the affinity matrix constructed using the mesh faces [51].

Another segmentation approach was based on the fitting of primitives. Attene et al. extended the hierarchical face clustering [21] and replaced the clustering metric by other similarity measures for the predefined primitives, such as spheres, cylinders and planes [4]. Mortara et al. detected the parts with tubular shape from the whole object [59]. They extracted the core part by excluding all the tubular parts from the object to complete the segmentation.

Pose-invariant mesh segmentation has attracted more attention in recent years. Such works focused on finding the consistent segmentation over different poses of a model. Katz et al. transformed the original model into a pose-invariant representation using multi-dimensional scaling and then used spherical mirroring to extract the core of the object and feature points to segment the objects [35]. In character animation, the pose-invariant segmentation can be achieved by finding the rigid components during the animation [41, 44, 43]. However, such methods are usually suitable only for articulated models and the requirement of animation sequences also poses some restrictions on the usability.

Consistent mesh segmentation aims to produce consistent segmentations for a set of meshes [78, 72, 84]. Golovinskiy and Funkhouser [23] employed rigid alignment [9] in a hierarchical clustering approach for consistent segmentation. Both the geometric features of individual meshes and the correspondence information between the set of meshes are considered. However, rigid alignment may not be able to correctly align the meshes in some cases, as reported in [34]. Kalogerakis et al. [34] proposed a scheme to compute segmentations and to assign labels for a set of meshes. The assignment of labels was formulated as an optimization problem and the objective function measured the consistency of primitives (i.e. triangles) with labels. The objective function was computed via a training process and then applied to the other meshes for computing consistent segmentation. Their approach handled various segmentations for a wide range of meshes. However, the training process is time consuming and usually takes hours of computing.

Golovinskiy and Funkhouser [22] defined a surface function called partition function that indicates how likely each edge is to lie on the boundary of a random segmentation drawn from a set of existing segmentations. Based on the partition function, a cut is associated with a consistency measure as the length-weighted average of the partition function values of its edges. The most consistent cuts defined as the set of cuts with highest con-

sistency are used for finding the part boundaries. Chen et al. proposed a benchmark for quantitative evaluation of mesh segmentation algorithms [10]. The benchmark includes a data set with 4,300 manually generated segmentations for 380 object meshes in 19 categories. It also provides software for producing four quantitative metrics for the comparison of segmentation algorithms.

## 2.3 Mesh Skeletonization

Most of the mesh skeletonization methods can be classified into two categories: volumetric and geometric. The volumetric methods perform thinning on the voxelization structure of the objects [42, 61, 55] or tracking on the field function derived from the objects [12, 26]. The review of volumetric methods is referred to the survey by Cornea and Min [15].

A common geometric approach is based on the construction of the *Voronoi diagram* [85] and its extension [2, 18, 17]. The Reeb graph [65] can also be used for computing the skeleton from the 3D model. There are a variety of techniques using different surface functions, such as the geodesic function [27, 83], harmonic function [8], and height function in 3D space [76, 3]. One challenge of this approach is to embed the skeleton into the geometry. In most cases, the skeleton is derived by averaging the position of vertices on the slab defined by the iso-contour of the function [62, 83]. Aujay et al. embedded the skeleton through an adaptively refinement by solving the Laplacian equation [8].

Sharf et al. constructed the skeleton of the 3D model by tracing the growing fronts of the blob inside the model [74]. Their method generates smoothed curved skeleton with topology homotopic to the deformable model. However, their work requires a deformable model for driving the process and a filtering process for removing the noisy branches.

There are skeletonization methods for extracting the skeleton directly from the surface of the model. Au et al. extracted the skeleton by iteratively contracting the mesh based on Laplacian smoothing [6]. The resulting skeleton has coarse node distribution around the junctions and within the core part. Moreover, the Laplacian smoothing requires several boundary constraints for ensuring that the skeleton branches are preserved. There is no intuitive interpretation in doing so. The *shape diameter function* (SDF) [73] can also be used to extract the skeleton. The candidate position of the skeleton is achieved by transforming mesh vertices using the half distance toward the inverse normal direction. The skeleton is then obtained using the moving least square reconstruction. Such skeletonization scheme

is not homotopic since it does not take the connectivity of the mesh vertices into account. Other than the skeletonization of polygonal meshes, Tagliasacchi et al. extracted the skeleton from the point cloud by finding the rotational symmetry axis [81]. Their scheme can compute the skeleton even though the point cloud data is incomplete.

## 2.4 User-Assisted Mesh Simplification

Level-of-detail (LOD) modeling aims to represent a complex mesh with several levels of detail, and from which an appropriate level is selected at run time to represent the original mesh. A number of methods have been proposed in the literature. Most methods simplify the given mesh by using a sequence of primitive collapsing operations, such as edge collapse [32], triangle collapse [25], vertex clustering [68], vertex removal [70], and multi-triangulations [16].

The primitive collapsing operations can be organized in various orders. The simplest way is to perform the operations in arbitrary order. A more sophisticate approach is to perform the operations in the increasing order of collapsing cost, which is analogous to the greedy algorithm. Several error metrics have been proposed to determine the cost of an edge collapsing operation, such as quadric error metrics (QEM) [20], appearance-preserving simplification (APS) [14], image-driven simplification (IDS) [49], and perceptually guided simplification of lit, textured meshes [86]. Each error metric has its own strength and weakness in preserving certain properties of the original mesh. For example, quadric error metrics [20] tends to preserve only the geometric accuracy during the simplification process, appearance-preserving simplification (APS) [14] takes the texture deviation into account, and image-driven simplification [49] aims to preserve the visual fidelity between the simplified mesh and the original mesh. Moreover, These metrics fail to consider semantic or functional features on the models. As a result, it is found in practice that these metrics alone are not able to produce satisfactory results when very low polygon count is the goal.

The first system that allows users to guide the simplification is Zeta proposed by Cignoni et al. [13]. Zeta takes a pre-computed sequence of primitive simplifications as an input, and utilizes hyper-triangulation model, which employs vertex decimation as the local mesh reduction operator. Users can selectively refine a model by locally changing error thresholds to extract different approximations that did not appear during the original simplification

process. *Semisimp* proposed by Li and Watson [47] provides three approaches for users to manipulate the simplification results using the simplification hierarchy. It allows users to improve mesh quality by manipulating the simplification orders, vertex positions, and the hierarchical partitioning of mesh during the simplification.

Kho and Garland [38] proposed a user-guided mesh simplification system particularly for meshes derived using QEM [20]. To increase resolution in a selected region, the system multiply quadric errors associated with vertices in the region by the weighting multiplier, and hence postpone the edge collapse operations in the region. The *constraint quadrics* can be augmented into optimal placement computation to bias the optimal position towards the constrained planes. Pojar et al. presented an approach that is very similar to the work of Kho and Garland [64]. A sophisticated Maya plug-in is provided to offer rich interface and great compatibility with other modeling applications. Since the distribution of QEM during simplification can not be described by a simple function, the weighting approach proposed in [64, 64] suffers from the problem that the value of multiplier has no direct relation to the increase in resolution. In consequence, the value of multiplier is chosen in a trial and error basis.

Hussain et al. [33] proposed a unified framework for constructing multiresolution mesh based on the simplification hierarchy and hypertriangulation model [13], called *adaptive simplification model (ADSIMP)*. It provides the ability of real time navigation across continuous LODs of meshes. Two operations, *selective refinement* and *selective simplification*, are provided to fine tune the simplified mesh at any level of detail.

A user-assisted simplification method for converting CAD models into the triangle meshes with boundary preservation was proposed by González et al. [24]. Their method allows users to specify different levels of detail for each subobject of the CAD models, and ensures the consistency of boundaries between subobjects. However, the requirement of subobjects limits their work to be useful only for man made CAD models.

CHAPTER 3

# Minimum Slice Perimeter Function

Our research focus on the derivation of surface properties that benefit the intermediate-level shape analysis, such as the object part identification. The major challenge of deriving the intermediate-level shape properties for 3D object is the high computational cost and how to define the affected range.

The short-cut rule [79] states that human vision prefers to use the shortest possible cuts to parse silhouettes. Moreover, the area bounded by the short cut is an appropriate measure for the local interior volume of the object around the surface point. Hence the short cut is an important property for segmenting parts. Deriving the short cuts on a 3D model, however, is difficult due to the complex relation between the 3D shapes and silhouettes. Instead, we approximate the short cut for a surface point by using the planar slice that passes across the model through the surface point and has the minimum perimeter.

## 3.1 Definition of Minimum Slice Perimeter Function

For a manifold surface $M$, we define the *Minimum Slice Perimeter* (MSP) function at a surface point $p$ as the minimum perimeter of the planar slices passing through $p$ and parallel to the surface normal at $p$. It is given as follows:

$$\mathbf{MSP}(p) = \min_{n} \|pl(n,p) \cap M\|, \tag{3.1}$$

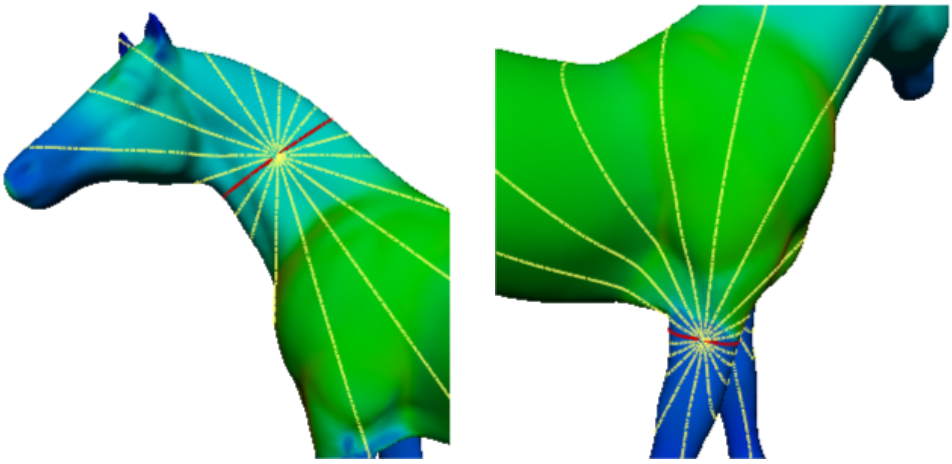


Figure 3.1: Slices for computing MSP.

where $pl(n,p)$ represents a plane passing $p$ with unit normal $n$ that is perpendicular to the surface normal at $p$ and $\|pl(n,p) \cap M\|$ represents the perimeter of the intersecting slice of $pl(n,p)$ and $M$. The planar slice having the minimum perimeter is called the *Minimum Perimeter Slice* of $p$.

For each surface point $p$ of the 3D mesh, the MSP value of $p$ is derived by first computing the intersecting slices for a predefined number of slice planes and then compute the minimum perimeter from the intersecting slices. The slice planes are distributed uniformly about the surface normal vector at $p$. The intersecting slice of a slicing plane and the surface is computed by performing a surface propagating process starting from the face containing $p$. A slice's perimeter is computed as the total length of the edges on the intersecting slice. The minimum perimeter is taken as the MSP value of $p$. Fig. 3.1 shows 10 intersecting slices and the slice with the minimum perimeter is shown in red.

Fig. 3.2 demonstrates the MSP function on some models. The color ranging from blue to red indicates MSP values from low to high. It is observed that the MSP function is effective for representing the local volume information. The core parts have higher MSP value than the salient parts for the articulated models and the parts can be distinguished easily according to the MSP distribution even for models with complex topological structures. In Fig. 3.3, the normal of the minimum perimeter slices is visualized on the surface. Such normals are able to reveal the overall orientation of the object shape. Since the minimum perimeter slice describes the object shape in intermediate scale, it's independent on the pose of object. As shown in Fig. 3.4, a cat model with different poses has similar MSP distributions. Similarly, a model with different resolutions also has similar MSP distributions,

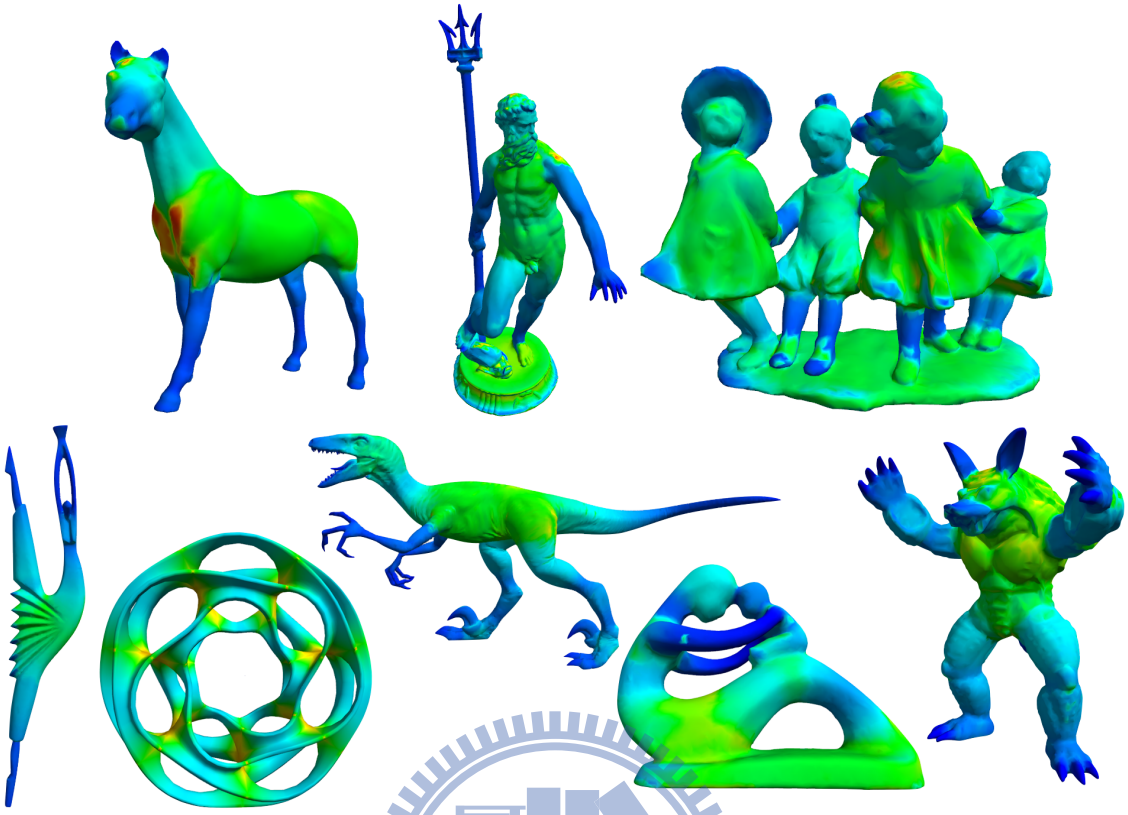


Figure 3.2: The MSP function for different 3D models.

as shown in Fig. 3.5.

Table 3.1 shows the timing data for computing MSP value using eight slices for each surface point on the models. The test platform is a PC with Intel Core i5 2.67Ghz CPU. Notice that we did not implement any acceleration structures for our MSP method. It turns out that the computation time of our method is similar to that of SDF [73]. The accuracy of

Table 3.1: Timing data for computing MSP values.

| Model | Number of polygons | Time (sec.) |
|---|---|---|
| Raptor | 30k | 2.77 |
| Heptoroid | 20k | 1.01 |
| Dancer | 20k | 2.07 |
| Armadillo | 20k | 2.93 |
| Neptune | 60k | 14.19 |
| Dancing Children | 30k | 4.93 |
| Fertility | 30k | 6.02 |
| Horse | 40k | 9.07 |

the MSP value for a surface point depends on the number of slices used. In our experiment, some noticeable noises can be observed for the number of slices less than four. For the

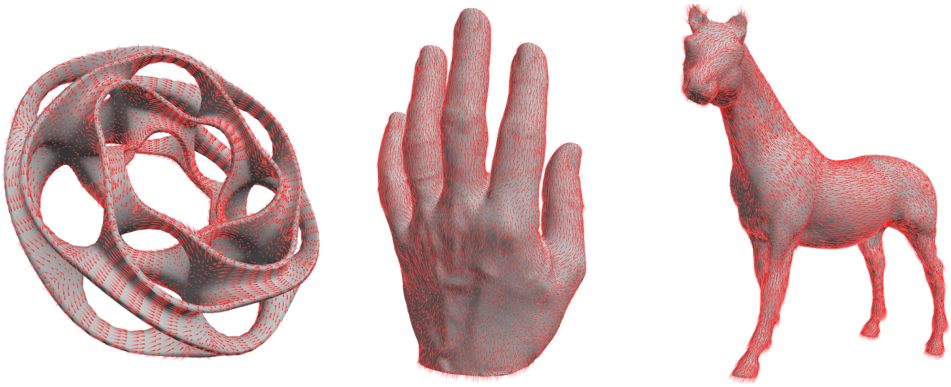


Figure 3.3: The normal of minimum perimeter slices that is projected onto the surface.

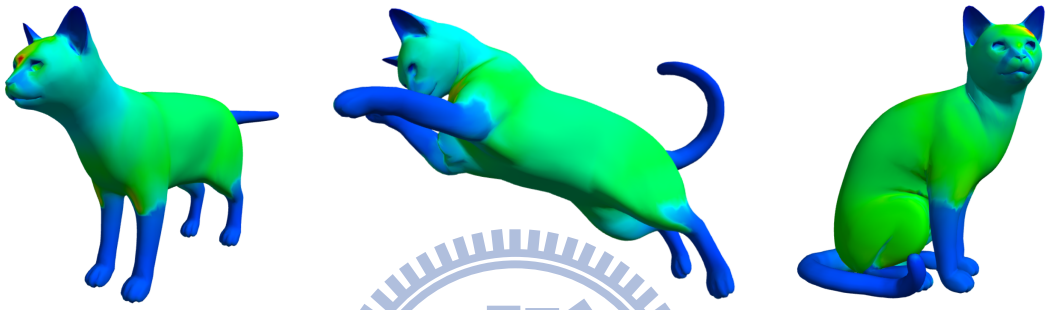


Figure 3.4: MSP function for the cat model in different poses.

number of slices higher than or equal to eight, the MSP values are almost the same. A typical range of the number of slices is from four to eight for trading off between quality and efficiency.

Some highlights of MSP value can be observed at the tips of object parts (such as the chest of horse, shoulder of nepture, and head of dancing children) in Fig. 3.2. This is due to the improper orientation of minimum perimeter slices, where the slice planes in such tip regions are almost parallel to the local symmetric axis. Moreover, the slicing along the normal direction may be sensitive to the surface noise. Fig. 3.6 illustrates two examples for the MSP function on noisy surfaces, where the raptor model with high resolution has details on its surface and the horse model is produced by applying a turbulence function on the model surface. Non-smooth MSP value can be observed, particularly at small regions where MSP values change dramatically.

## 3.2 Minimum Perimeter Slice Refinement

A good minimum perimeter slice should be capable of representing the shape in the slab of its local region well. Therefore, a good minimum perimeter slice can be considered as

Figure 3.5: MSP function for the raptor model in different resolutions.

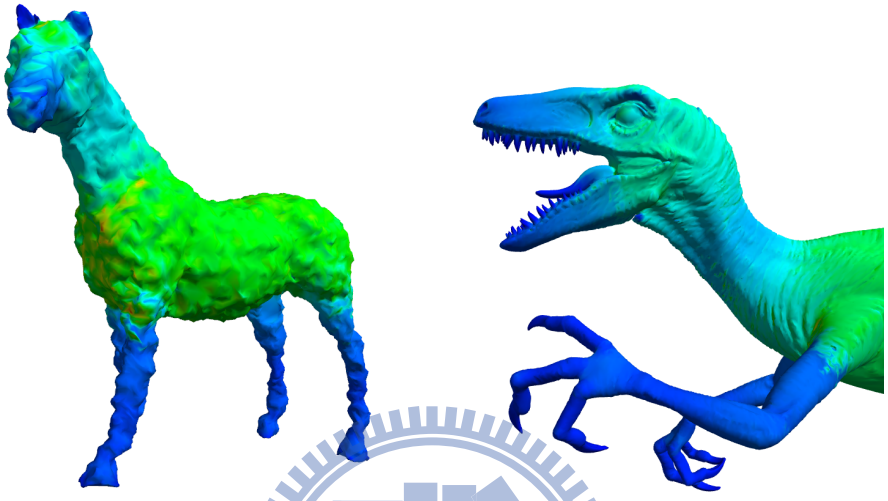


Figure 3.6: The MSP function of noisy surfaces.

the slice on which all the surface points along the slice yield similar minimum perimeter. Based on this observation, to measure how well a minimum perimeter slice is, we define the slice-deviation error for the minimum perimeter slice of a surface point $p$ as follows:

$$c_{dev}(p) = \frac{1}{\|S(p)\|} \int_{q \in S(p)} \cos^{-1}(n_p \cdot n_q)\, dq, \tag{3.2}$$

where $S(p)$ is the minimum perimeter slice at the surface point $p$, $n_p$ and $n_q$ are the unit plane normals of the minimum perimeter slices at $p$ and $q$, respectively. Fig. 3.7 depicts the slice-deviation error of the models shown in the top row of Fig. 3.2. High slice-deviation error can be observed at the tips of object parts and the junctions, where improper orientation may be used to derive the minimum perimeter slices.

According to the definition in [79], the short-cut is not necessary to be perpendicular to the surface normal, slicing along the surface normal may be too restrict for deriving a good minimum perimeter slice for approximating the short cut. It is reasonable to argue that an ideal minimum perimeter slice at the surface point $p$ should be minimal for both the slice perimeter and the slice deviation error. However, such a minimum perimeter slice may not always exist. Instead, we refine the orientation of the minimum perimeter slice at

Figure 3.7: The slice deviation error of 3D models.

the surface point $p$ by minimizing the objective function described in Eq. 3.3.

$$O_p(n) = \frac{\|pl(n,p) \cap M\|}{2\pi r_M} + \frac{c_{dev}(p)}{\pi/2}, \tag{3.3}$$

where the plane normal $n$ is not restricted to be perpendicular to the surface normal at point $p$, and $r_M$ denotes the half of the diagonal of the object's bounding box, which is used to normalize the slice perimeter.

Refining the minimum perimeter slice using Eq. 3.3 is, however, difficult due to the intercorrelation between slice normal and slice deviation error. Instead, we use an iteration-based method that takes the minimum perimeter slice computed using method described by Eq. 3.1 as an initial guess. For each iteration, the new slice that minimizes the combination of the slice perimeter and the slice deviation error is taken. The searching space for the new slice plane should cover the entire hemisphere centered at $p$, which is too large. For the minimum perimeter slice $S(p)$ of the surface point $p$, we consider the unit plane normal $n_q$ of the minimum perimeter slice of each point $q$ along $S(p)$. The average of those normal $n_q$ weighted by using the reciprocal of slice-deviation error at $q$ gives a reasonable orientation of the object part, and hence offers a candidate slice-plane normal for computing a better minimum perimeter slice; as shown in Eq. 3.4.

$$n_{avg}(S(p), p) = \int_{q \in S(p)} \frac{n_q}{c_{dev}(q)} dq, \tag{3.4}$$

Hence, the search of the slice-plane normal in the entire hemisphere space is reduced to the search in the range bounded by the normal of the minimum perimeter slice plane derived in previous iteration and the candidate slice-plane normal $n_{avg}$. The iterative process is

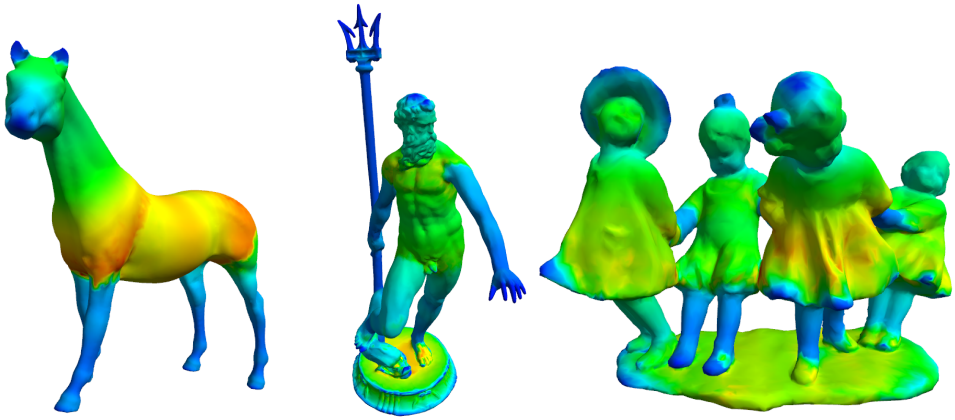


Figure 3.8: The refined MSP function of the 3D models.

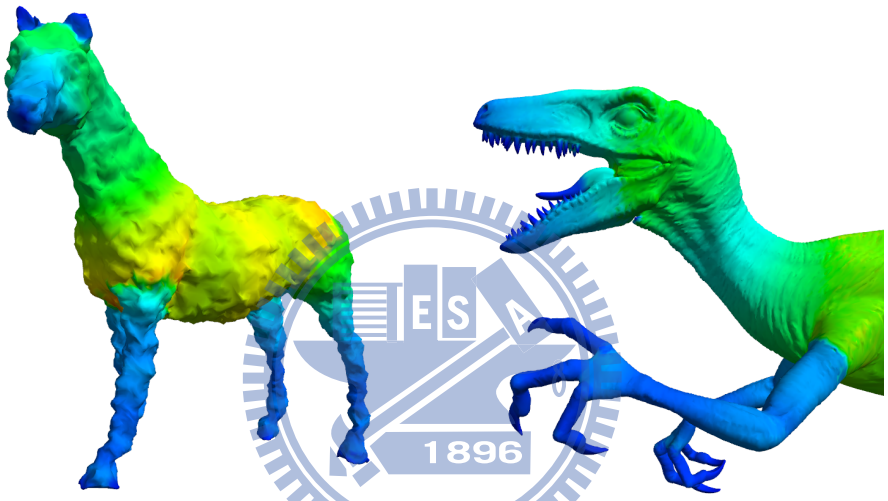


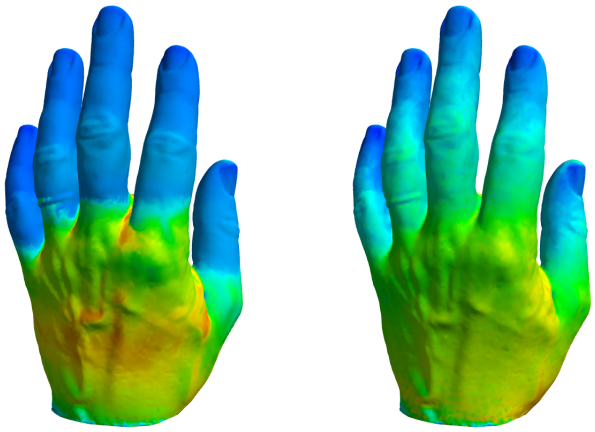
Figure 3.9: The refined MSP function of noisy surfaces.

performed until the normal difference between the new minimum perimeter slice plane and the previous one is below a user-specified threshold.

Fig. 3.8 illustrates the refined MSP function for models shown in the top row of Fig. 3.2. The highlights at the chest of horse, shoulder of nepture, and head of dancing children models are eliminated. Fig. 3.9 shows another refinement result on the noisy surfaces.

## 3.3 Comparisons

Both of MSP and SDF [73] reveal the local volume of the object, but by different definitions. SDF is an approximation of the medial axis which describes the local volume of object at the surface point by the maximum inscribed ball attaching to the surface point. Ray casting through the interior of object is applied to approximate the diameter of the maximum inscribed ball. Both MSP and SDF can reveal the local volume well for cylin-



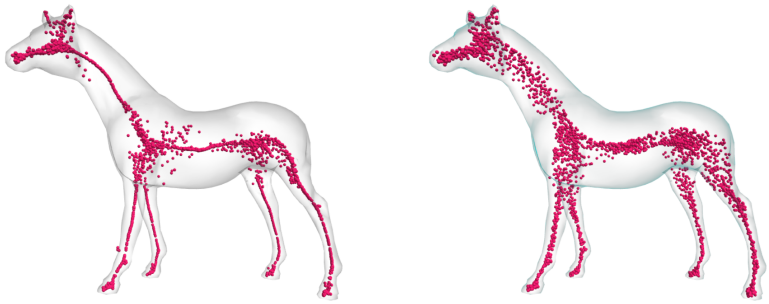
(a) MSP (b) SDF

(c) Gradient of MSP (d) Gradient of SDF

Figure 3.10: The distributions of MSP and SDF and their gradients.

drical parts, but MSP is capable of capturing better volume information than the SDF for those object parts that are non-cylindrical, such as the palm, as shown in Fig. 3.10. For non-cylindrical models or parts, the SDF value reflects only a portion of the internal volume information and measures the local thickness or local diameter of the model. The red color around the side of the palm means that the internal volume is much larger at the side of palm than at the center. Instead of MSP function, the minimum perimeter slice carries additional shape information about the object part, such as the slice shape and the orientation. The SDF, however, is unable to derive such information.

The surface function which describes the internal volume of the object is useful for part-based mesh segmentation. The averaging computation in SDF formulation can alleviate the problem induced by surface details or noise. It, however, tends to blur the change of volume, leading to some difficulties in deriving the part boundary, especially for the surface regions with dramatic change in local volume. Fig. 3.10(c) and Fig. 3.10(d) depict that the



(a) MSP (b) SDF

Figure 3.11: Vertex distribution of the shrunk mesh using MSP and SDF.

gradient of MSP changes significantly across the object parts, e.g. the regions between the fingers and the palm and the gradient of SDF does not change obviously across the object parts.

For skeletonization application, the centroid of the minimum perimeter slice has better representative for the curve skeleton, compared to the transformed vertices derived using the half-depth toward the inverse normal direction of SDF, as shown in Fig. 3.11.

## 3.4 Limitations

In some cases, the minimum perimeter slice does not faithfully capture the local shape of a part; e.g., at the feet of the dancing-children model in Fig. 3.2 where some planar slices pass across multiple parts and yield relatively larger MSP value than they should be.

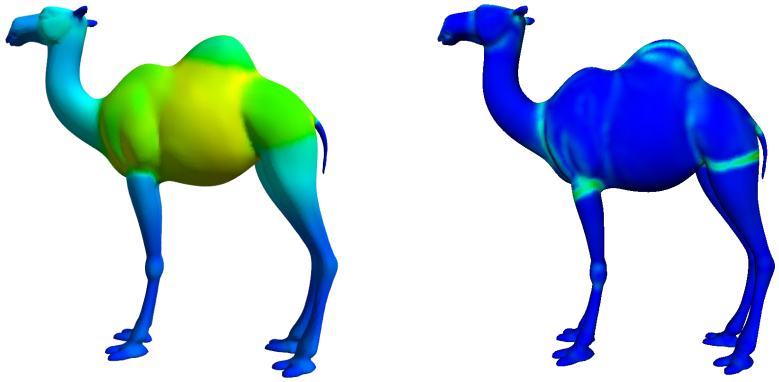
CHAPTER 4

# Volume Based Mesh Segmentation

## 4.1 Introduction

Due to the advance of 3D model acquisition equipment, 3D meshes with high polygon counts and complex topological structures can be obtained easily. As meshes are becoming larger and more complex, decomposing an object into smaller and simpler components is essential for many mesh techniques, including parameterization, texture mapping, morphing, editing, shape matching, compression and more. Thus mesh segmentation has become a key ingredient in many mesh applications.

Most of the existing mesh segmentation methods rely on either local detail geometric features or global topological structure. The use of too local or too global surface properties limits many segmentation algorithms to either decompose a model into several surface patches [21, 46, 60] or to handle only models with some specific topological structure such as core-salient features [36, 35]. As indicated in the part salience theory [29], the relative size of object region is an important characteristic for identifying the object part. The neighboring regions with similar volume tend to be grouped into a part and consequently the gradient of the local volume directly implies the potential regions for deriving part boundaries. The MSP function, describing the internal volume of the local region associated to the surface point, is best applicable for this scope. Fig. 4.1 depicts the MSP and the magnitude of MSP gradient on the camel model. The color ranges from blue to red



(a) MSP (b) Magnitude of MSP gradient

Figure 4.1: MSP and the magnitude of MSP gradient on the camel model.

represents the function value in ascending order. The magnitude of MSP gradient directly indicates the strength of surface regions to be boundaries.

Since complex models often contain features in different scales or salience, ranging from global structure to detail surface features, it is useful to decompose the models into components in several levels or hierarchically. Several approaches have been proposed in this direction. Hierarchical face clustering techniques construct the hierarchy in a bottom-up fashion [21]. Top-down approaches start from the root, which represents the whole object, and partition the component into two or more parts [36]. This process continues recursively until a certain condition is met. At each level of the top-down approach, the segmentation is usually derived implicitly by locating the best boundary between parts. Several hierarchical segmentation techniques have been proposed [36, 35, 45, 40]. Most of the techniques claim that components on a higher level reveal higher degree of salience than their descendant parts. But many of them cannot ensure that the components on each level of hierarchy have similar degree of salience.

Locating boundary between parts can be done by either boundary-based or region-based approach. Boundary-based approach uses local geometric properties, such as curvature, to locate boundary. Region-based approach seeks for regions with similar properties, such as the combination of geodesic and angular distances in [36], from which boundaries are derived. Since parts have different levels of salience, to evaluate the significance of a boundary between parts, we need to include the salience measures of the associated parts. However, the boundary computed by using boundary-based and region-based approach usually lacks the necessary salience measures for the parts associated with the boundary.

In this chapter, we introduce a new hierarchical segmentation scheme that decomposes

an object into meaningful parts in such a way that not only components on a higher level reveal higher degree of salience than their descendant parts but also the components on each level of hierarchy share similar degree of salience. Our segmentation is based on the MSP function which represents the object's internal volume on the surface. As neighboring regions having similar internal volume tend to be grouped together and form a part, the magnitude of MSP gradient can be used to locate the candidate segmentation regions that contain the boundaries. Moreover, the significance of segmentation regions is evaluated in the process of hierarchical segmentation. The evaluation takes into account the salience information of the parts associated with the segmentation region.

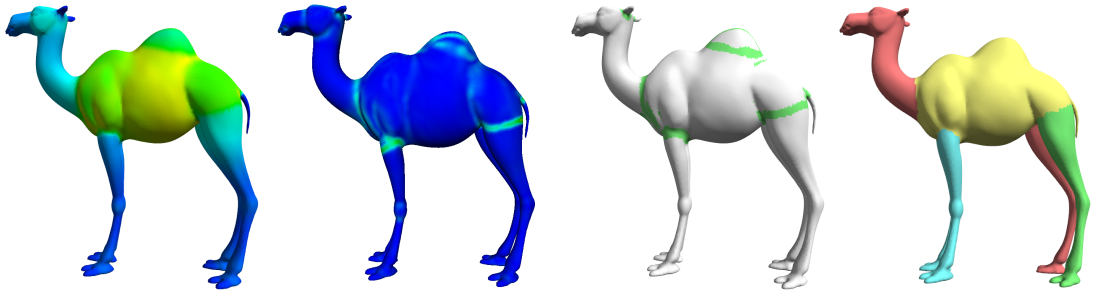
The proposed hierarchical segmentation scheme starts from the whole object and, for each level of the hierarchy, locates segmentation regions by applying a threshold to the function for the magnitude of MSP gradient, then evaluates the significance of segmentation regions. Finally, the scheme selects a set of the most significant segmentation regions for that hierarchy level. The boundaries for that level are then computed by using graph cut [36] with a capacity that considers both the curvature and the magnitude of MSP gradient.

We make the following contributions to the mesh segmentation process:

1. Propose a segmentation scheme based on the MSP function that encodes local volume information. The proposed segmentation scheme has several advantages:
   - Regions of similar volume are grouped into a part according to the MSP function.
   - A significance measure of a boundary that takes into account the local curvature and the changes in MSP value is presented.
2. Propose a hierarchical segmentation on which not only components on a higher level reveal higher degree of salience than their descendant parts but also the components on each level of hierarchy have similar degree of salience.

## 4.2 Volume Based Mesh Segmentation

The proposed segmentation scheme begins by deriving the gradient of MSP function for each face. Then, a set of segmentation regions is found by applying a threshold value to



(a) MSP (b) Magnitude of MSP Gradient (c) Segmentation Regions (d) Segmented Parts

Figure 4.2: Steps of the mesh segmentation process.

the magnitude of MSP gradient. Each segmentation region divides the object into two or more potential object parts. We test the saliency of the potential parts and obtain some most significant segmentation regions. Finally, a cut is derived within each selected segmentation region which separates the object into two parts. Figure 4.2 illustrates such segmentation process.

### 4.2.1 Segmentation regions finding

#### 4.2.1.1 Computing the gradient of MSP function

The magnitude of MSP gradient represents the rate of volume change in the neighborhood of the surface point. As shown in Fig. 4.1, there are large gaps in volume's size between the body and the neck and between the body and front limbs of the camel model, which are revealed in the distribution of MSP function (Fig. 4.1(a)) and the magnitude of its gradient (Fig. 4.1(b)). Computing the gradient of MSP function on a piecewise linear polygon mesh is not as trivial as that on the continuous surface. To compute the gradient of MSP function at a surface point $x$, we first derive a difference-vector for every edge in a neighborhood of $x$ and then compute the MSP gradient vector at $x$ as the average of all difference-vectors. The difference-vector for an edge is the vector in the direction of the edge vector and with the magnitude as the difference of the MSP values at its two endpoints.

The gradient of MSP function at $x$ is finally the averaged MSP difference-vector at $x$. Its detail equation is as follows:

$$\nabla \mathbf{MSP}(x) = \frac{1}{|N_x|} \sum_{e \in N_x} \left( \mathbf{MSP}(v_i) - \mathbf{MSP}(v_j) \right) \bar{e}, \tag{4.1}$$

where $N_x$ is the neighborhood of $x$ with a user-specified range, $v_i$ and $v_j$ are two endpoints of the edge $e$, possibly clamped to be within $N_x$, and $\bar{e}$ is the unit vector of the edge $e$ with direction from $v_j$ to $v_i$. Such a formulation is similar to the method for computing the curvature tensor in Alliez et al.[1]. For efficiency consideration, we define $N_x$ as the surface region within the sphere centering at $x$ with a user-specified radius.

#### 4.2.1.2 Deriving an appropriate threshold value

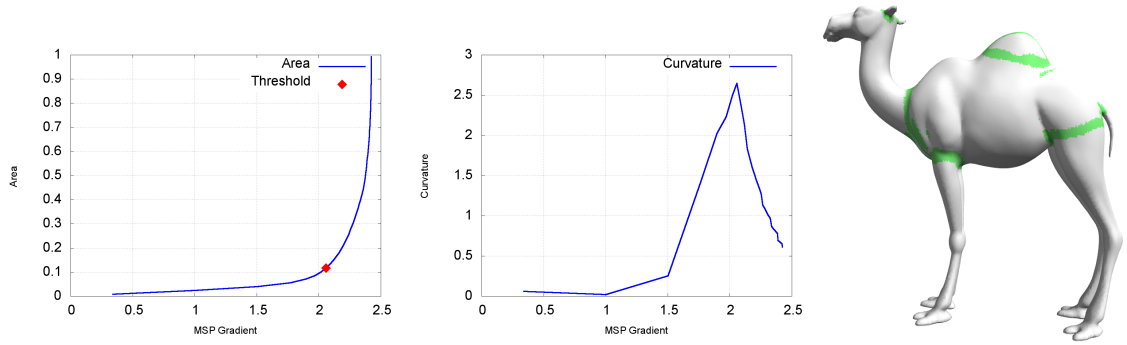
The segmentation regions are defined as the surface regions where the magnitude of MSP gradient is above a specific threshold value. An appropriate threshold value is hard to find in practice. A smaller threshold value will enlarge the segmentation regions, making the computation of a proper cut boundary more difficult. On the other hand, for a large threshold value, the segmentation region may not form in loops, even using extrapolation. To decide an appropriate threshold value automatically, we consider the cumulative function for the surface area with respective to the magnitude of MSP gradient as follows:

$$A(V) = \int_0^V a(v)dv, \tag{4.2}$$

where $a(v)$ denotes the area of the surface region having the magnitude of MSP gradient as $v$ and $A(V)$ is the total area of the surface region that has the magnitude of MSP gradient less than or equal to $V$. $A(V)$ is a monotonically increasing function, representing the changes of cumulative surface area with respect to the magnitude of MSP gradient. A good threshold value will be the magnitude of MSP gradient value that indicates a sudden drop on the value of $A(V)$ when the magnitude of MSP gradient decreases. Hence, by considering $A(V)$ as a 2D curve segment, the desired threshold value will be at the position where the curvature is maximal. Fig. 4.3(a) and Fig. 4.3(b) illustrate the graph of $A(V)$ and its curvature for the horse model, respectively. The segmentation regions extracted using the derived threshold value is shown in Fig. 4.3(c).

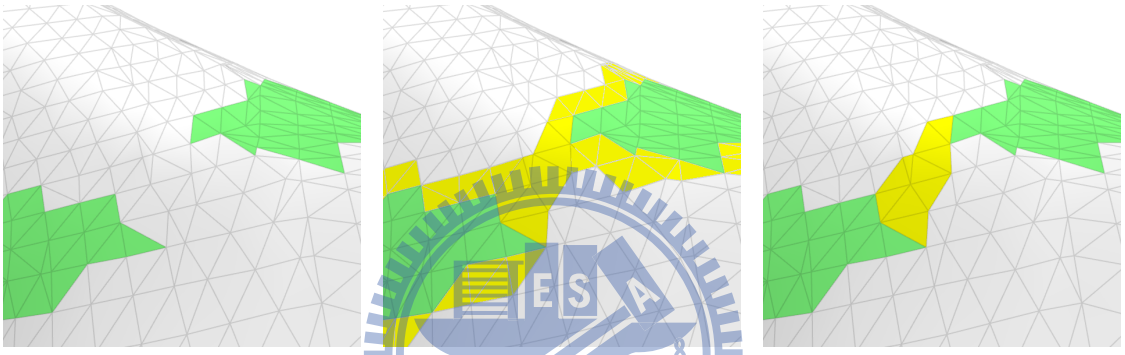
#### 4.2.1.3 Loop closing for segmentation regions

Since the cut boundary is extracted within the segmentation region, a segmentation region must form a loop. The segmentation region derived by applying threshold on MSP gradient in general does not guarantee to form a loop. We apply extrapolation to close those segmentation regions into loops. For a segmentation region that does not form a loop, a re-




(a) $A(V)$ (b) Curvature of $A(V)$ (c) Segmentation regions

Figure 4.3: $A(V)$ , curvature of $A(V)$, and segmentation regions derived for the camel model.



(a) Original segmentation regions (b) Region extrapolation (c) Grown region peering

Figure 4.4: The loop closing process.

gion growing process is performed starting from the region boundary, and in each iteration the face with the largest magnitude of MSP gradient is added to the segmentation region until the newly added face connects to another region or close the loop. After closing up a loop, the faces within the newly grown region are peeled away iteratively in the decreasing order of their magnitude of MSP gradients until a ribbon region with one face width left between the two newly connected segmentation regions. At this point, these two segmentation regions are merged. The merging process is executed iteratively until all segmentation regions form in loops. Fig. 4.4 illustrates the loop closing process.

### 4.2.2 Significant segmentation pairs selection

The segmentation region derived using the magnitude of MSP gradient represents the strength of the part boundary, but does not reveal any salience information of the parts it separated. However, the salience of segmented parts associated with the segmentation regions may range from detail feature to global structure.

There may have more than two parts associated with a segmentation region. Thus for a segmentation region, we define a segmentation pair to be two parts sharing the same segmentation region. The part saliency theory states that the salience of a 3D part depends on three factors: its size related to the whole object, the boundary strength, and the degree of protrusion [29]. Since we have derived the segmentation regions using the magnitude of MSP gradient, the local volume of an object part can be approximated by the average MSP value within the part region. The strength of boundary between a segmentation pair can be described as the difference of the averaged MSP values of the two parts. In order to measure the protrusion of the object part, we use the salience metric proposed by Gal and Cohen-Or [19] due to its computational efficiency and practicability for identifying surface features. Thus, we assign a salience-measure to each of segmentation pair $(P_a, P_b)$ as follows:

$$\mathbb{S}(P_a, P_b) = \mathbf{min}(\mathbf{RMSP}(P_a), \mathbf{RMSP}(P_b))\,\mathbf{min}(S(P_a), S(P_b))\,\|\mathbf{RMSP}(P_a) - \mathbf{RMSP}(P_b)\|, \tag{4.3}$$

where

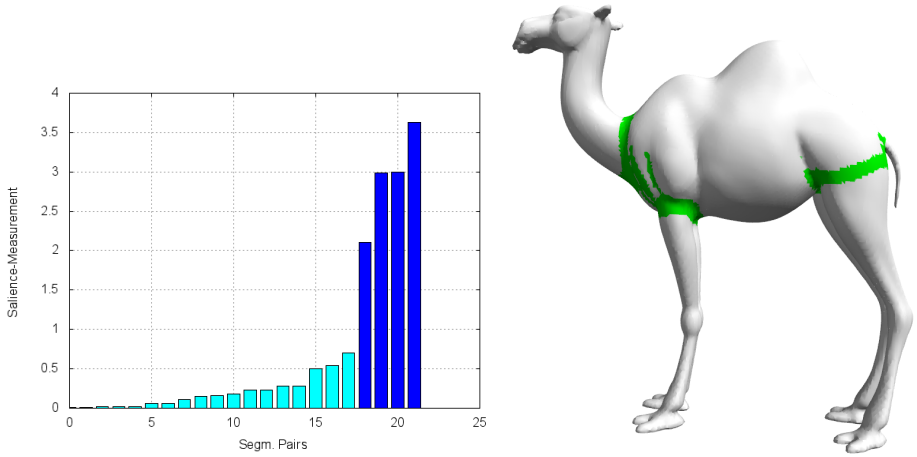
$$S(P) = \sum_{f \in P} Area(f)\, Curvature(f)^3,$$

and $P_a$ and $P_b$ are the two parts sharing the segmentation region, $\mathbf{RMSP}(P)$ is the averaged MSP value of the part $P$. $S(p)$ denotes the saliency of the part $P$, which is derived by the combination of surface area and the curvature. We use the Gaussian curvature in saliency computation since it has better description for the protrusion of object part.

To find the most significant segmentation pairs, we sort the value of $\mathbb{S}$ for all segmentation pairs in ascending order into a sorted sequence $\{\mathbb{S}^*(i)\}$, and seek an index $k$ such that $\mathbb{S}^*(k) - \mathbb{S}^*(k-1)$ is the maximum; that is, look for the largest gap among the sorted $\mathbb{S}^*(i)$. Those segmentation pairs that have $\mathbb{S}$ value higher than $\mathbb{S}^*(k-1)$ will be chosen as the segmentation regions for the current hierarchy level. As shown in Fig. 4.5, there is a large gap in the histogram of salience-measure between the segmentation pairs 17 and 18, and segmentation pairs 18 to 21 are selected.

### 4.2.3 Cut boundary extraction for the selected segmentation pair pair

Once a segmentation pair is selected, we next compute a cut boundary between parts by performing a modified graph cut [36]. We propose a hybrid capacity that combines the




(a) Salience-measure function histogram (b) Selected segmentation pairs

Figure 4.5: Selection of the segmentation pairs.

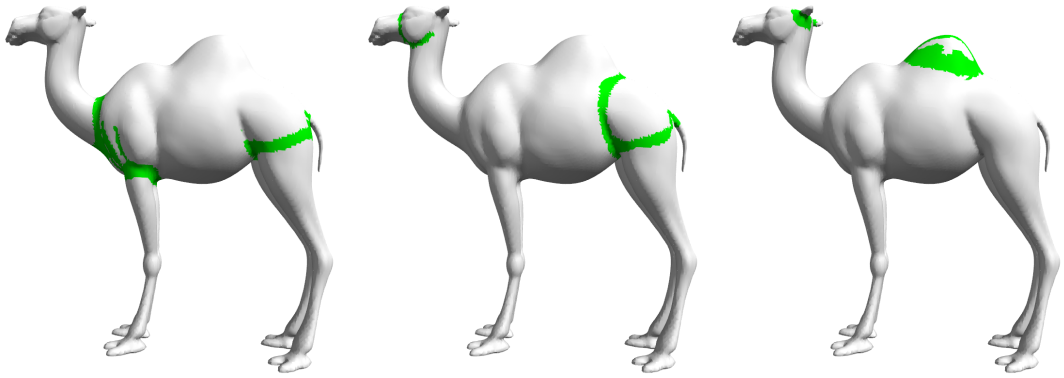
difference of MSP as well as angle difference as follows:

$$\mathbf{cap}(i,j) = \begin{cases} \dfrac{1}{1+\left(\dfrac{\theta(e_{ij})}{\theta_{avg}}\right)^2 + \dfrac{\Delta\mathbf{MSP}(e_{ij})}{\Delta\mathbf{MSP}_{avg}} + 2\dfrac{\|e_{ij}\|}{\|e_{avg}\|}} & e_{ij} \in E, i, j \neq S, T, \\ \infty & otherwise, \end{cases} \tag{4.4}$$

where $i$ and $j$ are two adjacent faces on the object that are not in both the source region $S$ and the target region $T$, $e_{ij}$ denotes the edge between faces $i$ and $j$, $E$ is the set of all the edges of the object, $\theta(e_{ij})$ is the dihedral angle of $e_{ij}$ which can be normalized by the average dihedral angle $\theta_{avg}$, and $\Delta\mathrm{MSP}(e_{ij})$ denotes the MSP difference across $e_{ij}$ which can be normalized by the average of MSP difference $\Delta\mathrm{MSP}_{avg}$, $\|e_{ij}\|$ is the edge length of $e_{ij}$ which can be normalized by the averaged edge length $\|e_{avg}\|$. $\theta_{avg}$, $\Delta\mathrm{MSP}_{avg}$, and $\|e_{avg}\|$ are computed using all the dihedral angles, all the faces, and all the edge of the object, respectively. The $\frac{\|e_{ij}\|}{\|e_{avg}\|}$ is a compensated term aiming to straighten the cut boundary since the path generated by using the tip of the magnitude of MSP gradient may not always be smooth.

## 4.3 Hierarchical Segmentation

The selection of the most significant parts directly implies that a hierarchical segmentation can be obtained easily by performing the segmentation process iteratively. For each level of hierarchy, the selection of the most significant segmentation pairs ensures that the selected



(a) Level 1 (0.148) (b) Level 2 (0.102) (c) Level 3 (0.07)

Figure 4.6: Segmentation regions at three levels on the camel models.

(a) Level 1 (b) Level 2 (c) Level 3

Figure 4.7: Hierarchical Segmentation of the camel model in three levels.

parts will have similar salience significance while having large differences compared to the remaining parts. Thus we obtain a top-down hierarchical segmentation that decomposes the object into parts into levels that reveal the salience significance down from the global structure to local features.

Our segmentation regions are found by applying a threshold value to the cumulative function shown in Eq. 4.2. In order to capture more boundaries in finer levels, for each level we ignore the surface areas nearby the part boundaries derived in previous levels in computing Eq. 4.2. As a result, a lower threshold value on the histogram can be obtained to enlarge the segmentation regions. Fig. 4.6 illustrates the segmentation regions found in three levels for the camel model. We observe that the threshold value applied decreases as the segmentation level gets deeper (see the numbers inside the parentheses) and the segmentation regions for smaller features, such as the mouth and the fingers, are found in finer levels. Fig. 4.7 reveals the segmentation results of the camel model corresponding to

the segmentation regions in Fig. 4.6.

## 4.4 Experimental Results

(a) hand (b) dinosaur (c) armadillo

(d) santa (e) nepture

Figure 4.8: The segmentation result of different models.

By using the volume information encoded in the MSP function, the proposed segmentation scheme can handle models of different topological types. Fig. 4.8 demonstrates the segmentation result for models shown in Fig. 3.2. Observed that the cut boundaries are located along the regions where there exists a large gap in MSP value and in the meantime follow the object's local features. Moreover, the salience-measure ensures that the most visually significant parts are segmented. The dinosaur (Fig. 4.8(b)) and armadillo (Fig. 4.8(c)) models have complex surface details and hence it is difficult to locate correct cut boundaries based on minima-rule alone. Since the MSP function is less sensitive to the

surface detail noise, we can generate meaningful parts and reasonably good boundaries using the proposed hybrid capacity on these two models. The nepture (Fig. 4.8(e)) model has genus higher than 1 and does not have obvious core-salient structure. Such kind of model can hardly be handled well using segmentation methods based on the global shape properties such as averaged geodesic distance [36, 35]. On the other hand, the proposed method finds no difficulty on such models since the volume information encoded by MSP function is less global and provides enough cues for identifying the parts from the models. Some smooth artifacts on cut boundaries may still be observed on some segmentation results such as the cut boundaries between the hind legs and the body of the camel (Fig. 4.7) and between the thighs and the body of the armadillo (Fig. 4.8(c)) and of nepture (Fig. 4.8(e)). In such regions, the tip of the magnitude of MSP gradient has large amount of disturbance and twice of compensation term in Eq. 4.4 may be not enough for yielding smooth boundaries. Moreover, we assume that the internal volume is uniform within the object part while noticeable volume change exists between parts. For models with some adjacent parts that have no noticeable volume change between them, our segmentation scheme may fail to separate them into different parts, such as the case in Fig. 4.8(e) where the hand and trident are not separated.

In Fig. 4.9, we compare the proposed method to other methods provided in the segmentation benchmark [10]. Cup and chair models do not have obvious core-salient structure on which the core extraction [35] and $K$-means [77] algorithms fail to produce good segmentations. The segmentation using SDF is based on a global fitting of the histogram function and is unable to reflect the local changes in object volume, leading to biased segmentation boundaries. Moreover, the SDF cannot correctly describe the non-cylindrical part and generates improper segments on the cup model. The randomized cuts method [22] generates good segmentation results for most of the models. However, its performance strongly depends on the segmentation methods based on local curvature and the geodesic distance. In consequence, it may not generate good segmentations for models with complex local features. Fig. 4.10 shows the segmentation result of dinosaur model using the proposed scheme and the randomized cuts [22]. The proposed segmentation can tolerate the complex local features of the dinosaur model and segments the models into parts with similar salience significance. On the other hand, randomize cuts algorithm produces improper segmentation boundaries at the neck, the body, and the tail. We also perform the benchmark

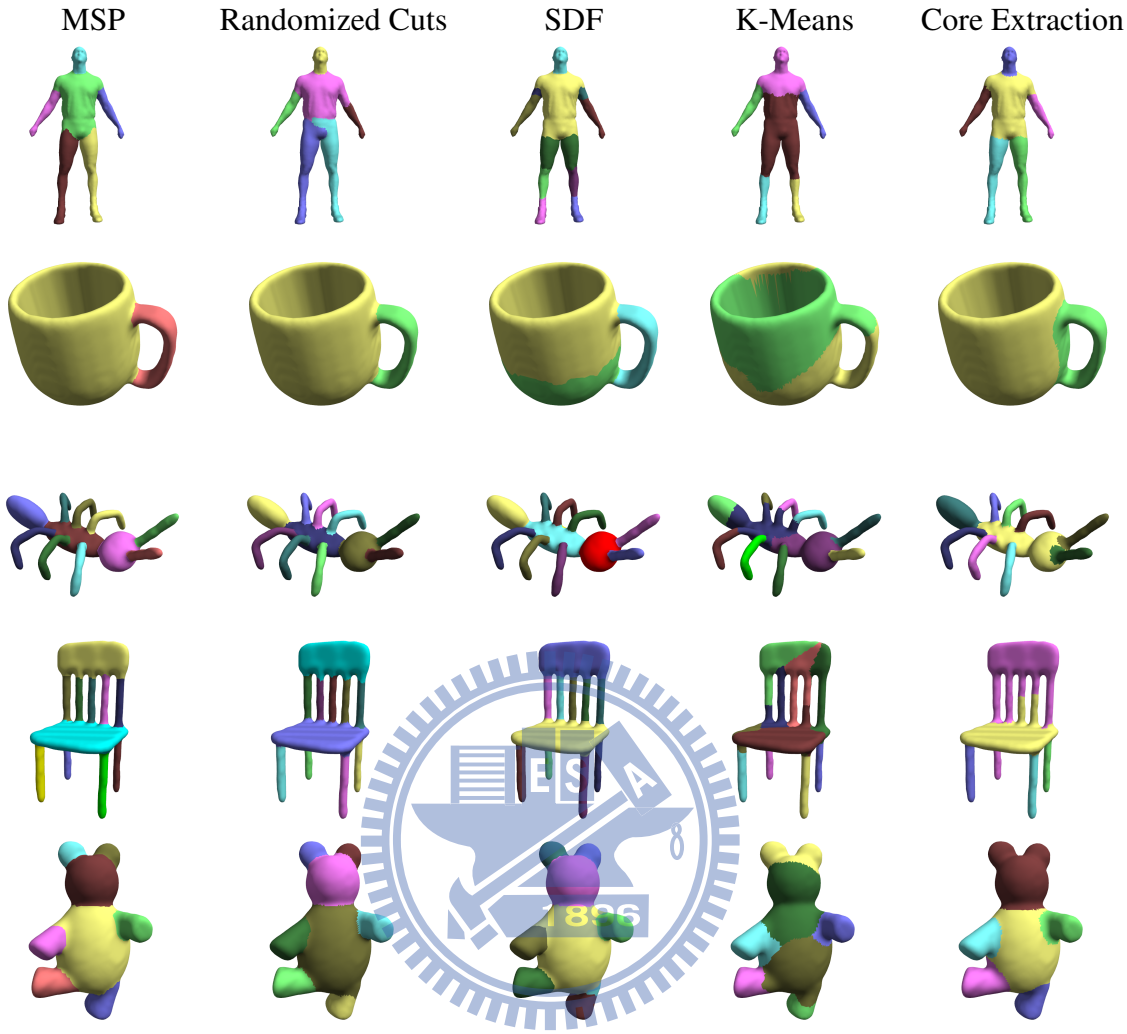
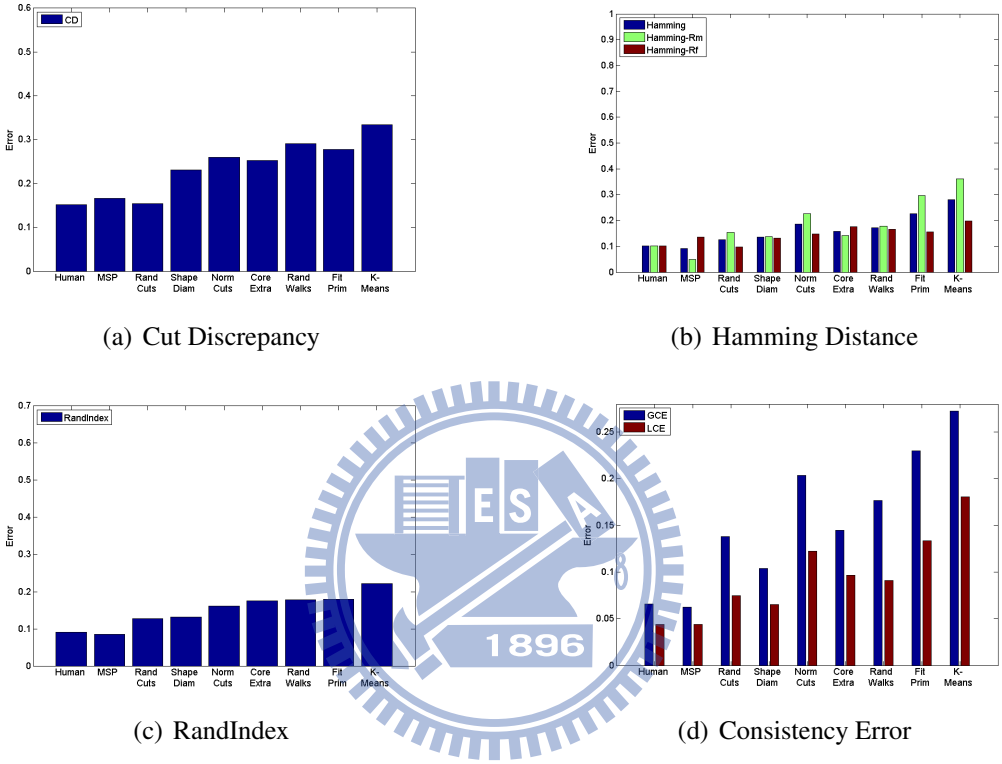



Figure 4.9: Comparison of the segmentation methods.

(a) MSP (b) Randomized Cuts

Figure 4.10: Comparison of dinosaur result using MSP and randomized cuts [22].

study of the proposed method against others using the segmentation benchmark [10]. The benchmark is obtained by performing the comparison based on 20 models selected from the object database of the segmentation benchmark (two models from each object category). Fig. 4.11 reveals that the proposed segmentation scheme always yields lower error than the four other metrics proposed in [10].




(a) Cut Discrepancy

(b) Hamming Distance

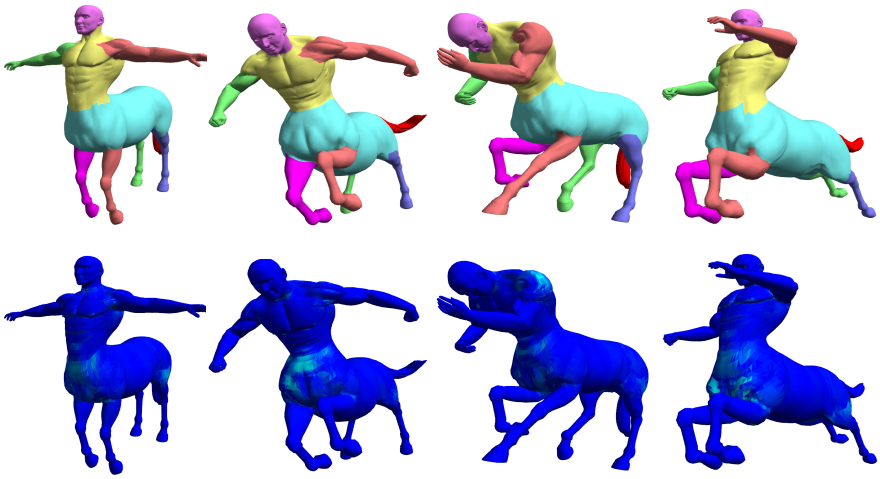
(c) RandIndex

(d) Consistency Error

Figure 4.11: The benchmark of our segmentation method.

Since the interior volume of a model is almost constant during animation, the proposed segmentation scheme is inherently pose invariant. We list the segmentation result of the animated centaur model in four poses in the top of Fig. 4.12. The bottom of Fig. 4.12 illustrates the average error rate of MSP function for each of the four poses. For each pose, the MSP's average error rate is computed by averaging the differences in MSP value between the pose and all other poses. The MSP is almost invariant to the change of pose, except in some joint regions where very small deviation of MSP value may exist.

We decompose the dinosaur and armadillo models into a hierarchy of four levels, as shown in Fig. 4.13. The columns from left to right indicate the levels in ascending order. The most significant parts, such as the body of armadillo and the four limbs, are decomposed at the first level. As going down in the hierarchy, we observe that parts at the same level have similar salience significance and parts in lower levels have less salience signif-

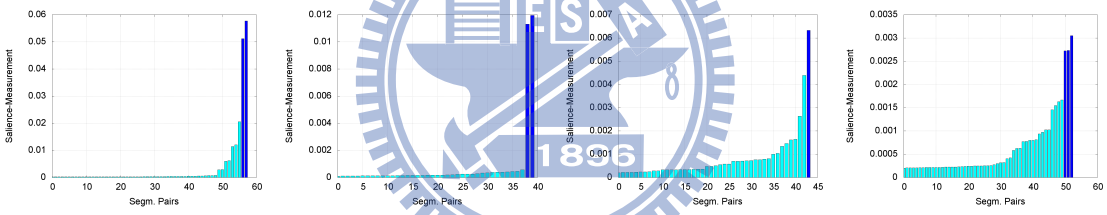


| | pose 0 | pose 1 | pose 2 | pose 3 |
|---|---|---|---|---|
| Avg. error rate | 0.012 | 0.009 | 0.011 | 0.015 |

Figure 4.12: Segmentation and average error rate for different poses of the animated centaur model.

icance. Fig. 4.14 depicts the histogram plots of salience-measure for the dinosaur model. The parts having similar meaning tend to have similar values of salience-measure and will be decomposed at the same level. Fig. 4.15 lists the hierarchical segmentation result using SDF [73]. The boundaries of core part for the dinosaur and armadillo are varying among different levels. Parts at the same level might differ greatly in salience significance and, moreover, parts at lower levels may not have less salience significance.

Figure 4.13: Hierarchical segmentation result of the dinosaur and armadillo models.




(a) Level 1 (b) Level 2 (c) Level 3 (d) Level 3

Figure 4.14: Histograms of the salience-measure at four levels for the dinosaur model.

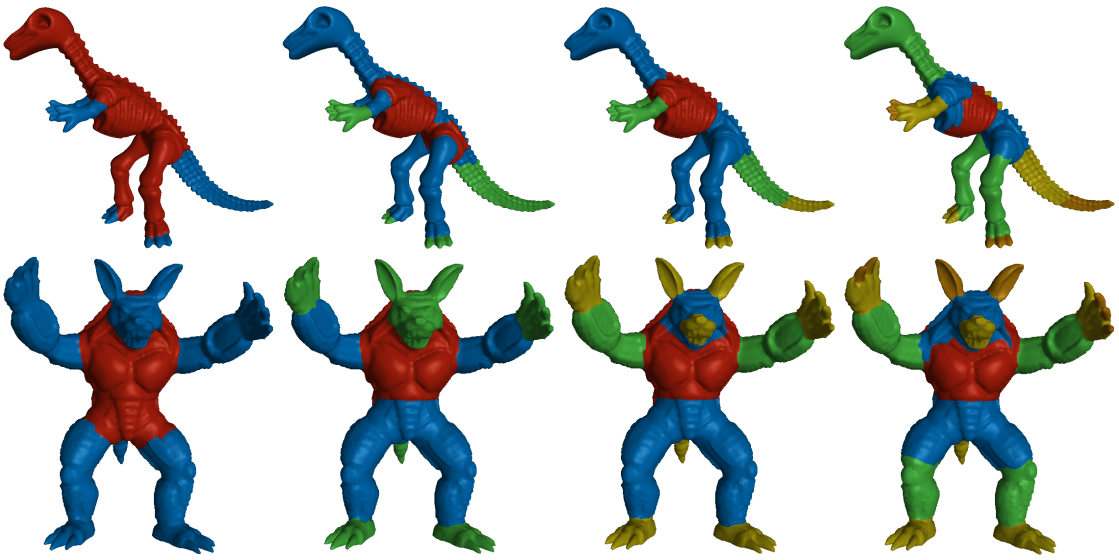


Figure 4.15: Hierarchical segmentation result of the dinosaur and armadillo models using SDF [73].

CHAPTER 5

# Mesh Skeletonization using Volume Based Surface Function

## 5.1 Introduction

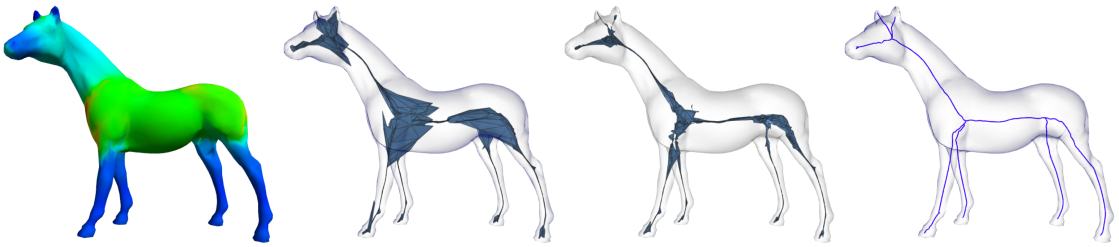
The skeleton of a 3D model is an 1D structure that represents the topological characteristics of the model. As a global shape descriptor of 3D models, the skeleton is useful for shape analysis, object retrieval, segmentation, and animation. In general, extracting skeleton from a 3D model is a costly process that usually requires the information about the internal volume of the model.

According to the short-cut [79], a good cut should be the shortest cut and cross an axis of local symmetry. Hence, the skeleton can be computed based on the union of the local symmetries of all short cuts on the surface. Based on the MSP function introduced in Chap. 3, we propose a new mesh skeletonization that derives the curved skeleton directly from the 3D model. The original model is shrunk directly to a skeleton-like shape while preserving the connectivity of the model by using the information associated with the minimum perimeter slices. A greedy framework is then invoked to iteratively alter the connectivity of the shrunk mesh and adjust its local geometry until an 1D curved skeleton is obtained. Although similar to the greedy framework commonly used in mesh simplification, our framework aims to degenerate the topology and to move transformed vertices towards the centerline. The mesh manipulation operator and the error metric are there-

fore completely different. The edge swap operator only alters the edge connectivity and is commonly used in mesh simplification to obtain a better mesh connectivity. Since the edge swap operator can be iteratively applied to reduce a 3D mesh to a 1D structure, it is used as a single edge operator in our greedy framework. To degenerate the shrunk mesh to a 1D curve skeleton, an error metric guides the edge swap sequence such that the edges with higher deviation from the centerline would be swapped first. Moreover, we apply a smoothing operation after each edge swap operation so as to move the vertices towards the centerline. To do so, a slice-deviation error derived from the minimum perimeter slice measures the deviation between a shrunk vertex and the centerline. The resulting skeleton of the greedy framework may contain small skeleton branches induced by the local surface features or noises. These small skeleton branches can be removed by testing the salience of their corresponding surface regions.

Our approach would attempt to retain vertices and to delete a small portion of the vertices associated with short branches during branch removal. The resulting skeleton possesses a dense node distribution at the core parts and around the junction nodes, and a skeleton-surface mapping. Previous skeleton extraction methods such as the ones using Reeb graph [27, 8, 83] or mesh contraction [6] usually generate skeletons with sparse node distribution, especially at the core parts of the model or around the junctions. Although the sparse node distribution may lead to zigzag skeleton structure in some cases, sparse nodes at the core parts are usually sufficient for some applications such as segmentation and skinning. Nevertheless, when we consider to embed the surface information to the skeleton, the denser node distribution is, the more information could be embedded. Some applications may be benefited greatly by the presence of dense node distribution together with the skeleton-surface mapping. For example, we are investigating how to find a surface-to-surface correspondence between two meshes by utilizing the skeleton-surface mapping and a mapping between dense nodes of two skeletons established with the aid of a well developed skeleton correspondence algorithm such as [7]. In this case, the density of skeleton nodes affects the accuracy of the surface correspondence.

Our contributions of this work are stated as follows. We propose a novel greedy framework for degenerating the 2-manifold connectivity of polygonal mesh to a 1D skeletal structure. The resulting skeleton possesses a dense node distribution at the core parts and around the junctions. A single salience parameter is required for controlling branch re-



(a) Original mesh (b) Shrunk mesh (c) Intermediate skeleton (d) Final skeleton

Figure 5.1: The skeletonization process.

moval so as to compute skeletons with varying details. Finally, the proposed method is able to generate consistent skeletons for models in different resolutions.

## 5.2 Mesh Skeletonization

To compute the skeleton, we first shrink the 3D mesh to a skeleton-like shape by transforming each vertex to the local symmetry axis approximated by the geometric center of the minimum perimeter slice associated with the vertex. A greedy edge-swap process based on the edge swap operator is invoked to iteratively degenerate the shrunk mesh to an 1D skeleton structure. A specially designed metric is proposed to guide the edge-swap sequence such that the edges deviate farther from the centerline would be swapped first. Furthermore, after each edge swap the vertices of the swapped edge are refined towards positions that represent the skeleton as the centerline by using a smoothing operator. The resulting skeleton may contain undesired small branches induced by detailed geometric features, noise, or the improper orientation minimum perimeter slices. After a branch is formed in the edge-swap process, we compute the saliency of the surface region corresponding to the branch and remove the branch if its associated salience value is smaller than a user-specified threshold. So that the threshold can be intuitively tuned and skeletons with varying resolutions can be effectively generated.

Fig. 5.1 depicts the proposed skeletonization process. First, the original mesh is shrunk to a skeleton-like shape, as shown in Fig. 5.1(b). Second, a sequence of edge-swap operators is performed to transform the shrunk mesh to a 1D skeleton structure. An intermediate result is shown in Fig. 5.1(c) and the resulting 1D skeleton is shown in Fig. 5.1(d).

### 5.2.1 Mesh Shrinking

The minimum perimeter slice associated with a surface point $p$ approximates the short-cut passing through $p$. The skeleton point that corresponds to the surface point $p$ therefore can be approximated by the geometric center of the simple polygon forming the minimum perimeter slice. Accordingly, the mesh is shrunk to a skeleton-like shape by transforming each surface point to its corresponding geometric center. Note that MSP slices, rather than the refined MSP slices, are used in the proposed skeletonization method since the metric that guides the edge-swapping and the smoothing operator to be described later is able to effectively handle the outlier cases of the shrunk mesh.

The shrunk mesh is, therefore, similar to the resulting skeleton in shape and preserves the connectivity of the original mesh. Some of the transformed vertices may deviate from the skeleton path and form protrusions on the shrunk mesh, especially around the skeleton junctions as shown in Fig. 5.1(b). However, most of the transformed vertices are distributed along the skeleton path, providing a good hint on how to locate the skeleton and how to adjust the outliers towards the skeleton path.

### 5.2.2 Mesh Degeneration

To extract an 1D skeleton from a manifold mesh is a degeneration process. An edge-swap operator flips the common edge of two adjacent triangles to the edge connecting two opposite vertices. Traditionally, the edge-swap operator is used in level-of-detail modeling to adjust the triangle connectivity in order to refine the mesh connectivity, while geometry simplification operators such as edge collapse are used to simplify the mesh. When the edge-swap operator is iteratively applied to a 3D mesh, the mesh can be degenerated to a 1D structure while all vertex positions are retained; as shown in Fig. 5.2 in which a tetrahedral is degenerated to a 1D structure. Our skeleton extraction algorithm is a greedy framework based on an edge-swap operator which is used for degenerating the shrunk mesh into a 1D skeleton structure. In this subsection, the error metric, smoothing operator, and branch removal scheme will be described.

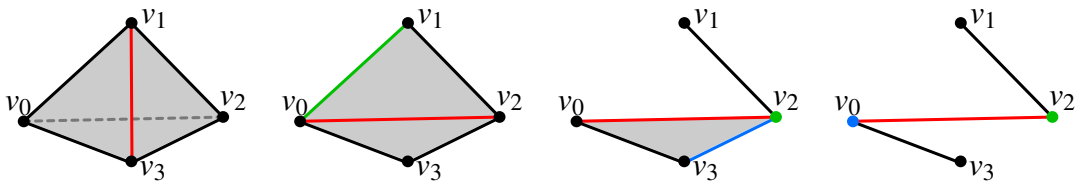
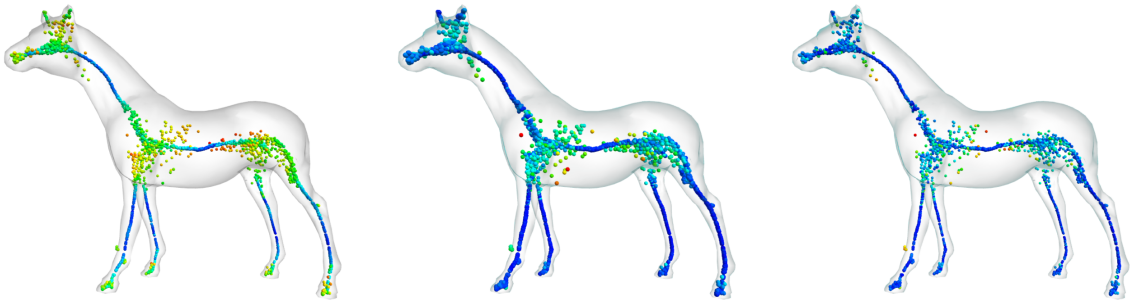



Figure 5.2: A tetrahedral is degenerated to a 1D structure using edge swap.



(a) Slice-deviation error (b) Vertex sparseness (c) Vertex protrusion

Figure 5.3: Metric associated with the transformed vertices on the horse model. The color ranging from blue to red represents the increasing value.

### 5.2.3 Error Metric

In traditional mesh simplification algorithms, the error metric measures the error incurs before and after the simplification [20]. The primitives with smaller error have higher priority to be simplified. In our case, the primitives are edges formed by transformed vertices. We need an error metric to measure how much an edge deviates from the skeleton path and the amount of changes in shape before and after an edge swap operation is performed.

To measure how much a transformed edge deviates from the skeleton path, we first measure how much its end vertices deviate from the skeleton path. For a surface vertex, its slice-deviation error defined by Eq. 3.2 measures how well its minimum perimeter slice approximates the short cut, which offers a good measure on how much its corresponding transformed vertex deviates from the skeleton path. For a transformed vertex $v$, we denote $c_{dev}(v)$ as the slice-deviation error of its original surface vertex. Fig. 5.3(a) shows the slice-deviation error of the transformed vertices of the horse model. Notice that minimum perimeter slices of some vertices near the ends of skeleton or junctions usually have higher slice-deviation errors.

Adopting the slice-deviation error alone does not guarantee that the shape of the skeleton would be preserved. This is because the slice-deviation error describes only the deviation of the minimum perimeter slice from the short cut. To preserve the shape and produce

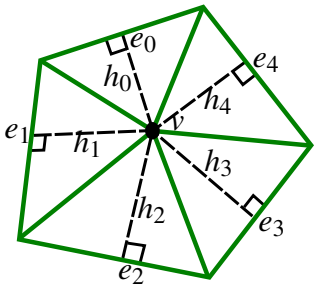



Figure 5.4: Measurement of the vertex sparseness.

a smooth curved skeleton, the change of the shape on the shrunk mesh should be taken into account. We observe that the transformed vertices usually distribute around the skeleton path and the vertices with higher slice-deviation error deviate farther from the skeleton path, are less dense and form protrusions on the shrunk mesh. In addition to slice-deviation error, we define the vertex density and the vertex protrusion for a vertex as the shape preserving metric. Measuring the vertex density in Euclidean space is a difficult task as an additional acceleration structure is required. Instead, we compute the sparseness of the transformed vertex $v$ as the averaged distance between $v$ and all the opposite edges of the one-ring faces of $v$, as shown in Fig. 5.4. The corresponding equation is given as follows:

$$c_s(v) = \frac{1}{n}\sum_{1}^{n} h_i. \tag{5.1}$$

The vertex protrusion $c_p(v)$ measures the protrusion of the transformed vertex $v$ relative to its one-ring neighborhood as shown in Eq. 5.2 as follows

$$c_p(v) = v - \sum_{v_i \in V} v_i, \tag{5.2}$$

where $V$ denotes the one-ring vertices of the vertex $v$. Fig. 5.3(b) and Fig. 5.3(c) illustrate the sparseness and the protrusion of the transformed vertices.

The skeleton is an 1D edge structure. Hence, we need one more metric for measuring the area difference of the shrunk mesh before and after swapping an edge $e$. This metric is called area-difference error $c_A(e)$. To increase the convergence rate of the skeletonization process, we would give the transformed edge with higher area-difference error to have higher priority to be swapped.

Finally, the metric used to guide the edge swapping process is a combination of slice-deviation error, vertex sparseness metric, vertex protrusion metric, and the area-difference

error. It is given in Eq. 5.3 as follows

$$C(e) = \frac{c_{dev}(e)}{\pi} + \frac{c_s(e) + c_p(e) + c_A(e)}{3\|e_{max}\|}, \tag{5.3}$$

where $\|e_{max}\|$ denotes the length of the longest edge in the shrunk mesh and is used for normalization, and $c_{dev}(e)$, $c_s(e)$ and $c_p(e)$ are computed as the averaged slice-deviation error, averaged sparseness and averaged protrusion of the two end vertices, respectively. At each iteration step of the edge-swapping process, the edge with the highest $C(e)$ is swapped first; that is, conceptually the edge that deviates most from the centerline would be swapped first.

### 5.2.4 Skeleton path smoothing

The proposed edge-swapping process ensures that the manifold topology of the shrunk mesh is degenerated to a 1D skeleton structure while all the vertices are preserved. However, vertices with high slice-deviation error cannot represent the skeleton well, leading to a zigzag skeleton path. To smooth the skeleton path, we perform a smoothing operation on the vertices after each edge swap operation. Consider the edge $e$ with end points $v_{s_1}$ and $v_{s_2}$, as shown in Fig. 5.5. After $e$ is swapped, the vertices $v_{t_1}$ and $v_{t_2}$ are refined to $v'_{t_1}$ and

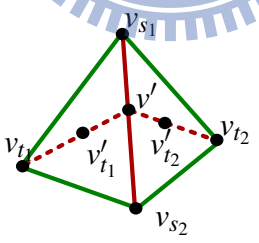
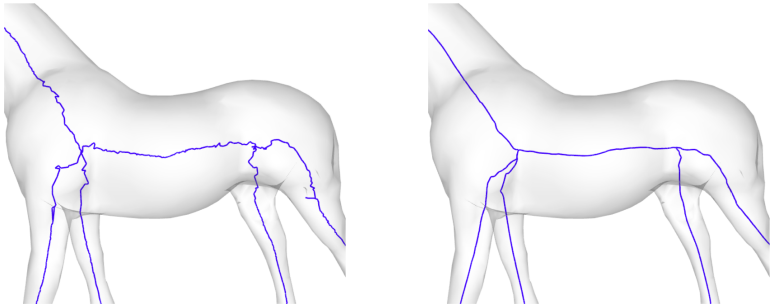



Figure 5.5: Bi-linear interpolation after the edge swapping.

$v'_{t_2}$, respectively. $v'_{t_1}$ and $v'_{t_2}$ are computed by a bilinear interpolation that first derives the interpolated position of $v_{s_1}$ and $v_{s_2}$, denoted as $v'$, and then interpolates $v'$ with $v_{t_1}$ and $v_{t_2}$, respectively. The slice-deviation errors are used to offer the weights such that vertices are moved towards the centerline. Eq. 5.4 and Eq. 5.5 list the equations.

$$\begin{aligned} v'_{t_1} &= \frac{v' c_{dev}(v_{t_1}) + v_{t_1} c_{dev}(v')}{c_{dev}(v') + c_{dev}(v_{t_1})} \\ v'_{t_2} &= \frac{v' c_{dev}(v_{t_2}) + v_{t_2} c_{dev}(v')}{c_{dev}(v') + c_{dev}(v_{t_2})}, \end{aligned} \tag{5.4}$$



(a) Without path smoothing (b) With path smoothing

Figure 5.6: Effect of skeleton path smoothing.

where

$$v' = \frac{v_{s_1} c_{dev}(v_{s_2}) + v_{s_2} c_{dev}(v_{s_1})}{c_{dev}(v_{s_1}) + c_{dev}(v_{s_2})}, \tag{5.5}$$

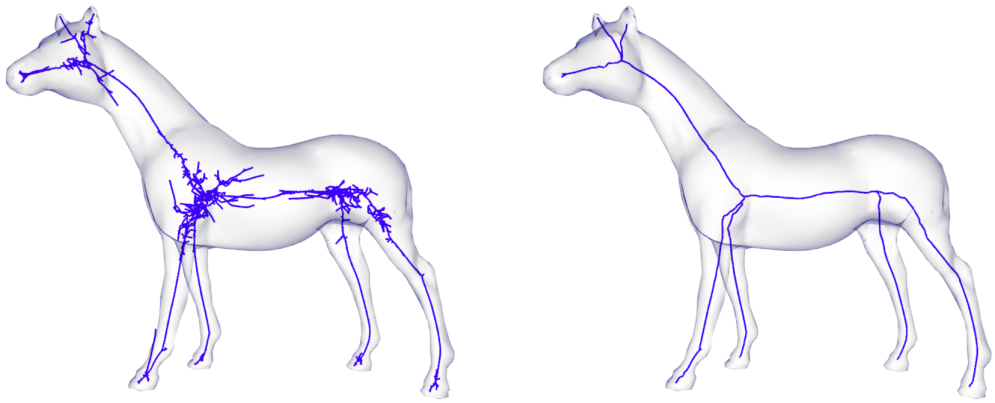
and $c_{dev}(v')$ is the interpolated value of the slice-deviation errors of $v_{s_1}$ and $v_{s_2}$.

The number of times for refining a vertex depends on the number of its outgoing edges being swapped. Fig. 5.6 depicts the resultant skeleton of a horse model without and with path smoothing enabled. In both cases, branch removal is enabled.

### 5.2.5 Skeleton branch removal

The skeleton resulted from the iterative edge swapping process may contain many small branches induced by detailed geometric features, noise, or the improper orientation minimum perimeter slices, as shown in Fig. 5.7(a). Such small branches may not be useful for describing the anatomical structure of the model. A skeleton branch represents a protrusion from the core part. So during our skeletonization process, a skeleton branch is removed if its corresponding surface region is insignificant in terms of the salience measurement.

According to the saliency geometric features proposed by Gal and Cohen-Or [19], the salience of a model's part is judged by the combination of the surface curvature and the area. The maximum curvature of the surface point on the protrude part can be described as the reciprocal of the radius of the maximum inscribed ball, which can be approximated by the distance between the surface point and the corresponding skeleton node. Moreover, the area of the protrusion can be described as the integral of the MSP function along the skeleton branch. Thus, we define the salience of the skeleton branch $b$ as the integral of the



(a) Without branch removal (b) With branch removal

Figure 5.7: Effect of branch removal.

MSP function multiplied by the curvature along the skeleton branch, as given in Eq. 5.6,

$$\mathbb{S}(b) = \int_{x \in b} \frac{\mathbf{MSP}(t(x))}{r(x)^3 (c_{dev}(t(x)) + 1)} dx, \tag{5.6}$$

where $t(x)$ denotes the transformation from the skeleton to the surface and

$$r(x) = \frac{\mathbf{MSP}(t(x))^2}{8\pi^2 \|x - t(x)\|} + \frac{\|x - t(x)\|}{2}$$

is the approximate radius of the maximum inscribed ball for the surface point $t(x)$. The division by slice-deviation error in Eq. 5.6 is used to reduce the contribution from poor representative MSP slices. Note that all the lengths in Eq. 5.6 are divided by the half length of the mesh diagonal for normalization.

During skeletonization, once a part of the shrunk mesh is degenerated to a 1D branch, the salience of the branch is computed. The branch is removed if its salience value is smaller than a user-specified threshold. Fig. 5.7 illustrates the resultant skeleton of a horse model without and with branch removal. In both cases, path smoothing is performed.

Since all the vertices are retained by the edge-swap operator with the smoothing operator and only a small portion of the vertices are deleted in the process of branch removal, the resulting skeleton possesses a dense node distribution.

### 5.2.6 Discussions

Our skeletonization scheme and contraction-based skeleton extraction [6] share similar ideas; but they are very different in terms of algorithms. Since connectivity is maintained

in the skeletonization, both methods generate skeletons that are homotopic to the original model. Both methods are rotation invariant and pose insensitive since they work directly on the original geometry.

We found in practice that the skeletons generated by [6] and ours are quite similar in shape; but have some different characteristics due to their algorithmic differences. For example, the skeletons generated by [6] have sparse nodes at the core parts of a model while the skeletons derived by the proposed method have a dense node distribution at the core parts and around the junctions. Moreover, the mapping vertices of a skeleton node in [6] form a cylinder-like or a sphere-like shape, which are much larger than the ones derived by our proposed method. Compared to [6], our skeleton-surface mapping together with the dense node distribution embed more accurate and detailed surface property or information on the skeleton, which are useful to applications such as surface-to-surface correspondence. Due to the fact that only vertices associated with insignificant branches are deleted in our skeletonization, the skeletons of models with extremely low polygon count can also be extracted. However, the method [6] may not be able to handle such kind of models. The saliency threshold in our approach is more intuitive to be specified than the initial weights for balancing the contraction and attraction constraints used in [6]. The initial weights do not have intuitive relation to the skeleton branching and higher weights do not guarantee to have branches for more detail features.

## 5.3 Results

We implemented our approach and applied it to 3D models with various topological types. Fig. 5.8 lists the resulting skeletons of some 3D models. Note that MSP, rather than the refined MSP, is used for all skeleton examples derived by the proposed method. The results show that the proposed skeletonization is homotopic not only for simple models but also for models with higher order genus, such as heptoroid, dancing children, or fertility. The small red spheres in Fig. 5.8 represent the nodes of the skeleton paths, indicating that the skeletons have a high node resolution at the core parts and junctions. Table 5.1 shows the percentages of surface vertices that are retained on the skeletons for some models.

Although MSP slicing is sensitive to noise, for applications such as segmentation and skeletonization such noises are harmless to the judgment of intermediate level features on the objects. The proposed skeletonization scheme is insensitive to the noisy surface models,

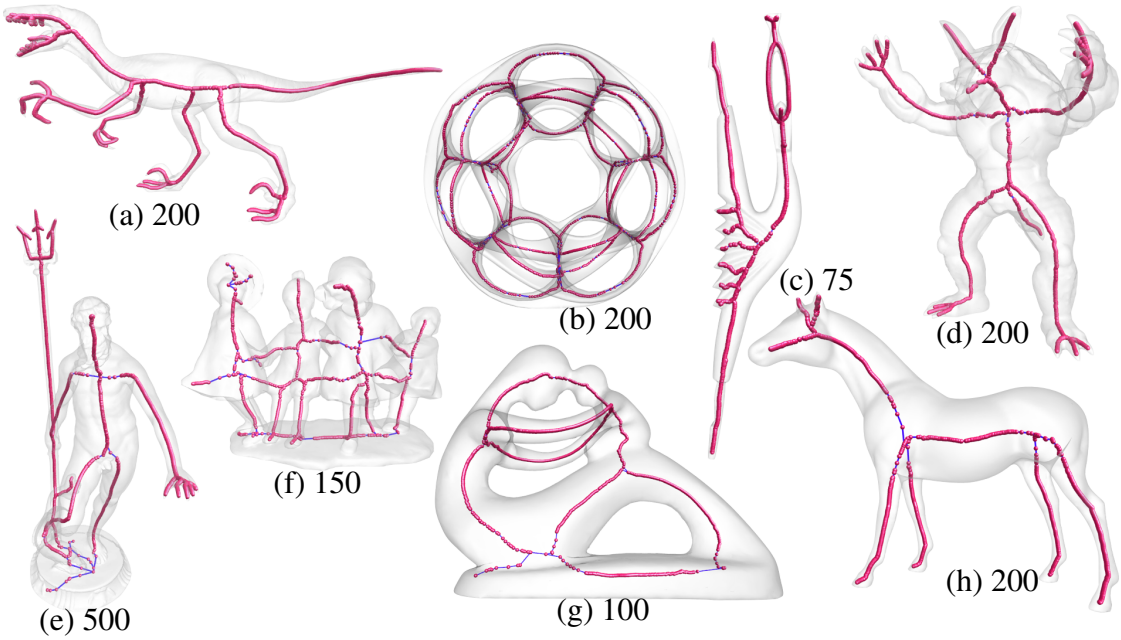



Figure 5.8: The skeletons of different models. The numbers indicate salience thresholds used and small red spheres represent the skeleton nodes.

Table 5.1: Number of surface vertices retained on the skeleton.

| Model | Number of vertices | Number of skeleton nodes |
|---|---|---|
| Raptor | 15,000 | 4,708 (31.39%) |
| Heptoroid | 9,950 | 3,234 (32.47%) |
| Dancer | 10,000 | 3,673 (36.73%) |
| Armadillo | 10,002 | 1,762 (17.62%) |
| Neptune | 29,996 | 5,515 (18.38%) |
| Dancing Children | 14,986 | 1,704 (11.37%) |
| Fertility | 14,994 | 3,266 (21.78%) |
| Horse | 19,851 | 4,033 (20.32%) |

as shown in our experiment. Fig. 5.9 shows the skeleton extracted from the noisy horse model, on which the resulting skeleton preserves the main structure of the horse model and is similar to the skeleton of the smooth horse model, as shown in Fig. 5.8. MSP function is pose invariant and so is the proposed skeletonization method. Fig. 5.10 shows the skeletons of the cat models of different poses in Fig. 3.4.

### 5.3.1 Comparisons

We compare our skeletonization method with the mesh contraction algorithm [6] which also derives the skeleton directly from the surface. The results, derived using the parameter setting as suggested in [6], are shown in Fig. 5.11. Our method and the mesh contraction

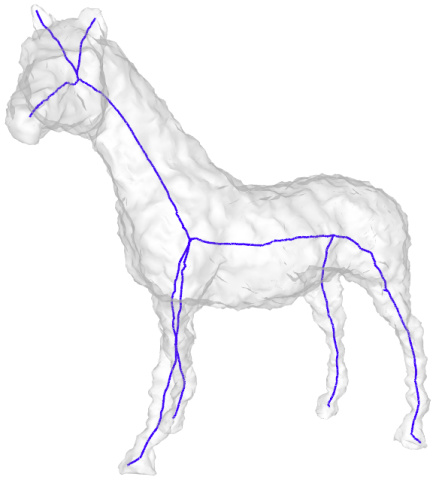


Figure 5.9: Skeleton extracted from a horse model with high noise.

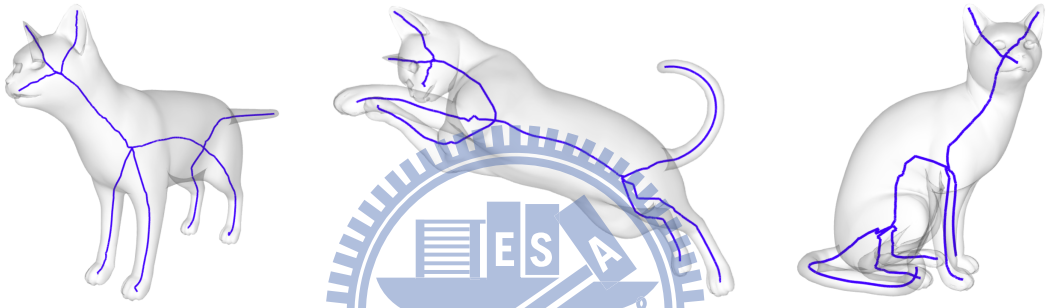


Figure 5.10: Skeletons for the cat model in different poses.

algorithm generate skeletons that are homotopic to the original models and are similar in shape. However, skeletons produced by our skeletonization have a denser node distribution than the ones produced by [6], especially at the core parts of the model and around the junctions.

Next, we show that our skeletonization provides a much denser mapping between the skeleton nodes and the surface than that of [6]. The skeleton-surface mapping are depicted using the raptor models of polygon count $2,500$ and $20,000$, as shown in Fig. 5.12. Notice that for models of higher resolution, the proposed approach retains more vertices for forming the skeleton nodes, leading to a dense skeleton-surface mapping; that is, each node maps to a smaller surface region.

The proposed skeletonization process requires only one user-defined parameter, namely the salience threshold. Fig. 5.13 shows the skeletons of different levels of details for the armadillo model. The skeleton computed with a smaller salience threshold preserves more branches for smaller geometric features, such as the tiny fingers (Fig. 5.13(b)). However, if the salience threshold is too small, many small branches are retained (Fig. 5.13(a)). On

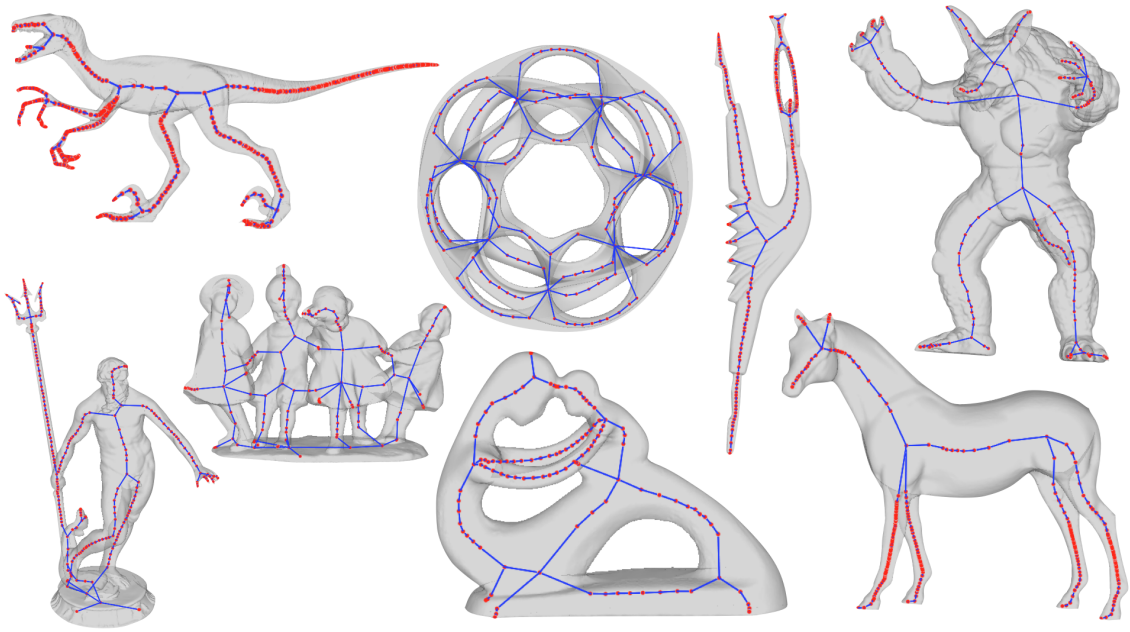


Figure 5.11: Skeletons computed by contraction method [6].

the other hand, a skeleton computed with a high salience threshold represents the main structure of the model. For the contraction method, it is hard to tune the parameter values for obtaining skeletons with decreasing levels of detail like the ones in Fig. 5.13.

It is desirable that skeleton extraction can be independent on the model resolution; that is, consistent skeletons can be extracted from a model in different resolutions. Fig. 5.14 shows the skeletons for the raptor model in different resolutions with MSP functions in Fig. 3.5, which are consistent in branching structure and shape. Notice that even for a model with low polygon count, such as 2,500 polygons, the skeleton can be extracted and retains the structure of the model, which is the limitation of [6].

### 5.3.2 Limitations

For certain man-made objects consisting of highly concave local regions, geometric centers of some MSP slices may not lay inside the object. In this case, the resulting skeleton is not guaranteed to be lying inside the model. Nevertheless, if the shrunk mesh is inside the object, the proposed skeletonization ensures that the skeleton is inside the object. Our skeletonization method may fail to generate high quality curve skeletons for the objects with holes. However, if the geometric centers can be well recovered or computed for those MSP slices that have missed segments, the skeletonization process is not affected by the holes.

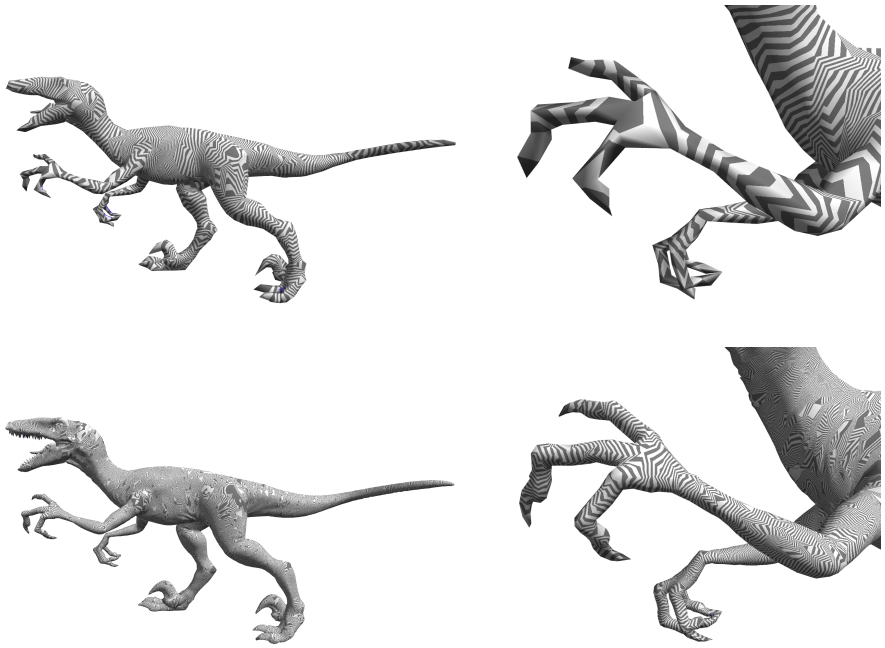
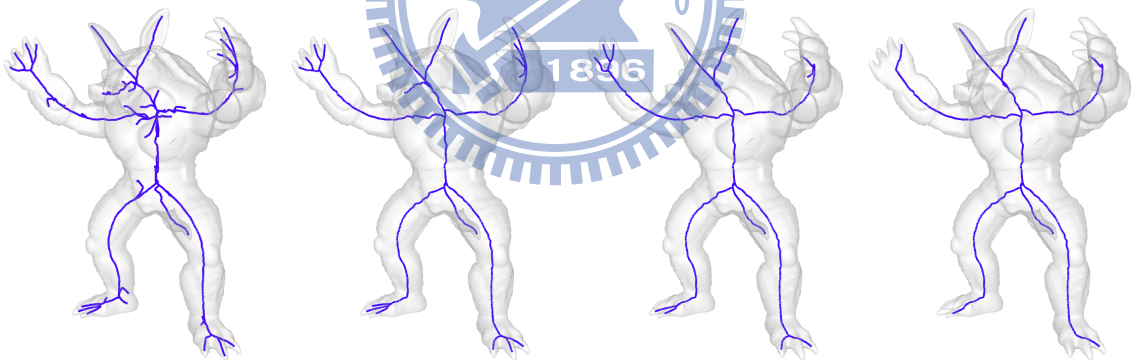
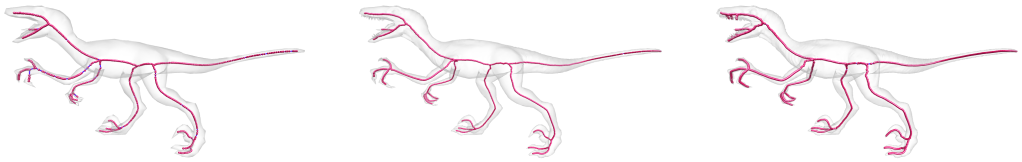


Figure 5.12: Dense skeleton-surface mapping on the raptor models with polygon count 2,500 (top) and 20,000 (bottom).



(a) Salience = 10 (b) Salience = 200 (c) Salience = 800 (d) Salience = 1,500

Figure 5.13: Skeletons of decreasing levels of details for the armadillo model.



(a) 2,500 polygons (706 nodes) (b) 10,000 polygons (2,345 nodes) (c) 20,000 polygons (3,781 nodes)

Figure 5.14: Skeleton of raptor model in different resolutions (salience threshold 500). The number inside the parentheses is the number of skeleton nodes.

CHAPTER 6

# User-Controllable Mesh Simplification

## 6.1 Introduction

In addition to the intermediate-level surface properties and its applications to mesh processing, we are also interested in how to employ semantic meaning of the object's feature in the geometric processing. The semantic meaning of the object's feature is important for describing the visual significance of shape feature, but cannot be described by the the existing surface properties. Hence, a user-assisted strategy is usually required to employing the semantic meaning of the shape feature in the geometric modeling process. To adapt a user-assisted scheme to the geometric modeling process, one crucial issue is how to quantize the semantic meaning of the shape feature into a value that can be effectively recognized by the applications. In this chapter, we focus on how to employ the semantic meaning of the shape feature in the mesh simplification application and propose a user-controllable scheme to assist users to attain a satisfactory resolution for regions of semantic importance.

In the past two decades, lots of mesh simplification metrics have been proposed, which consider either the geometric difference [30, 20], texture deviation [14], or visual difference [49]. Each of these metrics has its own strength and weakness in preserving geometric and texture features. However, all of these metrics do not take semantic or functional features into account. As a result, practitioners have found that these metrics are not able to produce satisfactory result when the simplified mesh of very low-polygon count are ex-

pected.

To overcome such limitations, the concept of *user-assisted or user-guided simplification* become attractive. One way to this end is to perform refinement or simplification on the simplification hierarchy [13, 47, 33]. Such setup is usually constrained by the vertex-split dependence problems. Another approach reorders the primitive collapsing by weighting the collapsing cost [38, 64]. Since the collapsing cost cannot be described by a simple function, the weights applied have no direct relation to the result of the refinement. In consequence, the weights are usually chosen in a trial and error basis. Moreover, the weights that are appropriate to a simplified mesh derived by an error metric may not be appropriate to the one derived by another error metric.

Our goal is a user-controllable simplification framework that allows users to improve the quality of simplified meshes derived by using any existing error metric, such as QEM [20] or APS [14]. The framework consists of two stages. The first stage employs weighting schemes that allow users to refine unsatisfactory regions and achieve user-expected resolutions. The second stage is a local refinement scheme that utilizes vertex splits performed on the vertex hierarchy [31], aiming to provide a user-guided fine-tune for recovering sharp features. To achieve the goal of user-controllable, we define the weight in the first stage to be the resolution improvement of the surface region. Then, a reorder scheme is used that the simplification operations in the weighted regions are postponed until the resolution increments are met. Two weighting schemes are proposed, namely uniform weighting and nonuniform weighting. In uniform weighting scheme, a weight value is applied to all original mesh vertices in a selected region, resulting in a uniform improvement on vertex resolution in the region. On the other hand, in the nonuniform weighting scheme varying weights are applied to vertices in a selected region and obtain a nonuniform resolution improvement in the region. The proposed weighting schemes differ from the previous approaches [38, 64] in that the proposed weighting schemes reorder the simplification sequence directly rather than by changing the simplification cost and then reordering the sequence indirectly. The proposed reordering mechanisms are designed to achieve the following goals:

- The resolution improvement for a given weighting value in a selected region is predictable.

- The weighting schemes are completely independent of the error metric used, that is,

same resolution improvement for a weighting value is obtained no matter which error metric is used.

- A weighting value will result in the same resolution improvement when it is applied to simplified meshes in different resolutions.

## 6.2 User-Controllable Mesh Simplification

The proposed user-controllable simplification framework allows users to achieve a predictable quality improvement in selected regions for a simplified mesh derived by any existing error metric, such as QEM [20] or APS [14]. The framework consists of two stages. The first stage employs the proposed weighting schemes that allow users to refine the unsatisfactory regions to the user-expected resolutions. The second stage is a local refinement based on the vertex hierarchy [31], aiming to provide a user-guided fine tune to recover sharp features via vertex splits. Fig. 6.1 depicts an overview of the framework. At the start-up, we construct a progressive mesh (PM) sequence for the input mesh using an automatic mesh simplification algorithm, such as QEM [20] or APS [14]. If the quality of the simplified mesh is not satisfactory, users can mark the unsatisfactory regions on the original mesh, assign weights to the vertices inside the regions, and apply the weighting scheme to increase the vertex resolution in the selected regions. The weighting scheme can be iteratively applied until the satisfactory result is obtained. The vertex hierarchy is then built according to the reordered PM sequence. Finally, if necessary, users can refine the local features such as sharp edges or corners by iteratively performing vertex splits.

Two weighting schemes are proposed. In the first scheme users assign a constant weighting value to all the vertices in a selected region marked on the original mesh. Each edge collapse associated with the vertex in the selected region are then reordered by comparing it to other edge collapses in PM sequence in such a way that the resulting resolution in the region is about a multiple of the number of the vertices in the region defined by the weighting value. Second scheme allows the user to specify a weighting value to each vertex independently in a unsatisfactory region. The reordering of the edge collapse associated with the vertex is compared only to its dependent edge-collapses in the PM sequence. Such a scheme is applied in per-vertex basis and hence provides a nonuniform weighting effect in the region if the varying value is applied to vertices in the region via surface painting

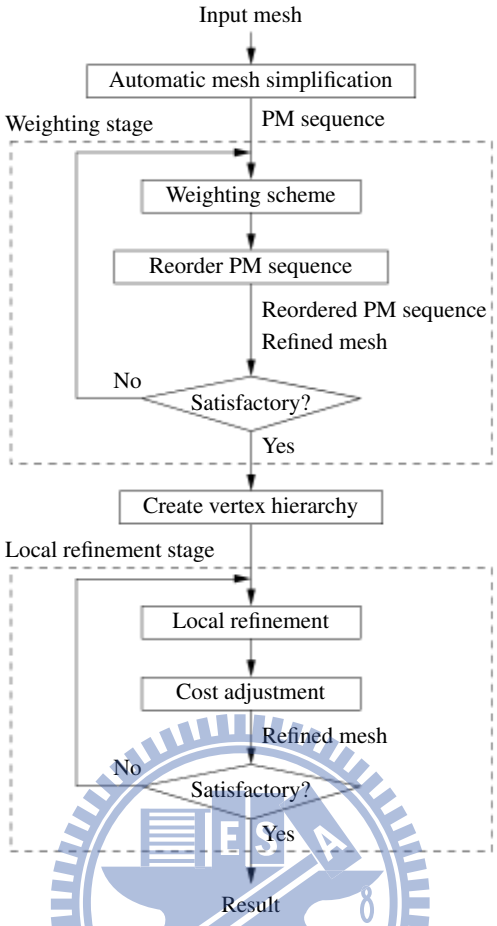



Figure 6.1: System overview.

system. After the reordering of edge collapses is completed, the input mesh is simplified according to the new order to a mesh that has same polygon count as before.

As stated in Section 1, the weighting scheme proposed here reorder the collapsing sequence directly, rather than indirectly via the weighting of the collapsing cost as in [38, 64]. Such a direct reordering mechanism can ensure a predictable improvement of vertex resolution in the selected region, normally by an increase as a multiple of the number of vertices in the region. This effect is usually impossible to be achieved by using previous methods. Moreover, the proposed schemes are quite unique in its capability to be both error-metric and resolution independent. That is, same resolution improvement in the selected region will be obtained for a particular weighting value no matter which error metric is used or whatever the resolution is for the simplified mesh.

Each of the two stages has its own strength and weakness. The weighting scheme reorders the edge collapsing sequence and may greatly alter the simplification result. As a result, the weighting scheme is more effective in overall refinement over the selected region,

but can be hardly used to fine tune the local features. On the other hand, the local refinement is restricted by the existing vertex hierarchy; but is effective in performing refinement over local areas to recover sharp features. In the meantime, the local refinement has relatively more control on where to get polygon budget, and hence can be applied to models with low polygon count.

Both the nonuniform weighting scheme and local refinement are based on the vertex hierarchy but with different goals and mechanisms. The nonuniform weighting scheme reorders collapsing order for the edge collapse associated with a vertex according to its relation to its designated ancestor in the vertex hierarchy while the local refinement, however, refines the mesh around a vertex by splitting the vertex; that is, moving up the active cut own the vertex hierarchy.

## 6.3 Uniform Weighting Scheme

We consider the weighting value as a *multiple value* for the expected increase on the vertex resolution in a selected region. That is, given a user-specified weighting value $\omega$ and a selected region containing $n$ vertices in the simplified mesh, all the edge collapses in the selected region will be delayed such that approximately $\omega \times n$ vertices will be preserved in the region while maintaining the same total polygon count for the simplified mesh.

Before getting into the detailed reordering scheme, we first define the *order* of an edge collapse. Consider a complete progressive mesh sequence (PM sequence) for simplifying a given original mesh to a vertex, the order of an edge collapse is its order in the PM sequence. For all edge collapses in the selected region, we enumerate them from back to front in the complete PM sequence and in the meantime define the the *rank* of the edge collapse according to the enumeration. To make the reordering computation clean, we enumerate starting from 0, that is, the rank of the last edge collapse in the selected region is 0, the last second is 1, and so on.

Consider a selected region $R$ specified on the original mesh and a user-specified weighting value $\omega$ assigned to the vertex $v$ in $R$. Let $e$ be the edge that is collapsed to $v$, and $r$ and $o$ be the rank and order of edge collapse $e$, respectively. The new rank $\tilde{r}$ of edge collapse $e$ is computed by

$$\tilde{r} = \frac{r}{\omega}. \tag{6.1}$$

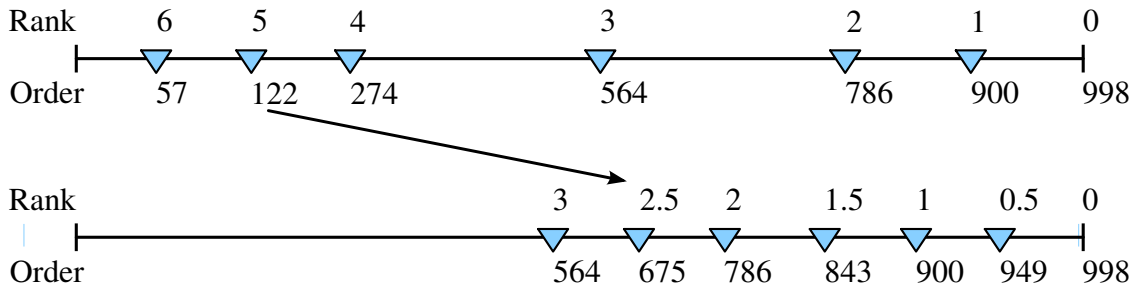



Figure 6.2: Reordering edge collapses in the selected region after applying weighting 2 (uniform weighting scheme).

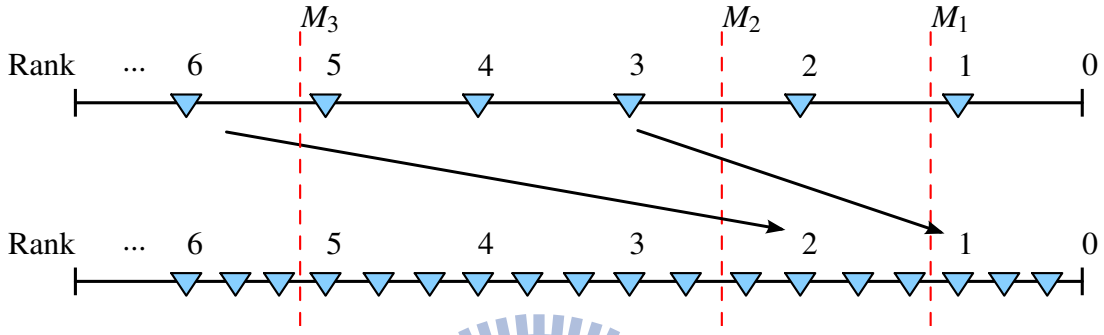



Figure 6.3: The effect of applying a weighting value to the selected regions with different resolutions (uniform weighting scheme).

The new collapsing order $\tilde{o}$ of edge collapse $e$ is obtained by the linear interpolation between $o_i$ and $o_{i+1}$, where $r_i < \tilde{r} \leq r_{i+1}$. That is, for the edge collapse $e$ having new rank $\tilde{r}$, we first find $o_i$ and $o_{i+1}$ such that $r_i < \tilde{r} \leq r_{i+1}$, and then perform the following linear interpolation:

$$\tilde{o} \quad = \quad (\tilde{r} - r_i) \times o_{i+1} + (r_{i+1} - \tilde{r}) \times o_i. \tag{6.2}$$

Let's illustrate the reordering process using the example shown in Fig. 6.2, where the triangle dots on the top horizontal line indicate edge collapses in the selected region and their ranks and orders before the weighting value 2 is applied, while the triangle dots on the bottom horizontal line represent reordered edge collapses and their new ranks and orders. The edge collapse with rank 5 is assigned a new rank $2.5\,(=5/2)$, and its new order 675 is the result of a linear interpolation between the orders of edge collapses whose ranks are 3 and 2 before weighting. As a result of the reordering, the first of these six edge collapses is reordered to a place where the fourth edge collapse most likely lies. Since the weighting scheme doesn't take the collapsing cost into account, it is apparent that its effectiveness is independent of the error metric employed.

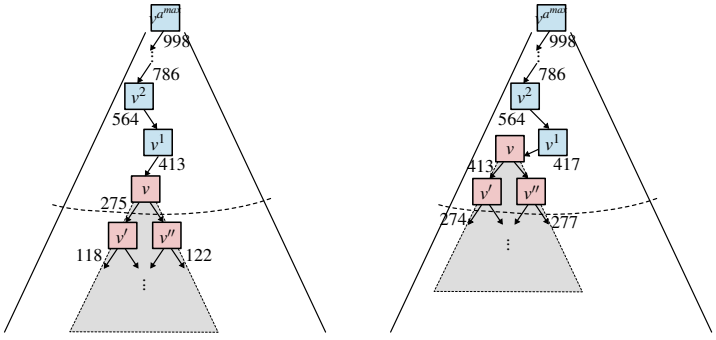
Since the proposed weighting scheme determines the new order for an edge collapse according to where its new rank lies in the original PM sequence, its effectiveness is applied

to whole PM sequence. Hence the effectiveness of a particular weighting value works for simplified meshes of different resolution. Take the example shown in Fig. 6.3, where weighting value is 3 and $M_1$, $M_2$, and $M_3$ represent the termination points of simplified meshes of three different resolutions. We can see that there are one edge collapse remains before the collapsing terminates for $M_1$. After applying weighting value 3, the number of edge collapses remain becomes 3. Similar results are observed for $M_2$ and $M_3$.

## 6.4 Nonuniform Weighting Scheme

In the uniform weighting scheme, the orders of all edge collapses in the selected region are delayed with the same amount of gap in the PM sequence such that an expected resolution increase defined by the weighting value can be achieved. In this weighting scheme, a consistent weighting value is applied to all vertices in the selected region. Such a setting may limit the flexibility that the designers expect to have; for example designers may expect to have resolution increase by varying orders for vertices within the unsatisfactory region.

We next propose a weighting scheme in which a weighting value is assigned to each individual vertex in the original mesh and then an increase in vertex resolution indicated by the weighting value will be obtained around the vertex. Thus for a selected region on the original mesh, different weighting values can be applied to vertices in the region by using a surface painting system and, as a result, varying resolution increases will be achieved within the region.




(a) Before weighting (b) After apply weighting value of 2

Figure 6.4: Effect of the nonuniform weighting scheme.

To this end, we formulate the weighting schedule based on the vertex hierarchy formed by the PM sequence; as shown in Fig 6.4. Consider the edge $e$ that collapses to the vertex $v$.

If the collapsing order of $e$ is delayed to that of its parent, our goal is to obtain two vertices replacing $v$. Similarly, four vertices is expected to be obtained if the order of $e$ is delayed to that of its grandparent. Based on the aforementioned observation, when the collapsing order of the edge $e$ associated with a vertex $v$ is delayed to that of $v$'s $i$-th ancestor on the vertex hierarchy, our goal is to obtain $2^i$ vertices to replace $v$. Consider an edge $e$ and its collapsed vertex $v$. Suppose $\omega$ is the weight assigned to the vertex $v$, indicating the number of vertices expected to replace $v$ in current level. We first find $i$ such that $2^i < \omega \leq 2^{i+1}$ and then compute the target collapsing order of $e$, denoted by $\tilde{o}$, by linearly interpolating the collapsing orders of $v$'s $i$-th and $(i+1)$-th ancestors, respectively, as follows,

$$\tilde{o} = \alpha\, o_{i+1} + \beta\, o_i, \tag{6.3}$$

where $o_i$ and $o_{i+1}$ are edge-collapse orders of $v$'s $i$-th and $(i+1)$-th ancestors, respectively, and

$$\alpha = \frac{\omega - 2^i}{2^{i+1} - 2^i},$$

and

$$\beta = \frac{2^{i+1} - \omega}{2^{i+1} - 2^i}.$$

The weighting value $\omega$ should be bounded since the highest ancestor for a vertex $v$ is the root of vertex hierarchy. Let's denote the root as the $a^{max}$-th ancestor of the vertex $v$. The weighting value $\omega$ assigned to $v$ should be bounded by $2^{a^{max}}$; that is, $\omega$ should be clamped to $\min\left(\omega, 2^{a^{max}}\right)$.

So far we have described how to reorder the collapsing order of the edge associated with a vertex in the nonuniform weighting scheme. To make it really works, by that we mean $2^i$ vertices is obtained to replace the vertex $v$ if the collapsing order of the edge $e$ associated with $v$ is delayed to that of $v$'s $i$-th ancestor on the vertex hierarchy, we need also to assign the same weighting value to the descendants of $v$, at least down to level of $2^{i-1}$ on the vertex hierarchy. In our interface, after examining the simplified mesh users assign varying weighting value to vertices in the selected regions on the original mesh by using a surface painting tool. For a vertex inside the selected regions, its descendants are around (some of them may be outside the region), and therefore are likely to be assigned with some weighting values. Due to this interface design, as we will see in the result section, the nonuniform weighting scheme may not always achieve the expected resolution increment.

## 6.5 Local Refinement

The weighting scheme, including previous methods, generally cannot recover sharp features, such as sharp edges and corners. The second stage of our user-controllable simplification framework is a local refinement scheme aiming to provide an effective tool for recovering local sharp features. The proposed refinement operation is similar to the selective refinement and simplification in view-dependent level-of-detail modeling [31]. The selective refinement (simplification) refines (simplifies) a mesh by moving down (up) the active cut of the vertex hierarchy.

Given a simplified mesh with its progressive mesh sequence, normally the result of the first stage, the system constructs the corresponding vertex hierarchy with collapsing cost recorded on each vertex and the active cut associated with the given simplified mesh. To do the local refinement, user selects a set of vertices and the system will perform vertex split on these vertices, and in the meantime do the vertex collapsing on some vertices to maintain the polygon count. Those vertices that have the lowest collapsing cost are the candidate vertices for edge collapsing. Note that the vertex split or collapsing are just the moving down or up of the active cut.

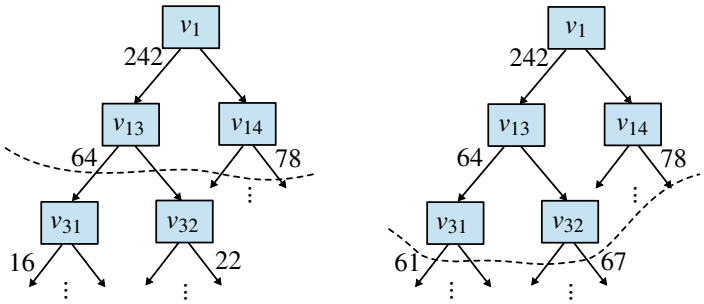
One thing worth mentioning is that the vertex split dependency problem may limit the ability of local refinement since a vertex can be split only if all its neighboring vertices after split are reachable. In our implementation, such problems are overcome by applying the approach proposed in [39]. Another issue needs to be addressed is that, after local refinement, vertices resulting from a vertex split normally have costs lower than their parent. After a sequence of vertex splits applied to a vertex $v$, the subtree originates from $v$ may have leaf vertices whose costs are relatively lower than that of vertices in the active cut. This implies that the split vertices may soon be collapsed when vertices in other region are split. To prevent this problem, we need to adjust the costs of split vertices such that they have about the same magnitude as the cost of $v$. Further, the cost difference for vertices in the subtree should be maintained to preserve the local features.

The cost adjustment is done along with the vertex split operations in the local refinement process. Let $c_v$ be the cost of a vertex $v$ to be split, and $c_1$ and $c_2$ be the costs of the children

of $v$. Suppose $c_1 \geq c_2$, $c_1$ and $c_2$ are adjusted to $c_1^*$ and $c_2^*$ as follows:

$$\begin{aligned} c_1^* &= c_v + \tfrac{c_1 - c_2}{2} \\ c_2^* &= c_v - \tfrac{c_1 - c_2}{2}. \end{aligned} \tag{6.4}$$

Note that Eq. 6.4 ensures that the average cost of the split vertices is the same as their parent and the cost difference between the split vertices is maintained.




(a) Before the vertex split (b) After the vertex split

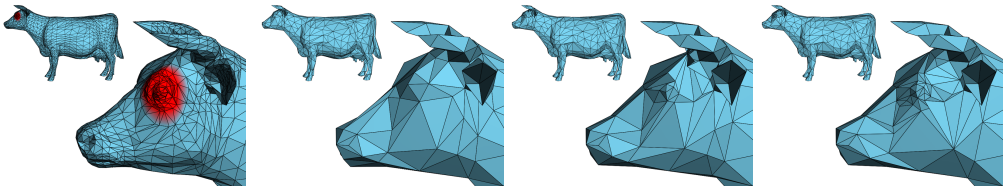
Figure 6.5: Cost adjustment in local refinement process.

As shown in Fig. 6.5, the costs of $v_{31}$ and $v_{32}$ are adjusted after $v_{13}$ is split, and the average cost of $v_{31}$ and $v_{32}$ are the same as their parent $v_{13}$. Moreover, the difference between $v_{31}$ and $v_{32}$ remains the same after local refinement.

## 6.6 Results

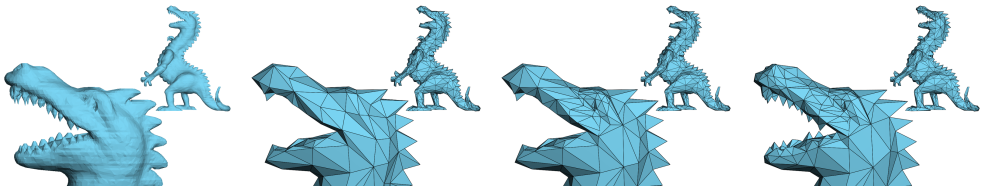
In the implementation, the proposed user-controllable mesh simplification framework supports QEM [20] and APS [14] as the cost measure for the edge collapsing. To preserve the simplification styles of the employed error metric, the simplification after applying weighting is executed in the same way as the automatic simplification process with the error metric, except that the edge collapses associated with the weighted vertices are not performed until the delayed orders are encountered.

Several experimental tests are performed to demonstrate the effectiveness of the proposed weighting schemes. First example is a cow model of 5,804 polygons (Fig. 6.6(a)), which is simplified to a mesh of 1,160 polygons (20% of the original mesh) by using QEM [20] (Fig. 6.6(b)). Different weighting values are applied to the region of left eye using uniform weighting scheme as shown in Fig. 6.6(a) by red color. Fig. 6.6(c) and Fig. 6.6(d) depict simplified result and the refined meshes after applying weighting values



(a) Original mesh (b) Simplified mesh (8 vertices in the selected region) (c) Weight = 2 (18 vertices in the selected region) (d) Weighting = 3 (27 vertices in the selected region)

Figure 6.6: Apply uniform weighting on the cow model using different weighting values.

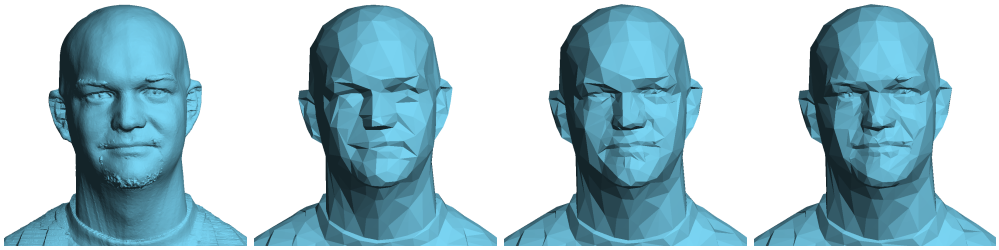


(a) Original mesh (b) Simplified mesh (c) After applying weighting scheme (d) After local refinement

Figure 6.7: Two-stage user-controllable simplification (with uniform weighting) on the dragon model.

2 and 3, respectively. Fig. 6.7 and Fig. 6.8 illustrate the effectiveness of two-stage user-controllable simplification on the dragon model of 50,000 polygons and male model of 151k polygons. Both models are first simplified to meshes of 1,500 polygons using QEM. For the dragon model, the uniform weighting scheme with weight value of 3 is applied to the regions of eyes, with the resultant mesh shown in Fig. 6.7(c). Local refinement is then applied to areas of teeth and nose, producing refined mesh shown in Fig. 6.7(d). For the male model, the uniform weighting scheme with weight value of 3 is applied to regions of eyes, lips, and nose, and local refinement is applied to recover sharp features such as eyeballs, eyebrows, and nose. Table 6.1 and Table 6.2 list the geometry and normal deviations, respectively, before and after the user-controllable simplification for the male model. The errors are measured using MeshDev, which a mesh comparison tool using attribute deviation metric [69]. Although the mean errors after applying user-controllable simplification are slightly increased, the errors are diffused over the regions that are considered perceptually less important. Fig. 6.9 visualizes the distributions of geometry and normal deviations for the male model. Noticeable improvements can be found in the selected regions, and the compensative error introduced by the proposed scheme is almost invisible and diffused over the less-important regions.

Fig. 6.10 illustrates the result of nonuniform weighting scheme and local refinement



(a) Original mesh (b) Simplified mesh (c) After applying weighting scheme (d) After local refinement

Figure 6.8: Two-stage user-controllable simplification (with uniform weighting) on the male model.

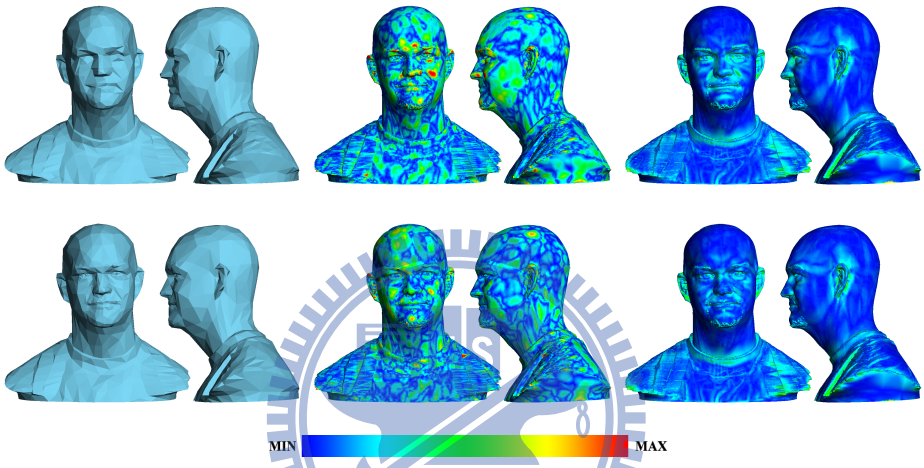



Figure 6.9: Visualization of the error distributions for simplified male model before (top) and after apply user-controllable simplification (bottom). The left column are the shaded models, the middle column shows the distributions of geometry deviation, and the right column visualizes the normal deviation. Both deviations are measured by using MeshDev [69].

applied to the buste model of originally 511k polygons. It is first simplified to 1,500 polygons using QEM. Three different levels of nonuniform weights are applied to the model according to the significance in perception; as shown in Fig. 6.10(c) on which the green, yellow, and red colors represent the weighting value 2, 3, and 4, respectively. Then, the local refinement is applied to the eyes, nose, and lips to recover crease features.

Fig. 6.11 compares the effectiveness of the proposed uniform weighting and nonuniform weighting schemes against the one proposed in [38]. The cow model is simplified to 1,160 polygons (20% of the original mesh) using QEM; as shown in Fig. 6.11(a). Weighting value 3 is assigned to the left eye as the red region shown in Fig. 6.6(a). The proposed uniform and nonuniform weighting schemes yield similar resolution increment for that region, namely increasing from 8 vertices to 27 and 25 vertices, respectively; see Fig. 6.11(b) and Fig. 6.11(c), respectively . The weighting scheme of [38] reorders the edge collapse

Table 6.1: Geometry deviation of the simplified male model before and after applying the user-controllable simplification.

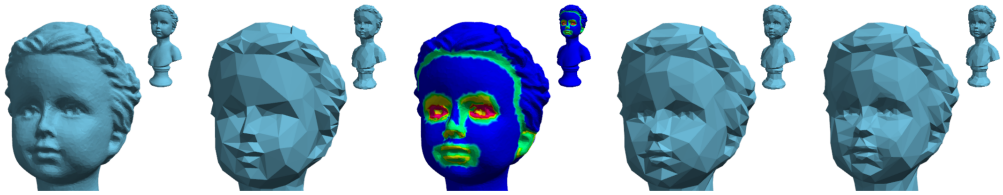
| | Simplified mesh | Simplified mesh after refinement |
|---|---|---|
| Minimum | 2.758e−8 | 3.187e−8 |
| Maximum | 5.045e−3 | 5.008e−3 |
| Mean | 5.131e−4 | 5.460e−4 |
| Variance | 1.880e−7 | 1.963e−7 |

Table 6.2: Normal deviation of the simplified male model before and after applying the user-controllable simplification.

| | Simplified mesh | Simplified mesh after refinement |
|---|---|---|
| Minimum | 7.984e−4 | 8.007e−4 |
| Maximum | 1.925 | 1.948 |
| Mean | 0.222 | 0.227 |
| Variance | 0.04469 | 0.04461 |

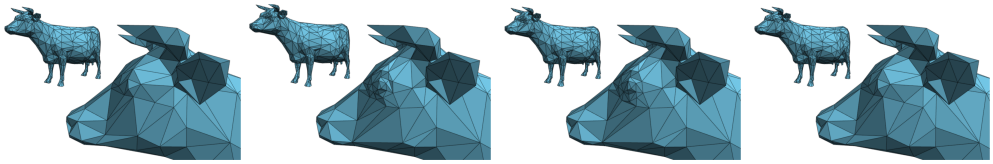
sequence by directly multiplying the weighting values to the corresponding quadric errors. Since modification of quadric error has no direct link to the resolution improvement, the resolution improvement is not predictable. In this test case, the number of vertices remain unchanged; as shown in Fig. 6.11(d). Next, we compare the effectiveness of the proposed uniform weighting, nonuniform weighting schemes, and the weighting scheme in [38] using the buste model. The buste model is first simplified to the meshes of $1,500$ polygons. Uniform weighting with values 2 and 3 is applied to the selected regions as shown in Fig. 6.10. For nonuniform weighting, values similar to that in Fig. 6.10 are applied to the selected regions. As shown in Fig. 6.12, the uniform weighting with value 2 may not preserve the eyes well (Fig. 6.12(b)) while the uniform weighting with value 3 seems over-preserve the eyes (Fig. 6.12(c)). The nonuniform weighting scheme is more capable of adapting to the expectation of users. Again, the weighting scheme of [38] with weighting value of 3 performs badly in this case.

Both of our proposed weighting schemes are independent on the resolution of the given meshes, meaning that similar resolution increment in the selected regions is achieved for the given simplified meshes in different resolutions. Fig. 6.13 depicts the results of applying the uniform weighting with value 3 to the cow models with polygon counts 500, $1,160$, $1,739$, and $2,321$. The selected region is the same as the one in Fig.6.6. The increased number of vertices in the selected region is shown in Table 6.3, which also includes



(a) Original mesh (b) Simplified mesh (c) Nonuniform weighting (d) After applying weighting scheme (e) After local refinement

Figure 6.10: Two-stage user-controllable simplification (with nonuniform weighting) on the Buste model.

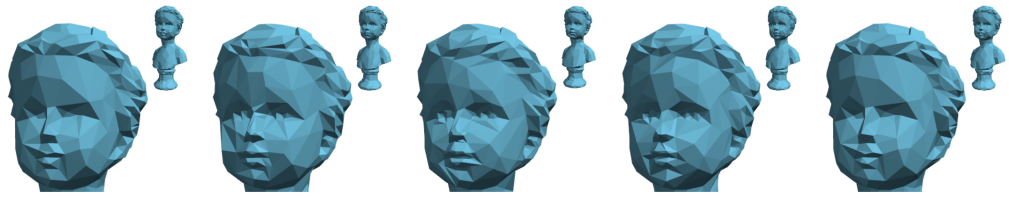


(a) Simplified mesh (8 vertices in the selected region) (b) Uniform weighting (27 vertices in the selected region) (c) Nonuniform weighting (25 vertices in the selected region) (d) Weighting scheme of [38] (8 vertices in the selected region)

Figure 6.11: Comparison of the proposed uniform weighting and nonuniform weighting schemes against the weighting scheme in [38].

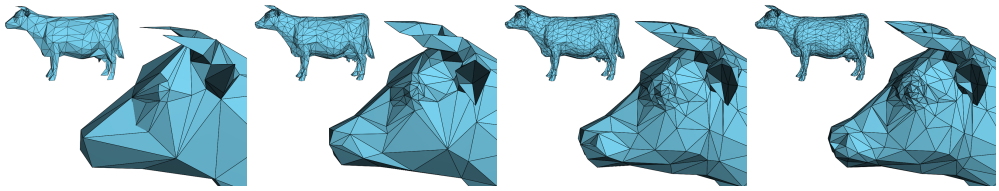
the performance of the proposed nonuniform weighting scheme and the scheme proposed in [38]. The three numbers in each item of the "vertex count in the selected region" indicate the vertex counts resulting from the uniform weighting scheme (top), the nonuniform weighting scheme (middle), and the weighting scheme of [38] (bottom). We observe that the performance of the uniforming and nonuniform weighting scheme is quite close to what we expect. Note that the small inaccuracy in hitting the expected target is due to the dependency problem in the dependency hierarchy that occurs on the boundary of the selected region. On the other hand, the resolution improvement of the weighting scheme of [38] is unpredictable. It is usually hard for users to specify the weight value for an expected resolution improvement.

The proposed weighting schemes are also independent on the error metric used in the mesh simplification. Since APS is a texture-deviation error metric, we consider the Parasaur model with texture mapped. The Parasaur model of 7685 polygons is first simplified to 750 polygons using QEM and APS. Then we apply the uniform weighting scheme with values 2 and 3 to the region of left eye. Table 6.4 shows the resolution increments in the region of left eye after we apply the uniform weighting scheme with values 2 and 3 to the region. As we can see that the obtained resolution increments are close to what we expect for both metrics. The proposed weighting schemes can be applied to models



(a) Simplified mesh (b) Uniform weighting value 2 (c) Uniform with weighting value 4 (d) Nonuniform with weighting (e) Weighting scheme of [38]

Figure 6.12: Comparison of the proposed uniform weighting, nonuniform weighting schemes, and the weighting scheme in [38].



(a) 500 (8.6%) polygons (b) 1,160 (20%) polygons (c) 1,739 (30%) polygons (d) 2,321 (40%) polygons

Figure 6.13: Results of applying the uniform weighting with value 3 to the cow model of resolutions in different resolutions.

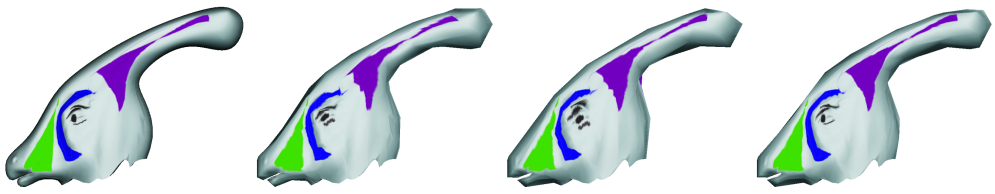
with texture mapped to reduce the texture distortion. Fig. 6.14 shows the result of applying the two-stage user controllable simplification scheme to the Parasaur model with texture mapped. Again, the Parasaur model is simplified to a mesh of 750 polygons using APS, on which noticeable texture distortion can be found; as shown in Fig. 6.14(b). Fig. 6.14(c) and Fig. 6.14(d) depict a great reduction in texture distortion after applying uniform weighting scheme with weighting value of 3 around the left eye and then local refinement on the texture boundaries.

Table 6.3: Comparison on the resolution improvement obtained by the proposed weighting schemes and the weighting scheme of [38].The three numbers in each item of the "vertex count in the selected region" indicate the vertex counts resulting from the uniform weighting scheme (top), the nonuniform weighting scheme (middle), and the weighting scheme of [38] (bottom).

| Polygon count of simplified mesh | Vertex count in the selected region | | |
|---|---|---|---|
| | Without weighting | Weighting value = 2 | Weighting value = 3 |
| 500 (8.6%) | 4 | 8<br>8<br>3 | 13<br>12<br>3 |
| 1,160 (20%) | 8 | 18<br>16<br>8 | 27<br>25<br>8 |
| 1,740 (30%) | 13 | 26<br>27<br>12 | 39<br>42<br>12 |
| 2,320 (40%) | 22 | 44<br>40<br>23 | 67<br>47<br>23 |
| 2,902 (50%) | 28 | 56<br>49<br>30 | 68<br>61<br>30 |

Table 6.4: Resolution improvement after applying constant weighting scheme on different error metrics.

| Error metric | Vertex count in the selected region | | |
|---|---|---|---|
| | W/O weighting | Weighting value = 2 | Weighting value = 3 |
| QEM | 15 | 33 | 49 |
| APS | 6 | 15 | 22 |



(a) Original mesh (b) Simplified mesh (c) After applying uniform weighting scheme (d) After local refinement

Figure 6.14: Applying two-stage user-controllable simplification to the Parasaur model.

CHAPTER 7

# Conclusions and Future Works

## 7.1 Conclusions

In this thesis, we have presented a new slice-based scheme for deriving an intermediate-level surface function that has advantages of low computational complexity and being compact in affected region. Based on the short-cut rule [79], human vision prefers to use the shortest possible cuts to parse silhouettes. We approximated the short cut passing a surface point by a planar slice on the surface, called minimum perimeter slice, that passes the point and has the minimum perimeter. Shape analysis of 3D object can then be performed using a series of 2D slices, hence reducing the computation cost. The surface function defined as the perimeter of the minimum perimeter slice, called minimum slice perimeter (MSP), for all surface points is able to represent the local volume of object around the surface point and possesses a better measure about the local volume information than other intermediate-level surface functions such as SDF [73]. Moreover, the orientation of the minimum perimeter slices on the surface reveals the flow of shape orientation on surface. Such shape properties are useful for describing the object's part level information, which can be useful in geometric modeling applications.

We applied the minimum perimeter slice to the mesh segmentation and skeletonization. For mesh segmentation, a new hierarchical segmentation scheme taking the local volume information into account is presented. The derived segmentation hierarchy possesses some

desired properties. For example, components on a higher level reveal higher degree of salience than their descendant parts and the components on each level of hierarchy have similar degree of salience. Moreover, the number of boundaries on each level of the hierarchy is determined automatically. With the aid of the local volume information provided by MSP, the proposed segmentation scheme can decompose object into several levels of parts in more natural ways, and can be applied to objects in different topological types.

The proposed MSP-driven skeletonization framework first transforms the original mesh to a shrunk skeleton-like mesh using MSP information and then employs a greedy edge-swap framework to degenerate the shrunk mesh into an 1D skeleton. In the greedy framework, edges that deviate farther from the centerline would be swapped first and vertices are moved towards the centerline. Small branches are removed by checking the salience value of their correspondence surface region whenever a branch is formed. The metric designed for the greedy edge-swap framework and salience evaluation are formulated based on the MSP function. The skeleton generated by the proposed method has a dense node distribution at the core parts and around the junctions, and inherently possesses a skeleton-surface mapping. The single salience parameter for branch removal provides a flexible control for deriving skeletons with varying details. Moreover, consistent skeletons can be extracted for a model in different resolutions and poses. We demonstrated the effectiveness of the proposed algorithm by a rather extensive testing and comparison to a state-of-the-art method.

Beyond the intermediate-level shape property and its applications, we also investigated how to interpret and achieve users' expectation when performing mesh simplification. A new concept of the user-controllable mesh simplification scheme is proposed in which the user-specified weights on some selected regions are used to reorder the simplification operator rather than first altering the simplification costs and then reordering the simplification operator. Our scheme results in a predictable resolution improvement over the selected regions and is metric independent since the user-specified weights are not used to alter the simplification cost.

## 7.2 Future works

There exists some limitations to the proposed minimum slice perimeter. Some small highlights of MSP value may be noticed on the surface since the slices may pass across multiple

object parts. A filter that is able to prune those slice segments outside the designate part is required to achieve a MSP that better describes the local volume information.

The existing salience definitions for the 3D object surface [41, 19] are based on the combination of local geometric properties such as curvature and area, which may not reveal the visual significance of object parts. We are also interested in investigating how the salience of a 3D object part can be defined based on the intermediate-level shape properties.

In addition to the derivation of minimum slice perimeter, the normal of the minimum perimeter slices associated with surface points reveals the orientation flow of the object shape, which can be used to guide the vector field generation on the object's surface. We will investigate how to derive a shape-aware vector field on the surface using the orientation vectors provided by the minimum perimeter slices.

Our proposed skeletonization scheme can extract curved skeleton with dense node distribution directly from 3D model. Such dense node distribution allows us to establish a reasonable many-to-one mapping between surface and skeleton. Such surface-skeleton mapping may be helpful in geometric modeling applications such as the surface-to-surface correspondence, mesh deformation, and mesh symmetrization. In the future, we will investigate how these geometric modeling problems can be benefited from the skeleton-surface mapping.

# Bibliography


[1] ALLIEZ, P., COHEN-STEINER, D., DEVILLERS, O., LÉVY, B., AND DESBRUN, M. Anisotropic Polygonal Remeshing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 22*, 3 (2003), 485–493.

[2] AMENTA, N., CHOI, S., AND KOLLURI, R. K. The Power Crust. In *ACM symposium on Solid modeling and applications* (2001), pp. 249–266.

[3] ATTENE, M., BIASOTTI, S., AND SPAGNUOLO, M. Shape understanding by contour-driven retiling. *The Visual Computer 19*, 2 (2003), 127–138.

[4] ATTENE, M., FALCIDIENO, B., AND SPAGNUOLO, M. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer 22*, 3 (2006), 181–193.

[5] ATTENE, M., KATZ, S., MORTARA, M. D., PATANÉ, G., SPAGNUOLO, M., AND TAL, A. Mesh segmentation-A comparative study. In *IEEE International Conference on Shape Modeling and Applications* (2006), pp. 7–18.

[6] AU, O. K.-C., TAI, C.-L., CHU, H.-K., COHEN-OR, D., AND LEE, T.-Y. Skeleton Extraction by Mesh Contraction. *ACM Transactions on Graphics (Proc. SIGGRAPH) 27*, 3 (2008), 1–10.

[7] AU, O.-K.-C., TAI, C.-L., COHEN-OR, D., ZHENG, Y., AND FU, H. Electors Voting for Fast Automatic Shape Correspondence. *Computer Graphics Forum (Proc. Eurographics) 29*, 2 (2010), 645–654.

[8] AUJAY, G., HÉTROY, F., LAZARUS, F., AND DEPRAZ, C. Harmonic Skeleton for Realistic Character Animation. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2007), pp. 160–169.

[9] BESL, P. J., AND MCKAY, N. D. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2 (1992), 239–256.

[10] CHEN, X., GOLOVINSKIY, A., AND FUNKHOUSER, T. A Benchmark for 3D Mesh Segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 3 (2009), 1–12.

[11] CHOI, H. I., CHOI, S. W., AND MOON, H. P. Mathematical Theory of Medial Axis Transform. *Pacific Journal of Mathematics 181*, 1 (1997), 57–88.

[12] CHUANG, J.-H., TSAI, C.-H., AND KO, M.-C. Skeletonization of Three-Dimensional Object Using Generalized Potential Field. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 11 (2000), 1241–1251.

[13] CIGNONI, P., MONTANI, C., ROCCHINI, C., AND SCOPIGNO, R. Zeta: A Resolution Modeling System. *Graphical Models and Image Processing 60*, 5 (1998), 305–329.

[14] COHEN, J., OLANO, M., AND MANOCHA, D. Appearance-Preserving Simplification. In *ACM SIGGRAPH* (1998), pp. 115–122.

[15] CORNEA, N. D., SILVER, D., AND MIN, P. Curve-Skeleton Properties, Applications, and Algorithms. *IEEE Transactions on Visualization and Computer Graphics 13*, 3 (2007), 530–548.

[16] DE FLORIANI, L., MAGILLO, P., AND PUPPO, E. Efficient implementation of multi-triangulations. In *IEEE Visualization* (1998), vol. 98, pp. 43–50.

[17] DEY, T. K., AND SUN, J. Defining and Computing Curve-Skeletons with Medial Geodesic Function. In *Eurographics Symposium on Geometry Processing* (2006), pp. 152–161.

[18] DEY, T. K., AND ZHAO, W. Approximating the Medial Axis from the Voronoi Diagram with a Convergence Guarantee. *Algorithmica 38*, 1 (2003), 179–200.

[19] GAL, R., AND COHEN-OR, D. Salient Geometric Features for Partial Shape Matching and Similarity. *ACM Transactions on Graphics 25*, 1 (2006), 130–150.

[20] GARLAND, M., AND HECKBERT, P. S. Surface Simplification Using Quadric Error Metrics. In *ACM SIGGRAPH* (1997), pp. 209–216.

[21] GARLAND, M., WILLMOTT, A., AND HECKBERT, P. S. Hierarchical Face Clustering on Polygonal Surfaces. In *Symposium on Interactive 3D Graphics* (2001), pp. 49–58.

[22] GOLOVINSKIY, A., AND FUNKHOUSER, T. Randomized Cuts for 3D Mesh Analysis. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA) 27*, 5 (2008), 1.

[23] GOLOVINSKIY, A., AND FUNKHOUSER, T. Consistent segmentation of 3D models. *Computers & Graphics 33*, 3 (2009), 262–269.

[24] GONZÁLEZ, C., GUMBAU, J., CHOVER, M., RAMOS, F., AND QUIRÓS, R. User-assisted simplification method for triangle meshes preserving boundaries. *Computer-Aided Design 41*, 12 (2009), 1095–1106.

[25] HAMANN, B. A data reduction scheme for triangulated surfaces. *Computer Aided Geometric Design*, 2 (1994), 197–214.

[26] HASSOUNA, M. S., AND FARAG, A. A. Robust Centerline Extraction Framework Using Level Sets. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2005), pp. 458–465.

[27] HILAGA, M., SHINAGAWA, Y., KOHMURA, T., AND KUNII, T. L. Topology Matching for Fully Automatic Similarity Etimation of 3D Shapes. In *ACM SIGGRAPH* (2001), pp. 203–212.

[28] HOFFMAN, D. D., AND RICHARDS, W. Parts of recognition. *Cognition 18*, 1-3 (1984), 65–96.

[29] HOFFMAN, D. D., AND SINGH, M. Salience of visual parts. *Cognition 63*, 1 (1997), 29–78.

[30] HOPPE, H. Progressive Meshes. In *ACM SIGGRAPH* (1996), pp. 99–108.

[31] HOPPE, H. View-Dependent Refinement of Progressive Meshes. In *ACM SIGGRAPH* (1997), pp. 189–198.

[32] HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. Mesh Optimization. In *ACM SIGGRAPH* (1993), pp. 19–26.

[33] HUSSAIN, M., OKADA, Y., AND NIIJIMA, K. User-Controlled Simplification of Polygonal Models. In *International Conference on 3D Data Processing, Visualization and Transmission* (2004), pp. 918–925.

[34] KALOGERAKIS, E., HERTZMANN, A., AND SINGH, K. Learning 3D Mesh Segmentation and Labeling. *ACM Transactions on Graphics (Proc. SIGGRAPH) 29*, 4 (2010), 1–12.

[35] KATZ, S., LEIFMAN, G., AND TAL, A. Mesh segmentation using feature point and core extraction. *The Visual Computer 21*, 8 (2005), 649–658.

[36] KATZ, S., AND TAL, A. Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts. *ACM Transactions on Graphics (Proc. SIGGRAPH) 22*, 3 (2003), 954–961.

[37] KAZHDAN, M., CHAZELLE, B., DOBKIN, D., FUNKHOUSER, T., AND RUSINKIEWICZ, S. A Reflective Symmetry Descriptor for 3D Models. *Algorithmica 38*, 1 (2003), 201–225.

[38] KHO, Y., AND GARLAND, M. User-Guided Simplification. In *Symposium on Interactive 3D Graphics* (2003), pp. 123 – 126.

[39] KIM, J., AND LEE, S. Truly Selective Refinement of Progressive Meshes. In *Graphics Interface* (2001), pp. 101–110.

[40] LAI, Y.-K., ZHOU, Q.-Y., HU, S.-M., AND MARTIN, R. R. Feature Sensitive Mesh Segmentation. In *ACM Symposium on Solid and Physical Modeling* (2006), pp. 17–25.

[41] LEE, C. H., VARSHNEY, A., AND JACOBS, D. W. Mesh Saliency. *ACM Transactions on Graphics (Proc. SIGGRAPH) 24*, 3 (2005), 666.

[42] LEE, T.-C., KASHYAP, R. L., AND CHU, C.-N. Building Skeleton Models via 3-D Medial Surface/Axis Thinning Algorithms. *Graphical Models and Image Processing 56*, 6 (1994), 462–478.

[43] LEE, T.-Y., LIN, C.-H., WANG, Y.-S., AND CHEN, T.-G. Animation Key-Frame Extraction and Simplification Using Deformation Analysis. *IEEE Transactions on Circuits and Systems for Video Technology 18*, 4 (2008), 478–486.

[44] LEE, T.-Y., WANG, Y.-S., AND CHEN, T.-G. Segmenting a Deforming Mesh into Near-Rigid Components. *The Visual Computer 22*, 9-11 (2006), 729–739.

[45] LEE, Y., LEE, S., SHAMIR, A., COHEN-OR, D., AND SEIDEL, H.-P. Mesh scissoring with minima rule and part salience. *Computer Aided Geometric Design 22*, 5 (2005), 444–465.

[46] LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. Least Squares Conformal Maps for Automatic Texture Atlas Generation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 21*, 3 (2002), 362–371.

[47] LI, G., AND WATSON, B. Semiautomatic Simplification. In *Symposium on Interactive 3D Graphics* (2001), pp. 43–48.

[48] LI, X., WOON, T. W., TAN, T. S., AND HUANG, Z. Decomposing Polygon Meshes for Interactive Applications. In *Symposium on Interactive 3D Graphics* (2001), pp. 35–42.

[49] LINDSTROM, P., AND TURK, G. Image-Driven Simplification. *ACM Transactions on Graphics 19*, 3 (2000), 204–241.

[50] LIPMAN, Y., CHEN, X., DAUBECHIES, I., AND FUNKHOUSER, T. Symmetry Factored Embedding and Distance. *ACM Transactions on Graphics (Proc. SIGGRAPH) 29*, 4 (2010), 103–114.

[51] LIU, R., AND ZHANG, H. Segmentation of 3D Meshes Through Spectral Clustering. In *Pacific Graphics* (2004), pp. 298–305.

[52] LIU, R., AND ZHANG, H. Mesh Segmentation via Spectral Embedding and Contour Analysis. *Computer Graphics Forum (Proc. Eurographics) 26*, 3 (2007), 385–394.

[53] LIU, R., ZHANG, H., SHAMIR, A., AND COHEN-OR, D. A Part-aware Surface Metric for Shape Analysis. *Computer Graphics Forum (Proc. Eurographics) 28*, 2 (2009), 397–406.

[54] LLOYD, S. P. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory 28*, 2 (1982), 129–137.

[55] MA, C.-M., WAN, S.-Y., AND LEE, J.-D. Three-Dimensional Topology Preserving Reduction on the 4-Subfields. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 12 (2002), 1594–1605.

[56] MANGAN, A. P., AND WHITAKER, R. T. Partitioning 3D Surface Meshes Using Watershed Segmentation. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (1999), 308–321.

[57] MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. H. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. *Visualization and mathematics 3*, 7 (2002), 34–57.

[58] MITRA, N. J., GUIBAS, L. J., AND PAULY, M. Partial and Approximate Symmetry Detection for 3D Geometry. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25*, 3 (2006), 560–568.

[59] MORTARA, M. D., PATANÉ, G., SPAGNUOLO, M., FALCIDIENO, B., AND ROSSIGNAC, J. Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies. In *ACM symposium on Solid Modeling and Applications* (2004), p. 344.

[60] PAGE, D. L., KOSCHAN, A., AND ABIDI, M. A. Perception-based 3D Triangle Mesh Segmentation Using Fast Marching Watersheds. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2003), pp. 27–32.

[61] PALÁGYI, K., AND KUBA, A. A Parallel 3D 12-Subiteration Thinning Algorithm. *Graphical Models and Image Processing 61*, 4 (1999), 199–221.

[62] PASCUCCI, V., SCORZELLI, G., BREMER, P.-T., AND MASCARENHAS, A. Robust On-line Computation of Reeb Graphs: Simplicity and Speed. *ACM Transactions on Graphics (Proc. SIGGRAPH) 26*, 3 (2007), 58.

[63] PODOLAK, J., SHILANE, P., GOLOVINSKIY, A., RUSINKIEWICZ, S., AND FUNKHOUSER, T. A Planar-Reflective Symmetry Transform for 3D Shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25*, 3 (2006), 549–559.

[64] POJAR, E., AND SCHMALSTIEG, D. User-Controlled Creation of Multiresolution Meshes. In *Symposium on Interactive 3D Graphics* (2003), pp. 127–130.

[65] REEB, G. Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. *Comptes Rendus de L'Académie ses Séances 222* (1946), 847—-849.

[66] RENIERS, D., AND TELEA, A. Skeleton-based Hierarchical Shape Segmentation. In *IEEE International Conference on Shape Modeling and Applications* (2007), pp. 179–188.

[67] RENIERS, D., AND TELEA, A. Part-type Segmentation of Articulated Voxel-Shapes using the Junction Rule. *Computer Graphics Forum (Proc. Pacific Gtaphics) 27*, 7 (2008), 1845–1852.

[68] ROSSIGNAC, J., AND BORREL, P. Multi-Resolution 3D Approximations for Rendering Complex Scenes. In *Modeling in Computer Graphics: Methods and Applications*. Springer-Verlag, 1993, pp. 455—-465.

[69] ROY, M., FOUFOU, S., AND TRUCHETET, F. Mesh Comparison using Attribute Deviation Metric. *International Journal of Image and Graphics 4*, 1 (2004), 1–14.

[70] SCHROEDER, W. J., ZARGE, J. A., AND LORENSEN, W. E. Decimation of Triangle Meshes. In *ACM SIGGRAPH* (1992), pp. 65–70.

[71] SHAMIR, A. A survey on Mesh Segmentation Techniques. *Computer Graphics Forum 27*, 6 (2008), 1539–1556.

[72] SHAPIRA, L., SHALOM, S., SHAMIR, A., COHEN-OR, D., AND ZHANG, H. Contextual Part Analogies in 3D Objects. *International Journal of Computer Vision 89*, 2-3 (2009), 309–326.

[73] SHAPIRA, L., SHAMIR, A., AND COHEN-OR, D. Consistent Mesh Partitioning and Skeletonisation using the Shape Diameter Function. *The Visual Computer 24*, 4 (2008), 249–259.

[74] SHARF, A., LEWINER, T., SHAMIR, A., AND KOBBELT, L. On-the-fly Curve-skeleton Computation for 3D Shapes. *Computer Graphics Forum (Proc. Eurographics) 26*, 3 (2007), 323–328.

[75] SHEFFER, A. Model simplification for meshing using face clustering. *Computer-Aided Design 33*, 13 (2001), 925–934.

[76] SHINAGAWA, Y., KUNII, T. L., AND KERGOSIEN, Y. L. Surface Coding Based on Morse Theory. *IEEE Computer Graphics and Applications 11*, 5 (1991), 66–78.

[77] SHLAFMAN, S., TAL, A., AND KATZ, S. Metamorphosis of Polyhedral Surfaces using Decomposition. *Computer Graphics Forum (Proc. Eurographics) 21*, 3 (2002), 219–228.

[78] SIMARI, P., NOWROUZEZAHRAI, D., KALOGERAKIS, E., AND SINGH, K. Multi-objective shape segmentation and labeling. *Computer Graphics Forum 28*, 5 (2009), 1415–1425.

[79] SINGH, M., SEYRANIAN, G. D., AND HOFFMAN, D. D. Parsing silhouettes: The short-cut rule. *Perception and Psychophysics 61*, 4 (1999), 636.

[80] SURAZHSKY, V., SURAZHSKY, T., KIRSANOV, D., GORTLER, S. J., AND HOPPE, H. Fast Exact and Approximate Geodesics on Meshes. *ACM Transactions on Graphics (Proc. SIGGRAPH) 24*, 3 (2005), 553.

[81] TAGLIASACCHI, A., ZHANG, H., AND COHEN-OR, D. Curve Skeleton Extraction from Incomplete Point Cloud. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 3 (2009), 1–9.

[82] TAUBIN, G. Estimating the Tensor of Curvature of A Surface from A Polyhedral Approximation. In *International Conference on Computer Vision* (1995), p. 902.

[83] TIERNY, J., VANDEBORRE, J.-P., AND DAOUDI, M. Enhancing 3D Mesh Topological Skeletons with Discrete Contour Constrictions. *The Visual Computer 24*, 3 (2008), 155–172.

[84] VAN KAICK, O., TAGLIASACCHI, A., SIDI, O., ZHANG, H., COHEN-OR, D., WOLF, L., AND HAMARNEH, G. Prior Knowledge for Part Correspondence. *Computer Graphics Forum (Proc. Eurographics) 30*, 1 (2011), 553–562.

[85] VORONOÏ, G. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire. Sur quelques propriétés des formes quadratiques positives. *Journal für die reine und angewandte 133* (1908).

[86] WILLIAMS, N., LUEBKE, D., COHEN, J. D., KELLEY, M., AND SCHUBERT, B. Perceptually Guided Simplification of Lit, Textured Meshes. In *Symposium on Interactive 3D Graphics* (2003), pp. 113 – 121.

[87] XU, K., ZHANG, H., TAGLIASACCHI, A., LIU, L., LI, G., MENG, M., AND XIONG, Y. Partial Intrinsic Reflectional Symmetry of 3D Shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA) 28*, 5 (2009), 138.

[88] ZHANG, E., MISCHAIKOW, K., AND TURK, G. Feature-Based Surface Parameterization and Texture Mapping. *ACM Transactions on Graphics 24*, 1 (2005), 1–27.