Topological Approaches for 3D Object Processing and Applications

Khaled Tarmissi

A Thesis

in

The Department

of

Electrical and Computer Engineering

Concordia University

Presented in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy at

Concordia University

Montréal, Québec, Canada

March 2010

Your file Votre référence
ISBN: 978-0-494-67369-0
Our file Notre référence
ISBN: 978-0-494-67369-0

Abstract

Topological Approaches for 3D Object Processing and Applications

Khaled Tarmissi

The great challenge in 3D object processing is to devise computationally efficient algorithms for recovering 3D models contaminated by noise and preserving their geometrical structure. The first problem addressed in this thesis is object denoising formulated in the discrete variational framework. We introduce a 3D mesh denoising method based on kernel density estimation. The proposed approach is able to reduce the over-smoothing effect and effectively remove undesirable noise while preserving prominent geometric features of a 3D mesh such as sharp features and fine details. The feasibility of the approach is demonstrated through extensive experiments.

The rest of the thesis is devoted to a joint exploitation of geometry and topology of 3D objects for as parsimonious as possible representation of models and its subsequent application in object modeling, compression, and hashing problems. We introduce a 3D mesh compression technique using the centroidal mesh neighborhood information. The key idea is to apply eigen-decomposition to the mesh umbrella matrix, and then discard the smallest eigenvalues/eigenvectors in order to reduce the dimensionality of the new spectral basis so that most of the energy is concentrated in the low frequency coefficients. We also present a hashing technique for 3D models using spectral graph theory and entropic spanning trees by partitioning a 3D triangle mesh into an ensemble of sub-meshes, and then applying eigen-decomposition to the Laplace-Beltrami matrix of each sub-mesh, followed by computing the hash value of each sub-mesh. Moreover, we introduce several statistical distributions to analyze the topological properties of 3D objects. These probabilistic distributions provide useful information about the way 3D mesh models are connected. Illustrating experiments with synthetic and real data are provided to demonstrate the feasibility and the much improved performance of the proposed approaches in 3D object compression, hashing, and modeling.

Acknowledgements

I am thankful to Allah SWT for giving me the opportunity, wisdom, strength, support, guidance and all blessing upon me to process this research. With a deep sense of gratitude, I wish to express my sincere thanks to my supervisor Professor A. Ben Hamza whose constant guidance, unlimited support and comments made this work possible. Words are not enough to express what he has done to me, I consider myself very lucky to be supervised by a person of such wonderful personality and enthusiasm.

I wish to acknowledge all my committee members for giving me the privilege of having them on my doctoral committee and for their valuable suggestions during my PhD proposal, seminar and this dissertation.

I would like to thank all my friends in Canada who gave me very beautiful memories and made me feel not so far away from home specially: Omar Yagob, Saad Elmansori, Esam Elsheh, Mohamed Hajaje, and Ahmed Alasoad.

Also I would like to thank all the employees in the Libyan Embassy (Libyan Culture Section in Ottawa) for their help and support specially : Esa Elroab, Khaled Zekri, Moftah Najem.

Finally, I wish to express my deepest gratitude to my father, my mother, my sisters, and my wife for their encouragement, support, and love. This achievement is as much theirs as it is mine.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Framework and Motivation

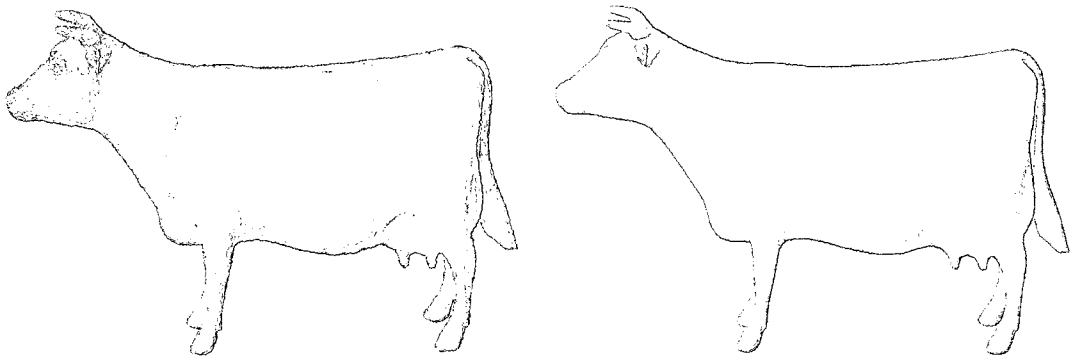
Recent advances in computer technology have contributed to the emergence and the increase of 3D digital data activity. With the increasing use of 3D scanners to create 3D models, 3D object processing, transmission and visualization have become active research fields.

This chapter contains problems definition, motivation, objectives, and a brief review of essential concepts and definitions which we will refer to throughout the thesis, and also presents a short summary of material relevant to 3D mesh processing.

1.1 Problems definition

1.1.1 3D mesh denoising

The great challenge in image processing and computer graphics is to devise computationally efficient and optimal algorithms for recovering images and 3D models contaminated by noise and preserving their geometrical structure. With the increasing use of 3D scanners to create 3D models, there is a rising need for robust and efficient 3D mesh denoising techniques to remove undesirable noise from the data. Even with high-fidelity scanners, the acquired models are invariably noisy, and therefore require filtering. Mesh denoising refers to the process of recovering a 3D model contaminated by noise as shown in Figure 1.1. The challenge of the problem of interest lies in faithfully recovering the original model from the observed model, and furthering the estimation by making use of any prior knowledge/assumptions about the noise process.



Noisy 3D Cow Model Denoised 3D Cow Model

Figure 1.1: Illustration of 3D mesh denoising.

1.1.2 3D mesh compression

Compression of images and shapes has long been the central theme of image processing and computer vision. Its importance is increasing rapidly in the field of computer graphics and multimedia communication because it is difficult to transmit digital information efficiently over the internet without its compression. 3D objects consist of geometric and topological information, and their compressed representation is an important step towards a variety of computer graphics applications including indexing, retrieval, and matching in a database of 3D models.

1.1.3 3D mesh fingerprinting

The increasing use of 3D models in multimedia applications and the wide demand of online services have opened the doors for users to modify the digital content without leaving any perceptual traces. To tackle this problem, cryptographic hash functions could help in ensuring the authentication and the integrity of data. The problem of 3D mesh hashing is relatively new compared to 2D hashing and has received less attention partly because the technology that has been used for image and video analysis cannot be easily adapted to 3D objects.

1.1.4 Statistical 3D mesh distributions

Shape analysis of 3D objects has become an active research field with the recent developments in solid modeling and visualization. Nowadays, vast amounts of 3D models are being developed and are distributed freely or commercially on the internet. 3D graphics are commonly used in several multimedia applications such as video gaming, engineering design, virtual reality, e-commerce and scientific visualization. 3D objects consist of geometric and topological information. Topology is the property that determines which parts of the shape of objects are connected to which other parts, while geometry determines where, in a given coordinate system, each part is located.

1.2 Objectives

A 3D object is usually represented by a triangle mesh which consists of geometric and topological data. In this context geometric models are often acquired by 3D scanning techniques and have to go through post-processing and shape optimization techniques before being actually used in production. Thus, our objectives may be summarized as follows:

- Develop an efficient kernel-based algorithm to remove undesirable noise from the 3D models contaminated by noise while preserving their geometrical and topological structure, and perform an experimental comparative study with the state-of-the-art denoising techniques.
- Devise a novel computationally efficient algorithm to compress the 3D models while preserving their geometrical and topological structure.
- Design a robust fingerprint function that produces a unique identifier for a 3D model to ensure the authentication and the integrity of data.
- Analyze and convert the 3D mesh data into useful and meaningful information which can be used in 3D object processing. To convert the 3D mesh data into information, we need appropriate probabilistic tools and techniques. Such statistical methods will help us quantify the mesh topological properties, and to also gain very useful information by presenting the same data graphically.

1.3 Background

This thesis addresses the application of computational geometry and topology algorithms to three-dimensional surfaces. The following background material is presented to provide context for this work.

1.3.1 Triangle mesh

In computer graphics and geometric-aided design, 3D objects are usually represented as polygonal or triangle meshes. A triangle mesh $\mathbb{M}$ is a triple $\mathbb{M} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, where $\mathcal{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$ is the set of vertices, $\mathcal{E} = \{e_{ij}\}$ is the set of edges, and $\mathcal{T} = \{\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n\}$ is the set of triangles. Each edge $e_{ij} = [\boldsymbol{v}_i, \boldsymbol{v}_j]$ connects a pair of vertices $\{\boldsymbol{v}_i, \boldsymbol{v}_j\}$. Two distinct vertices $\boldsymbol{v}_i, \boldsymbol{v}_j \in \mathcal{V}$ are adjacent (written $\boldsymbol{v}_i \sim \boldsymbol{v}_j$) if they are connected by an edge, i.e. $e_{ij} \in \mathcal{E}$. The neighborhood (also referred to as a ring) of a vertex $\boldsymbol{v}_i$ is the set $\boldsymbol{v}_i^\star = \{\boldsymbol{v}_j \in \mathcal{V} : \boldsymbol{v}_i \sim \boldsymbol{v}_j\}$. The degree d_i of a vertex $\boldsymbol{v}_i$ is simply the cardinality of $\boldsymbol{v}_i^\star$. We denote by $\mathcal{T}(\boldsymbol{v}_i^\star)$ the set of triangles of the ring $\boldsymbol{v}_i^\star$, and by $\boldsymbol{t}_i^\star$ the set of all triangles sharing a vertex or an edge with a triangle $\boldsymbol{t}_i \in \mathcal{T}$ of a mesh $\mathbb{M} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$. Figure 1.2 depicts an example of a neighborhood $\boldsymbol{v}_i^\star$, where the degree of the vertex $\boldsymbol{v}_i$ is $d_i = 6$, and the number of triangles of the set $\mathcal{T}(\boldsymbol{v}_i^\star)$ is also equal to 6. An illustration of $\boldsymbol{t}_i^\star$ is provided in Figure 1.3.

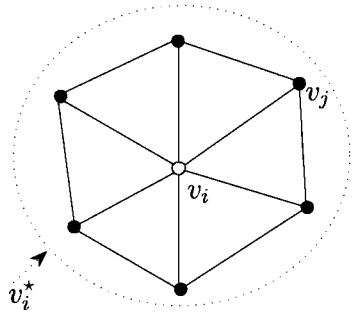


Figure 1.2: Vertex neighborhood $\boldsymbol{v}_i^\star$.

Given a triangle $\boldsymbol{t}_j \in \mathcal{T}$, we denote by area$(\boldsymbol{t}_j)$ and $\boldsymbol{n}(\boldsymbol{t}_j)$ the area and the unit normal of $\boldsymbol{t}_j$ respectively. Consider a triangle $\boldsymbol{t}_j$ with vertices A, B and C, angles α, β and γ and sides a, b and c

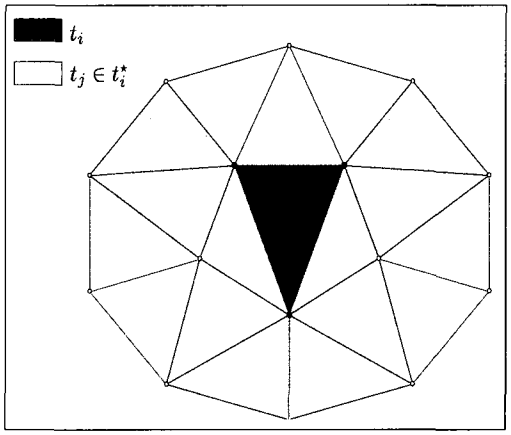


Figure 1.3: triangle neighborhood $\boldsymbol{t}_i^\star$.

as illustrated in Figure 1.4. The triangle normal $\boldsymbol{n}(\boldsymbol{t}_j)$ can be calculated as the vector cross product of two edges of the triangle, and a numerically stable Heron's formula for computing area$(\boldsymbol{t}_j)$ is given by

$$\text{area}(\boldsymbol{t}_j) = \frac{1}{4}\sqrt{(a+(b+c))(a+(b-c))(c+(a-b))(c-(a-b))}, \tag{1}$$

where the length of the sides are arranged such that $a \geq b \geq c$.

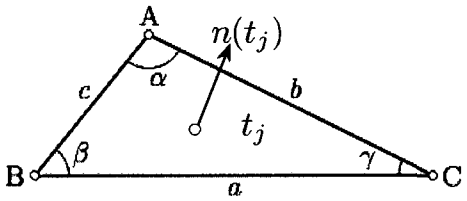


Figure 1.4: Illustration of $A(\boldsymbol{t}_j)$ and $\boldsymbol{n}(\boldsymbol{t}_j)$.

The normal $\boldsymbol{n}_i$ at a vertex $\boldsymbol{v}_i$ is obtained by averaging the normals its neighboring triangles

$$\boldsymbol{n}_i = \frac{1}{d_i}\sum_{\boldsymbol{t}_j \in \mathcal{T}(\boldsymbol{v}_i^\star)} \boldsymbol{n}(\boldsymbol{t}_j). \tag{2}$$

Figure 1.5 depicts the vertex normals of a triangle mesh.

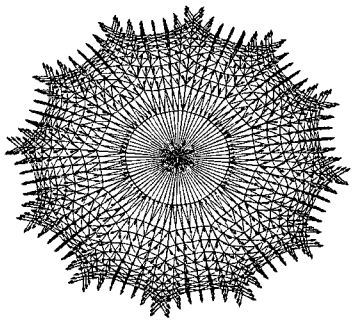


Figure 1.5: Illustration of the vertex normals.

The normal may be defined by weight-averaging the normals of the neighboring triangles (followed by a normalization step)

$$\boldsymbol{n}_i = \sum_{\boldsymbol{t}_j \in \mathcal{T}(\boldsymbol{v}_i^*)} \omega_{ij}\, \boldsymbol{n}(\boldsymbol{t}_j). \tag{3}$$

where ω_{ij} is a normalized weight given e.g. by $1/d_i$, or by the angle formed by the edges of $\boldsymbol{t}_j$ incident to $\boldsymbol{v}_i$, or by the area of each triangle $\boldsymbol{t}_j$.

1.3.2 Edge matrix of a triangle mesh

Given a triangle mesh $\mathbb{M} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, the mean edge length $\bar{\ell}$ of the mesh is given by

$$\bar{\ell} = \frac{1}{|\mathcal{E}|} \sum_{e_{ij} \in \mathcal{E}} \|e_{ij}\|, \tag{4}$$

where $\|e_{ij}\| = \|\boldsymbol{v}_i - \boldsymbol{v}_j\|$.

The edge matrix of a triangle mesh is then given by

$$E = (e_{ij}) = \begin{cases} \|\boldsymbol{v}_i - \boldsymbol{v}_j\| & \text{if } \boldsymbol{v}_i \sim \boldsymbol{v}_j \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Spectral graph theory uses the spectra of matrices associated with the graph, such as the adjacency matrix, the Laplacian matrix, or the normalized Laplacian, to provide information about the graph. One goal is to characterize a graph or obtain information about the graph from the spectra of these matrices.

1.3.3 Normalized Laplacian matrix of a triangle mesh

The Laplacian matrix of a triangle mesh $\mathbb{M} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$ is given by $L = D - A$, where $A = (a_{ij})$ is the adjacency matrix between the vertices, that is $a_{ii} = 0$ and $a_{ij} = 1$ if $\boldsymbol{v}_i \sim \boldsymbol{v}_j$; and $D = \text{diag}\{d_i : i = 1, \ldots, m\}$ is the degree matrix (diagonal matrix whose (i, i) entry is d_i). It is worth pointing out that the number of edges of a triangle mesh $\mathbb{M} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$ is given by $|\mathcal{E}| = \text{trace}(D)/2 = \text{trace}(A^2)/2$. The Euler characteristic is then given by $\chi(\mathbb{M}) = |\mathcal{V}| - |\mathcal{E}| + |\mathcal{T}|$.

The Laplacian matrix is defined as [71]

$$L = (\ell_{ij}) = \begin{cases} d_i & \text{if } \boldsymbol{v}_i = \boldsymbol{v}_j \\ -1 & \text{if } \boldsymbol{v}_i \sim \boldsymbol{v}_j \\ 0 & \text{o.w.} \end{cases} \tag{6}$$

Figure 1.6 and 1.7 illustrate examples of 2D and 3D triangle meshes and their Laplacian matrices, respectively.

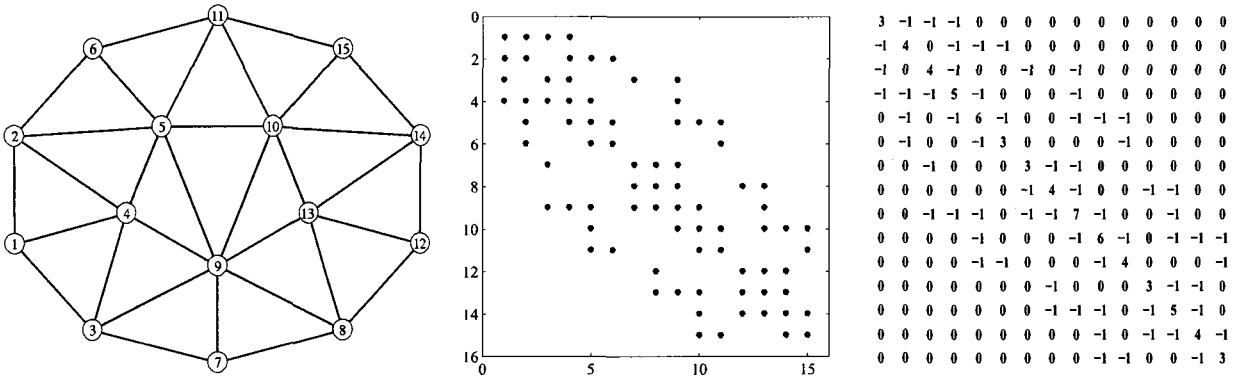


Figure 1.6: 2D triangle mesh and its Laplacian matrix.

The normalized Laplacian matrix $\mathcal{L}$ is given by

$$\mathcal{L} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2} = \begin{cases} 1 & \text{if } \boldsymbol{v}_i = \boldsymbol{v}_j \\ -\frac{1}{\sqrt{d_i d_j}} & \text{if } \boldsymbol{v}_i \sim \boldsymbol{v}_j \\ 0 & \text{o.w.} \end{cases} \tag{7}$$

and may be viewed as an operator defined on the space of functions $\varphi : \mathcal{V} \longrightarrow \mathbb{R}$ as follows

$$\mathcal{L}\varphi(\boldsymbol{v}_i) = \sum_{\boldsymbol{v}_j \in \boldsymbol{v}_i^*} \frac{1}{\sqrt{d_i}} \left(\frac{\varphi(\boldsymbol{v}_i)}{\sqrt{d_j}} - \frac{\varphi(\boldsymbol{v}_j)}{\sqrt{d_i}} \right), \quad \forall \boldsymbol{v}_i \in \mathcal{V}. \tag{8}$$

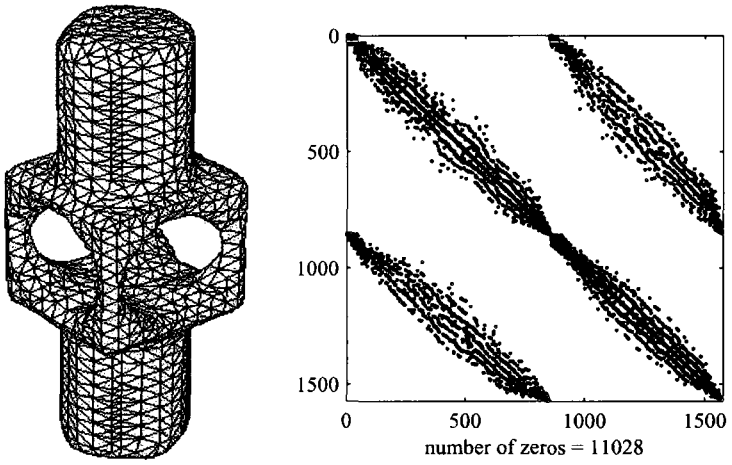


Figure 1.7: 3D triangle mesh and its Laplacian matrix.

The eigenvalue spectrum of the normalized Laplacian matrix allows us to compare the structure of graphs of different sizes.

1.3.4 Vertex differential operators

Given a triangle mesh $\mathbb{M}$, we define the vertex gradient operator $\nabla \boldsymbol{v}_i$ as

$$\nabla \boldsymbol{v}_i = \left\{ \frac{\boldsymbol{v}_j}{\sqrt{d_j}} - \frac{\boldsymbol{v}_i}{\sqrt{d_i}} : \boldsymbol{v}_j \in \boldsymbol{v}_i^\star \right\}. \tag{9}$$

We also define the vertex Laplace operator as

$$\mathrm{div}(\nabla \boldsymbol{v}_i) = \Delta \boldsymbol{v}_i = \sum_{\boldsymbol{v}_j \in \boldsymbol{v}_i^\star} \frac{1}{\sqrt{d_i}} \left(\frac{\boldsymbol{v}_j}{\sqrt{d_j}} - \frac{\boldsymbol{v}_i}{\sqrt{d_i}} \right), \tag{10}$$

where $\mathrm{div}(\cdot)$ is the divergence operator.

1.4 Thesis Overview

This thesis presents research work carried out towards the objectives mentioned above. The entire thesis consists of five chapters. Each of these chapters deals with separate but inherently integrated tasks. The organization of this thesis is as follows:

- In Chapter 2, we propose a 3D mesh denoising technique using Gaussian kernel density estimators in order to reduce the over-smoothing problem and remove the noise effectively while preserving the nonlinear features of the 3D mesh.
- In Chapter3, we propose a 3D mesh compression technique using the centroidal mesh neighborhood information.
- In Chapter 4, we propose a 3D mesh fingerprinting technique that encodes the geometric and topological information of a 3D object into a unique hash value.
- In Chapter 5, we introduce several statistical distributions to analyze the topological properties of 3D objects.
- In Chapter 6, we summarize the contributions in this thesis and propose some future research directions.

CHAPTER 2

Mesh Denoising via Multivariate Kernel Diffusion

2.1 Introduction

Recent advances in computer and information technology have increased the use of 3D models in many fields including medicine, the media, art and entertainment. With the increasing use of 3D scanners to create 3D models, which are usually represented as triangle meshes, there is a rising need for robust mesh denoising techniques to remove inevitable noise in the measurements. Even with high-fidelity scanners, the acquired 3D models are usually contaminated by noise, and therefore a reliable mesh denoising technique is often required.

In recent years, a variety of techniques have been proposed to tackle the 3D mesh denoising problem [7–11]. The most commonly used mesh denoising method is the so-called Laplacian flow which repeatedly and simultaneously adjusts the location of each mesh vertex to the geometric center of its neighboring vertices [7]. Although the Laplacian smoothing flow is simple and fast, it produces, however, the shrinking effect and an oversmoothing result. The most recent mesh denoising techniques include the mean, median, and bilateral filters [12–14] which are all adopted from the image processing literature. Also, a number of anisotropic diffusion methods for triangle meshes and implicit surfaces have been proposed recently. Desbrun *et al.* [15, 16] introduce a weighted Laplacian smoothing technique by choosing new edge weights based on curvature flow

operators. This denoising method avoids the undesirable edge equalization from Laplacian flow and helps to preserve curvature for constant curvature areas. However, re-computing new edge weights after each iteration results in more expensive computational cost. Clarenz *et al.* [17] propose a multiscale surface smoothing method based on the anisotropic curvature evolution problem. By discretizing nonlinear partial differential equations, this method aims to detect and preserve sharp edges by two user defined parameters which are a regularization parameter for filtering out high frequency noisy and a threshold for edge detection. This multiscale method was also extended to the texture mapped surfaces [18] in order to enhance edge type features of the texture maps. Different regularization parameters and edge detection threshold values, however, need to be defined by users onto noisy surfaces and textures respectively before the smoothing process. Bajaj *et al.* [19] present a unified anisotropic diffusion for 3D mesh smoothing by treating discrete surface data as a discretized version of a 2D Riemannian manifold and establishing a partial differential equation (PDE) diffusion model for such a manifold. This method helps enhancing sharp features while filtering out noise by considering 3-ring neighbors of each vertex to achieve non-linear approach of smoothing process. Tasdizen *et al.* [20, 21] introduce a two-step surface smoothing method by solving a set of coupled second-order PDEs on level set surface models. Instead of filtering the positions of points on a mesh, this method operates on the normal map of a surface and manipulates the surface to fit the processed normals. All the surfaces normals are processed by solving second-order equations using implicit surfaces. In [22], Hildebrandt *et al.* present a mesh smoothing method by using a prescribed mean curvature flow for simplicial surfaces. This method develops an improved anisotropic diffusion algorithm by defining a discrete shape operator and principal curvatures of simplicial surfaces. Peng *et al.* [23] have successfully applied locally adaptive Wiener filtering to 3D meshes. Delouille *et al.* [24] proposed wavelet-based approaches to denoise a signal defined on an irregular bivariate grid that represent the denoised data in the wavelet domain by a few scaling coefficients present at the coarsest scale together with the detail coefficients. Another multiscale approach is proposed in Le Faucheur *et al.* [25], where the authors presented a Bayesian shrinkage framework for spherical wavelets with interscale dependency and intrascale smoothing considerations and showed how local consistency can help outperform

uniform shrinkage rules. El Ouafdi *et al.* [26] proposed a stochastic diffusion-based approach for mesh denoising using normalized transition probability and diffusion tensor.

Roughly speaking, mesh denoising techniques can be defined as the requirement to adjust vertex positions without changing the connectivity of the 3D mesh, and may be classified into two main categories: one-step or two-step approaches. The one-step approaches directly update vertex positions using the original vertex coordinates and a neighborhood around the current vertex, and sometimes face normals too. On the other hand, the two-step approaches first adjust face normals and then update vertex positions using some error minimization criterion based on the adjusted normals. In many cases, a single pass of a one-step or two-step approach does not yield a satisfactory result, and therefore iterated operations are performed. In this chapter, we present a 3D mesh denoising method based on kernel density estimation. The proposed technique falls into the category of one-step approaches. The main idea is to use Laplacian smoothing algorithm combined with Gaussian kernel density estimators in order to reduce the over-smoothing problem and remove the noise effectively while preserving the nonlinear features of the 3D mesh such as curved surface regions, sharp edges, and fine details.

The rest of this chapter is organized as follows. In the next section, a general formulation of 3D mesh denoising problem is stated. In Section 2.2, a kernel-based nonlinear diffusion is introduced. In Section 2.3, we provide experimental results to demonstrate a much improved performance of the proposed method in 3D mesh denoising.

2.1.1 Mesh denoising model

In all real applications, measurements are perturbed by noise. In the course of acquiring, transmitting or processing a 3D model for example, the noise-induced degradation often yields a resulting vertex observation model, and the most commonly used is the additive one,

$$\boldsymbol{v} = \boldsymbol{u} + \boldsymbol{\eta}, \tag{1}$$

where the observed vertex $\boldsymbol{v}$ includes the original vertex $\boldsymbol{u}$, and the random noise process $\boldsymbol{\eta}$ which is usually assumed to be Gaussian with zero mean and standard deviation σ.

Mesh smoothing refers to the process of recovering a 3D model contaminated by noise. The challenge of the problem of interest lies in recovering the vertex $\boldsymbol{u}$ from the observed vertex $\boldsymbol{v}$, and furthering the estimation by making use of any prior knowledge/assumptions about the noise process $\boldsymbol{\eta}$, as well as the unknown original mesh.

Generally, 3D mesh denoising methods may be classified into two major categories: isotropic and anisotropic. The former techniques filter the noisy data independently of direction, while the latter methods modify the diffusion equation to make it nonlinear or anisotropic in order to preserve the sharp features of a 3D mesh. Most of these nonlinear methods were inspired by anisotropic-type diffusions in the image processing literature. The diagram shown in Figure 2.1 summarizes the classification of the 3D mesh denoising approaches.

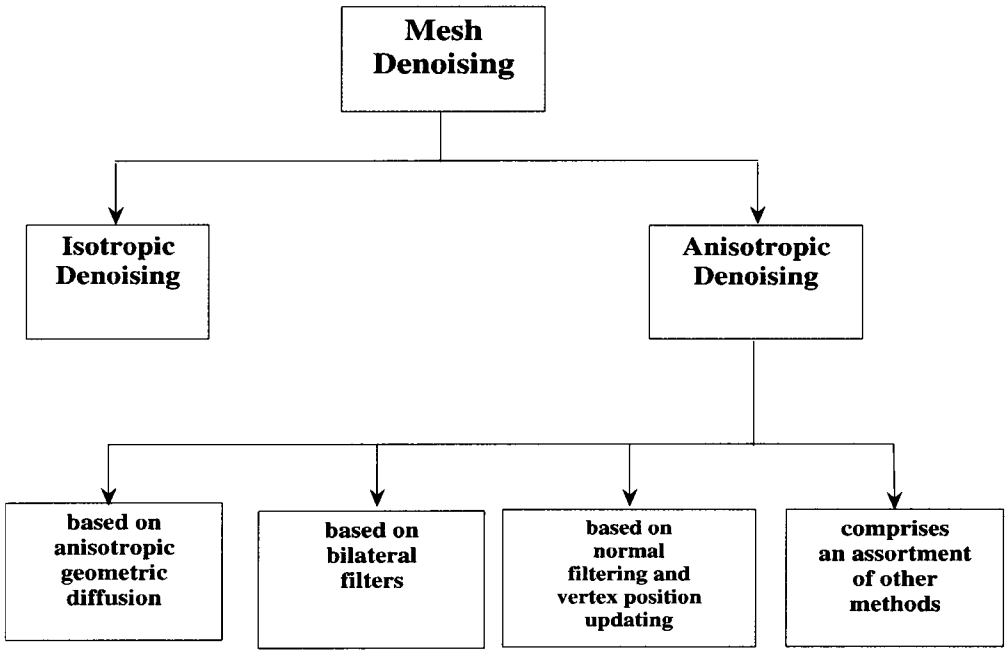


Figure 2.1: Classification of 3D mesh denoising techniques.

2.2 Proposed Mesh Denoising Approach

The proposed method is inspired by the good performance of anisotropic diffusion in image denoising. In [27], we defined a vertex-based anisotropic diffusion as follows

$$\boldsymbol{v}_t = \mathrm{div}(g(|\nabla \boldsymbol{v}|)\nabla \boldsymbol{v}), \tag{2}$$

where g is a redescending function of the vertex gradient magnitude. This function is chosen to allow more smoothing in homogenous regions of 3D mesh, and less smoothing around sharp features. That is, the function g satisfies $g(x) \rightarrow 0$ when $x \rightarrow \infty$ so that the diffusion is "stopped" across sharp details of the mesh. More specifically, the smoothing effect of a vertex-based anisotropic diffusion may be explained as follows: in flat regions of a 3-D mesh where the vertex gradient magnitudes are relatively small, Eq. (2) is reduced to the heat equation which tends to smooth more but the smoothing effect is unnoticeable. And around the sharp features of the 3-D mesh where the vertex gradient magnitudes are large, the diffusion flow given by Eq. (2) tends to smooth less and hence leads to a much better preservation of the mesh geometric structures.

In discrete form, it can be easily shown that the vertex-based anisotropic diffusion may be reduced to the following update rule

$$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \sum_{j\sim i} \varphi(\boldsymbol{v}_i - \boldsymbol{v}_j)\frac{1}{\sqrt{d_i}}\left(\frac{\boldsymbol{v}_j}{\sqrt{d_j}} - \frac{\boldsymbol{v}_i}{\sqrt{d_i}}\right), \tag{3}$$

where the φ-function is defined as $\varphi(\boldsymbol{v}_i - \boldsymbol{v}_j) = g(|\nabla \boldsymbol{v}_i|) + g(|\nabla \boldsymbol{v}_j|)$, and the gradient magnitudes are given by

$$|\nabla \boldsymbol{v}_i| = \left(\sum_{j\sim i} \left\|\boldsymbol{v}_i/\sqrt{d_i} - \boldsymbol{v}_j/\sqrt{d_j}\right\|^2\right)^{1/2}, \tag{4}$$

and

$$|\nabla \boldsymbol{v}_j| = \left(\sum_{k\sim j} \left\|\boldsymbol{v}_j/\sqrt{d_j} - \boldsymbol{v}_k/\sqrt{d_k}\right\|^2\right)^{1/2}. \tag{5}$$

Kernel density estimates are output as smooth curves with the amount of smoothing governed by a bandwidth value used during calculation [28]. Densities are calculated by placing kernels over the distribution of data points. Kernels that overlap one another increase density values in shared areas of the distribution. For univariate and multivariate data, the Gaussian kernel is the

most commonly used one. In particular, for 3D data, the standardized Gaussian kernel function (see Figure 2.2) is given by

$$K(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{3}{2}}} \exp\left(-\frac{\|\boldsymbol{x}\|^2}{2}\right), \quad \forall \boldsymbol{x} \in \mathbb{R}^3. \tag{6}$$

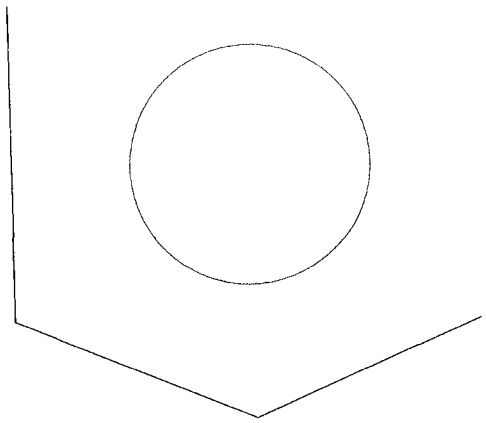


Figure 2.2: 3D level surface of the Gaussian kernel function $K(\boldsymbol{x})$.

Motivated by kernel density estimation as an important data analytic tool that provides a very effective way of showing structure in a set of a data [28], we propose a mesh kernel flow. This mesh denoising flows updates iteratively each mesh vertex according to the following rule

$$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \sum_{j \sim i} \mathcal{K}_{H_i}(\boldsymbol{v}_i - \boldsymbol{v}_j) \frac{1}{\sqrt{d_i}} \left(\frac{\boldsymbol{v}_j}{\sqrt{d_j}} - \frac{\boldsymbol{v}_i}{\sqrt{d_i}} \right) \tag{7}$$

where the φ-function is given by

$$\mathcal{K}_{H_i}(\boldsymbol{v}_i - \boldsymbol{v}_j) = \det(H_i)^{-1/2} \, K\big(H_i^{-1/2}(\boldsymbol{v}_i - \boldsymbol{v}_j)\big) \tag{8}$$

and H_i is a symmetric positive semi-definite matrix. This matrix defines a covariance matrix around the neighborhood of vertex v_i, and it is given by

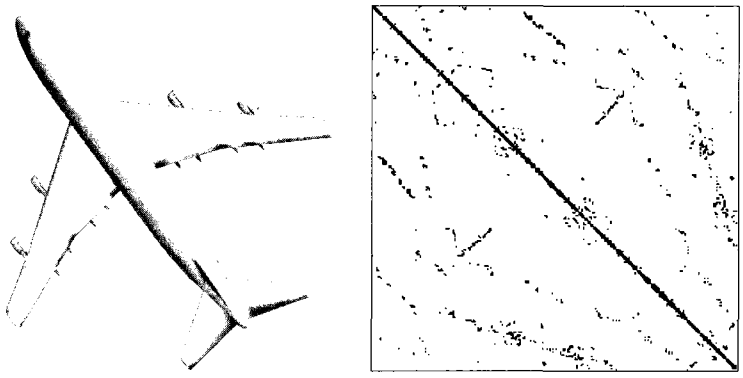
$$H_i = \sum_{j \sim i} (\boldsymbol{v}_j - \boldsymbol{c}_i)(\boldsymbol{v}_j - \boldsymbol{c}_i)^T, \text{ where } \boldsymbol{c}_i = \frac{1}{d_i} \sum_{j \sim i} \boldsymbol{v}_j. \tag{9}$$

It is worth pointing out that H_i is also called the bandwidth matrix in the context of kernel smoothing and it measures the amount of smoothing. Also, note that the choice of the kernel function appears to have very little effect on the quality of the proposed denoising approach. However, the selection of the bandwidth matrix is widely recognized to have more effect on the performance of kernel density estimation. In this proposed method, we use the trivariate Gaussian density as a kernel function, and the data-driven neighborhood covariance as a bandwidth matrix.

The neighborhood weighting kernel $\mathcal{K}_{H_i}$ may be expressed in matrix form as $\mathcal{K} = (\kappa_{ij})$, which will be referred to as mesh neighborhood weighting kernel matrix. Each element κ_{ij} of this $m \times m$ sparse matrix is given by the right-hand side of Eq. (8). Thus, the mesh neighborhood weighting kernel matrix may be written as

$$\mathcal{K} = (\kappa_{ij}) = \begin{cases} \det(H_i)^{-1/2}(2\pi)^{-3/2} & \text{if } \boldsymbol{v}_i = \boldsymbol{v}_j \\ \det(H_i)^{-1/2}\, K\big(H_i^{-1/2}(\boldsymbol{v}_i - \boldsymbol{v}_j)\big) & \text{if } \boldsymbol{v}_i \sim \boldsymbol{v}_j \\ 0 & \text{o.w.} \end{cases} \tag{10}$$

Note that the value of the the first row of Eq. (10) results directly from Eq. (6) when $\boldsymbol{x} = 0$, that is $K(\mathbf{0}) = (2\pi)^{-3/2}$. Therefore, like many iterative methods the update flow of the proposed approach can be easily implemented in terms of matrix-vector products. Figure 2.3 displays a 3D object and its mesh neighborhood weighting kernel matrix.



(a) (b)

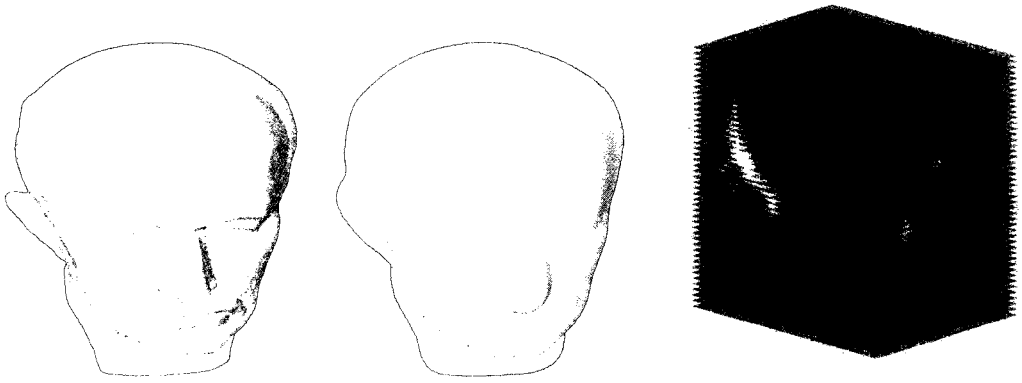
Figure 2.3: (a) 3D object with $m = 3403$ vertices and its (b) mesh neighborhood weighting kernel matrix.

Next we show how kernel density estimation can be used for 3D mesh reconstruction. Given

m mesh vertices $\boldsymbol{v}_i$, let $\boldsymbol{v}$ be a 3D vector whose i-th realization is $\boldsymbol{v}_i$. Thus, the mesh kernel density estimate (KDE) may be written in the general form

$$\hat{f}(\boldsymbol{v}) = \frac{1}{m}\sum_{i=1}^{m} \det(H)^{-1/2}\, K\big(H^{-1/2}(\boldsymbol{v}-\boldsymbol{v}_i)\big) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{K}_H(\boldsymbol{v}-\boldsymbol{v}_i), \tag{11}$$

where $H = \sum_{i=1}^{m}(\boldsymbol{v}_i - \boldsymbol{c})(\boldsymbol{v}_i - \boldsymbol{c})^T$ is the mesh covariance matrix which controls the smoothness of the resulting density estimate, and $\boldsymbol{c} = (1/m)\sum_{i=1}^{m} \boldsymbol{v}_i$ is the mesh centroid. The mesh KDE is a trivariate volumetric function which may be graphically visualized by plotting a level surface (also called implicit surface or isosurface) of $\hat{f}$ as shown in Figure 2.4(b). This figure displays an isosurface of the mesh KDE using the vertices of the 3D object shown in Figure 2.4(a). The horizontal slices of the mesh KDE are also depicted in Figure 2.4(c).



(a) (b) (c)

Figure 2.4: (a) 3D object, (b) mesh KDE, and (c) horizontal slices of the mesh KDE.

2.3 Experimental Results

This section presents experimental results where the mean filtering [12], angle median filtering [12], Laplacian flow [7], weighted Laplacian flow [15, 16], geometric diffusion [17], bilateral filtering [13], and the proposed method [1] are applied to noisy 3D models obtained by adding Gaussian noise to the original 3D models. The standard deviation of the noise was set to 2% of the mean edge length, that is $\sigma = 0.02\,\bar{\ell}$, where $\bar{\ell}$ is given by Eq. (4) in chapter 1.

In practical applications, the covariance matrix H_i given by Eq. (9) may become singular. To circumvent this singularity problem and also to ensure the stability of the proposed algorithm, we use a regularized covariance matrix as follows

$$H_i = H_i + \lambda I \tag{12}$$

where I is a 3×3 identity matrix and λ is a positive regularization parameter.

Figure 2.5 displays the mesh denoising results obtained by our proposed method for different values of the regularization parameter λ, where the number of iterations was to set to 5. As can be seen in Figure 2.5, the value $\lambda = 0.4$ gives the best denoising result for the 3D rabbit model. And as a the value of λ increases, the rabbit model becomes more noisier. Also, we noticed through experimentation that a smaller value of λ often tends to produce a geometrically distorted shape of the 3D object as shown in Figure 2.5(c). Therefore, the regularization parameter should be tuned to be small enough to capture the intrinsic shape of a 3D object and large enough not to recapture noise.

Figure 2.6 depicts the output results of the proposed approach at different iteration numbers. These results show that, using the proposed approach, the noise can be removed with just a small number of iterations and that the sharp features are well preserved when the regularization parameter is appropriately chosen.

2.3.1 Qualitative evaluation of the proposed method

Figure 2.7(c) through Figure 2.7(h) show the denoising results obtained via Laplacian flow, weighted Laplacian flow, mean filtering, angle median filtering, bilateral filtering, and the proposed method respectively. These results clearly show that our method outperforms all the mesh filtering techniques used for comparison. Moreover, the proposed method is simple and easy to implement. One main advantage of the proposed algorithm is that it requires only few iterations to smooth out the noise, whereas the weighted Laplacian flow, the mean and the angle median filters require substantial computational time. In Figure 2.8, we use the zoom tool to enlarge the view of the 3D cow model's head in order to clearly show the better performance of our proposed

algorithm. In particular, the geometric structures and the fine details around the eye and the ear of the 3D cow model are very well preserved by our method. More experimental results showing the better performance of the proposed algorithm are presented in Figure 2.9 and Figure 2.10.

In all the experiments, we observe that the proposed technique is able to suppress noise while preserving important geometric structure of the 3D models in a very fast and efficient way. This better performance is in fact consistent with a variety of 3D models used for experimentation.

2.3.2 Quantitative evaluation of the proposed method

Let $\mathbb{M}$ and $\widehat{\mathbb{M}}$ be the original model and the smoothing result model with vertex sets $\mathcal{V} = \{\boldsymbol{v}_i\}_{i=1}^m$ and $\widehat{\mathcal{V}} = \{\hat{\boldsymbol{v}}_i\}_{i=1}^m$ respectively. To quantify the performance of the proposed approach, we computed the visual error metric [29] given by

$$E = \frac{1}{2m}\left(\sum_{i=1}^{m}\|\boldsymbol{v}_i - \hat{\boldsymbol{v}}_i\|^2 + \sum_{i=1}^{m}\|\mathcal{I}(\boldsymbol{v}_i) - \mathcal{I}(\hat{\boldsymbol{v}}_i)\|^2\right), \tag{13}$$

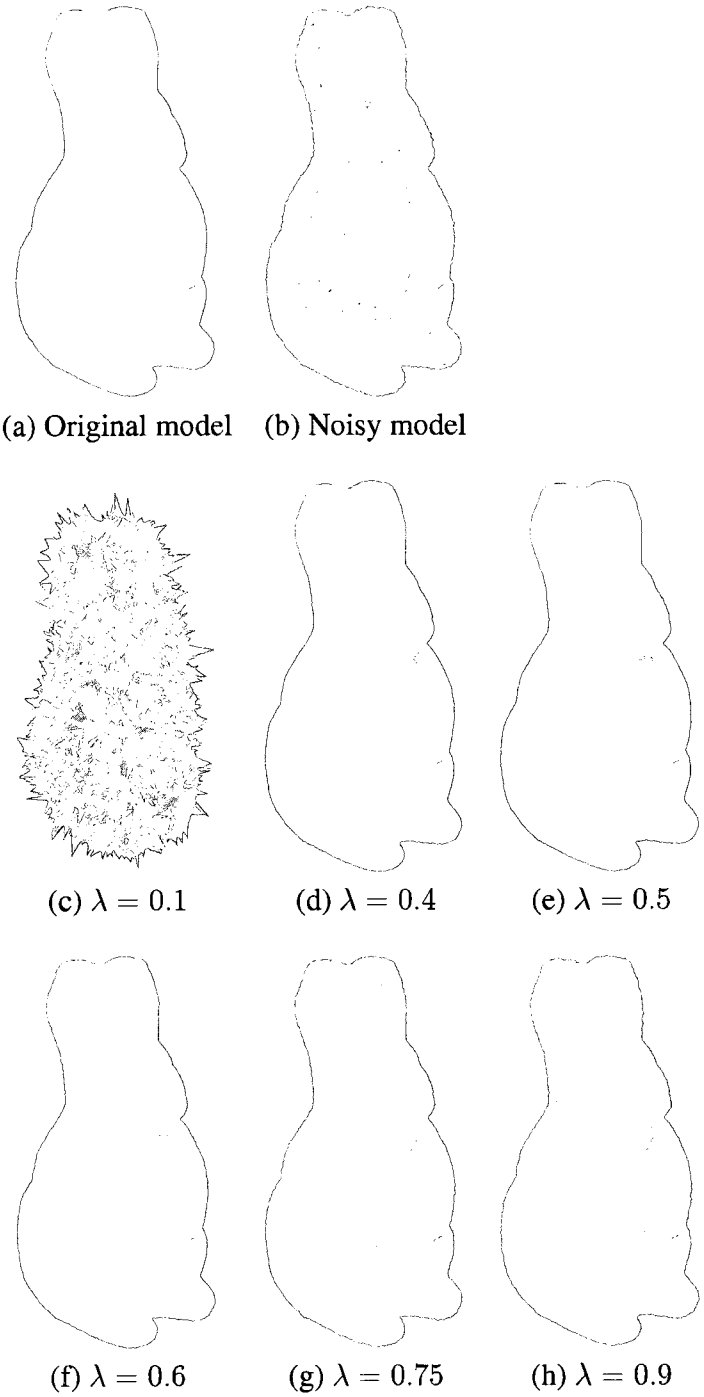
where $\mathcal{I}$ is the geometric Laplacian operator defined as

$$\mathcal{I}(\boldsymbol{v}_i) = \boldsymbol{v}_i - \frac{1}{d_i}\sum_{j\sim i}\boldsymbol{v}_j. \tag{14}$$

Intuitively, the error metric given by Eq. (13) captures the visual difference between the original model and the denoised one by taking into account geometric closeness and local smoothness difference. More specifically, the first term of this visual metric measures how close the vertices in both models are, whereas the second term captures the object smoothness which basically represents the visual properties of the human eye. The values of the visual error metric for some experiments are depicted in Figure 2.11(a) and Figure 2.11(b) which clearly show that the proposed method gives the best results, indicating the consistency with the subjective comparison. Unlike the Laplacian flow which tends to produce spherically-shaped outputs at higher iteration numbers [7], the proposed approach is, however, experimentally shown to stabilize as depicted in Figure 2.12 and Figure 2.13. It is apparent from these figures that the output results at iterations 20 are 30 are visually indistinguishable, indicating that the proposed algorithm produces a stable solution.

2.3.3 Choice of the regularization parameter

A mentioned earlier, the regularization parameter should be chosen appropriately in order to obtain satisfactory mesh denoising results. This parameter may be estimated experimentally using the visual error as shown in Figure 2.14(a) and Figure 2.14(b), which display the plots of the visual error vs. the regularization parameter for different iteration numbers of the proposed approach.

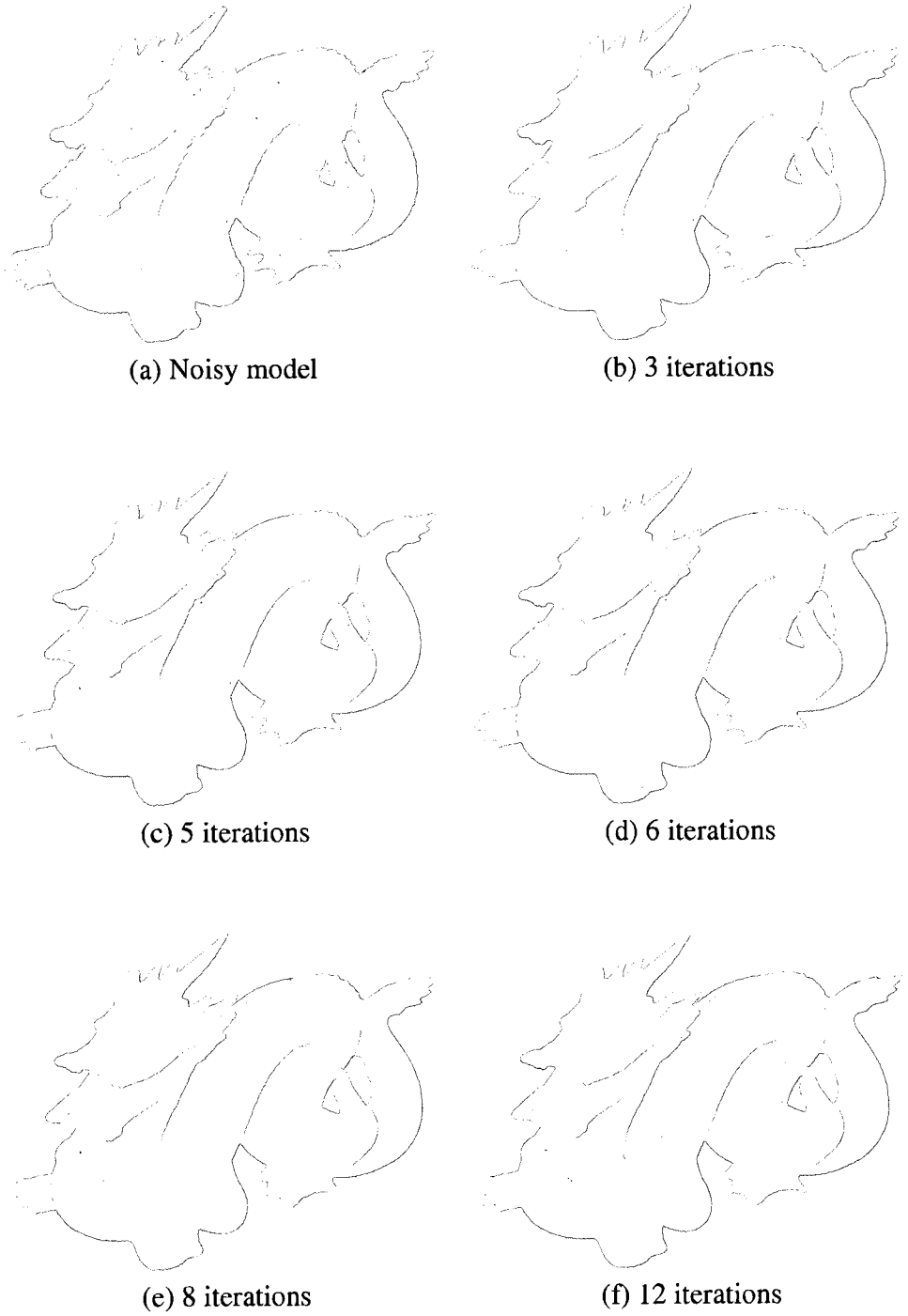


(a) Original model (b) Noisy model

(c) $\lambda = 0.1$ (d) $\lambda = 0.4$ (e) $\lambda = 0.5$

(f) $\lambda = 0.6$ (g) $\lambda = 0.75$ (h) $\lambda = 0.9$

Figure 2.5: Output results of our proposed mesh denoising approach for different values of the regularization parameter. The number of iterations is set to 5.



(a) Noisy model (b) 3 iterations

(c) 5 iterations (d) 6 iterations

(e) 8 iterations (f) 12 iterations

Figure 2.6: Output results of our proposed mesh denoising approach at different iteration numbers. The regularization parameter is set to $\lambda = 0.8$

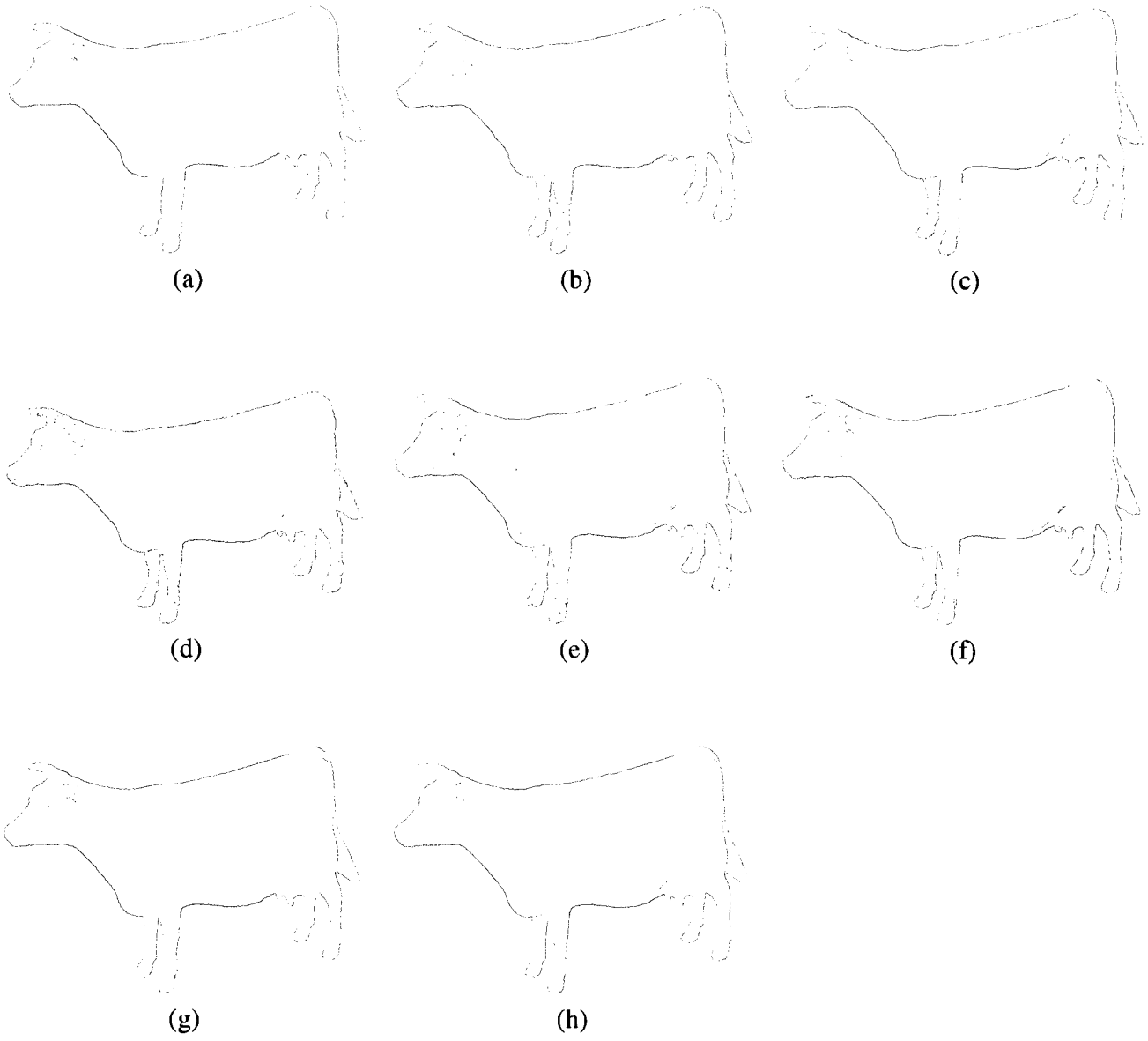


Figure 2.7: Denoising results for the 3D cow model: (a) original model; (b) noisy model; (c) Laplacian flow; (d) weighted Laplacian flow; (e) mean filtering; (f) angle median filtering; (g) bilateral mesh flow; (h) our proposed approach. The number of iterations is set to 6 in each case.

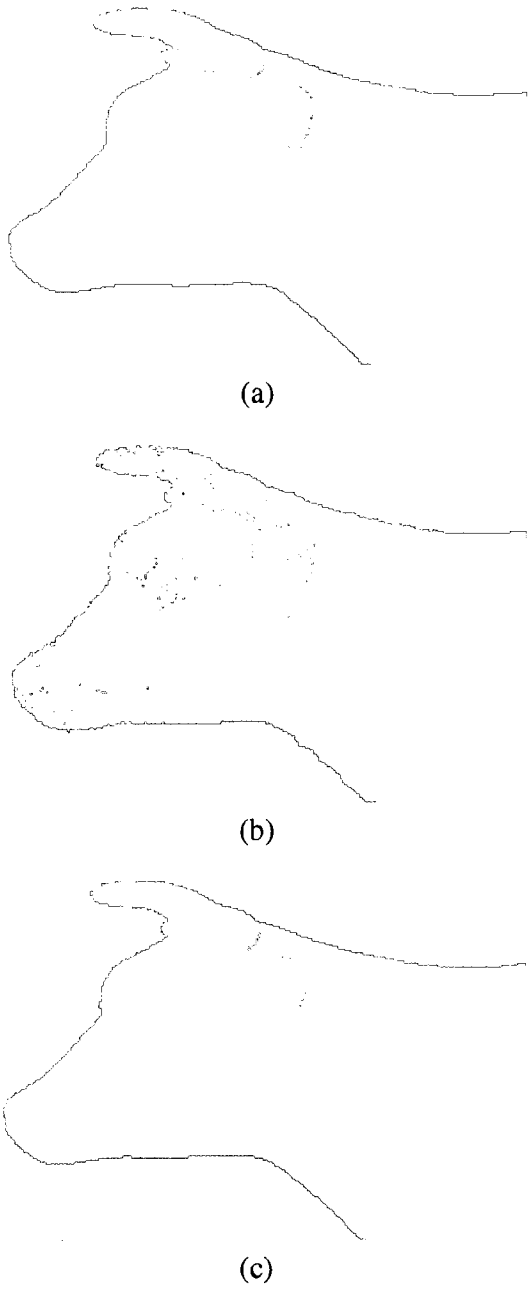


Figure 2.8: Zoomed portion of the 3D cow model: (a) original model; (b) noisy model; (c) output result of our proposed approach with $\lambda = 0.8$. The number of iterations is set to 6.

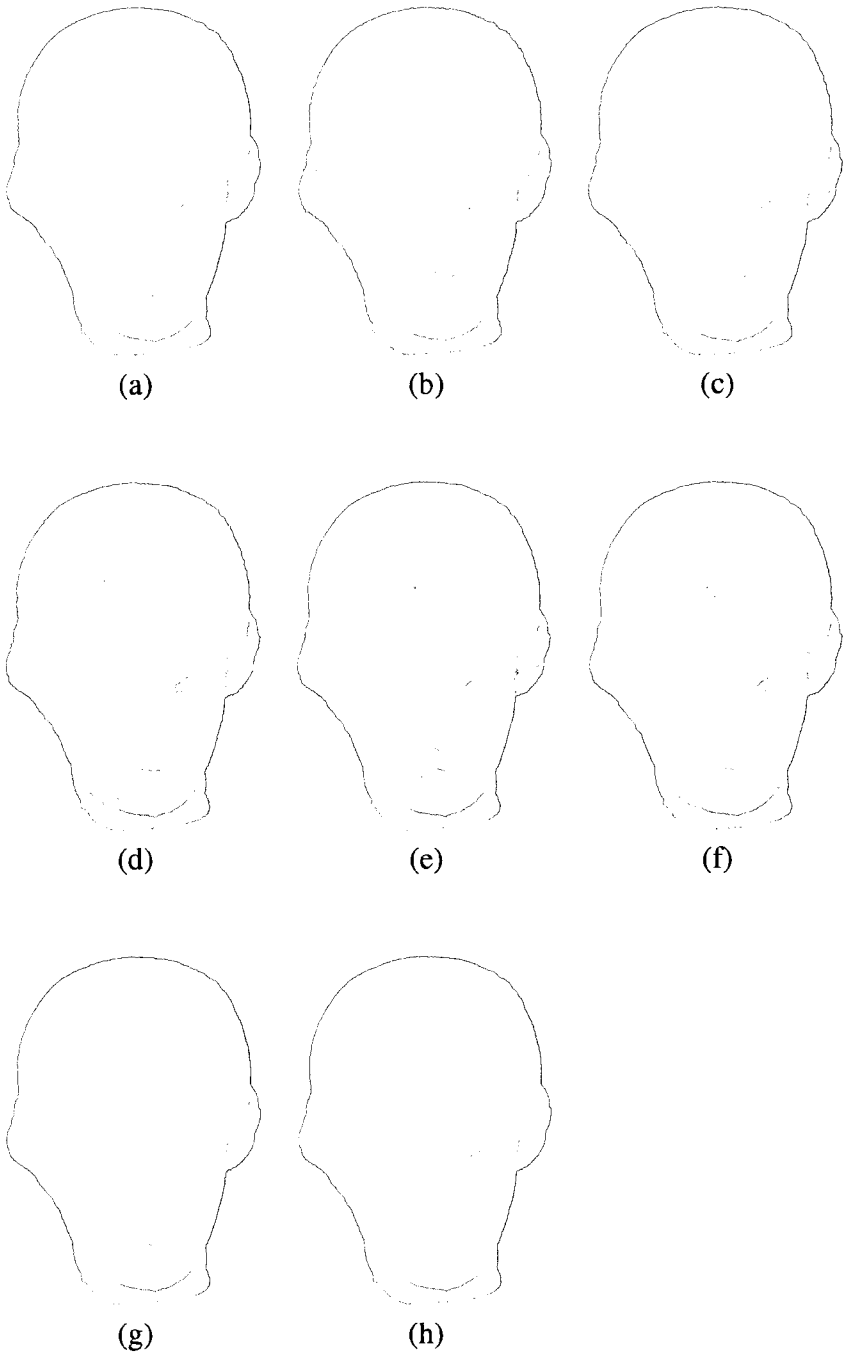


Figure 2.9: Denoising results for the 3D igea model: (a) original model; (b) noisy model; (c) Laplacian flow; (d) weighted Laplacian flow; (e) mean filtering; (f) angle median filtering; (g) bilateral mesh flow; (h) our proposed approach. The number of iterations is set to 6.

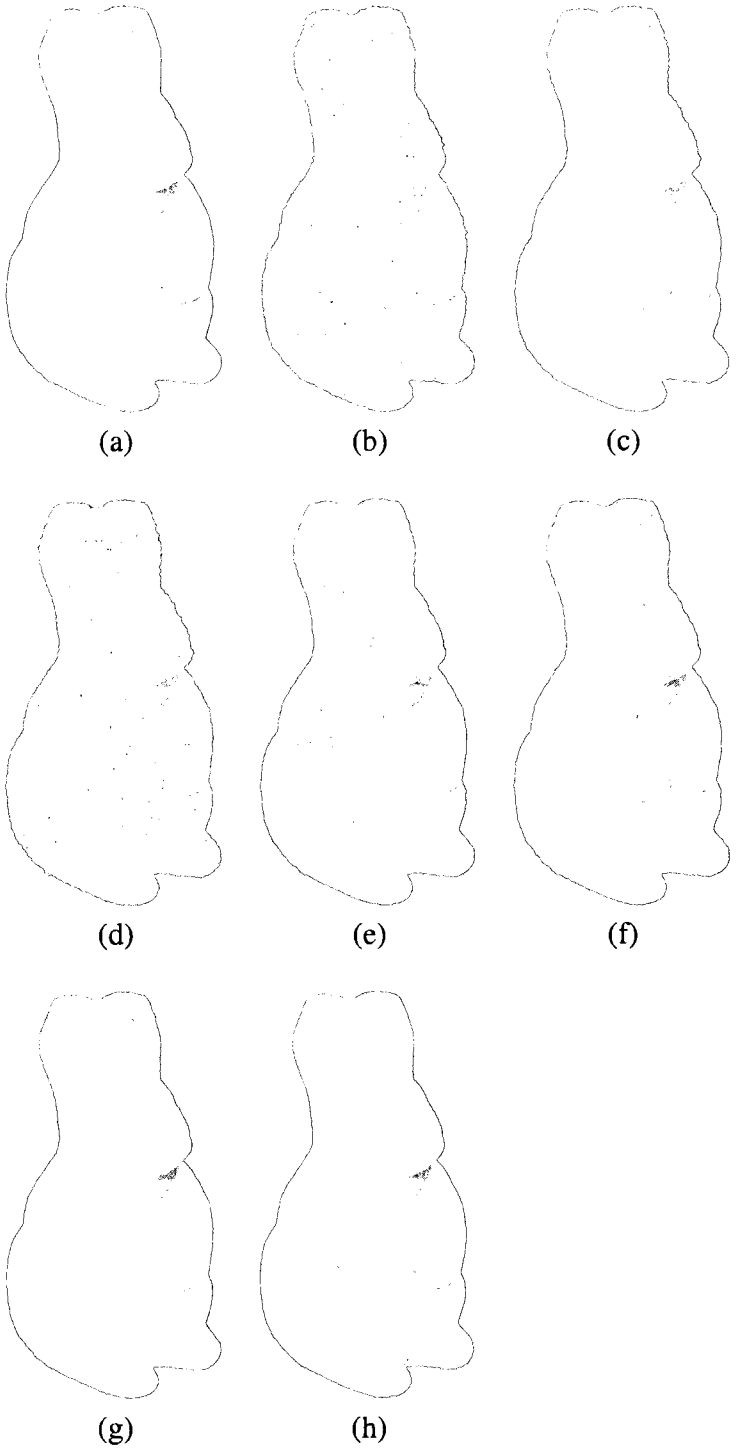


Figure 2.10: Denoising results for the 3D rabbit model: (a) original model; (b) noisy model; (c) Laplacian flow; (d) weighted Laplacian flow; (e) mean filtering; (f) angle median filtering; (g) bilateral mesh flow; (h) our proposed approach. The number of iterations is set to 3.

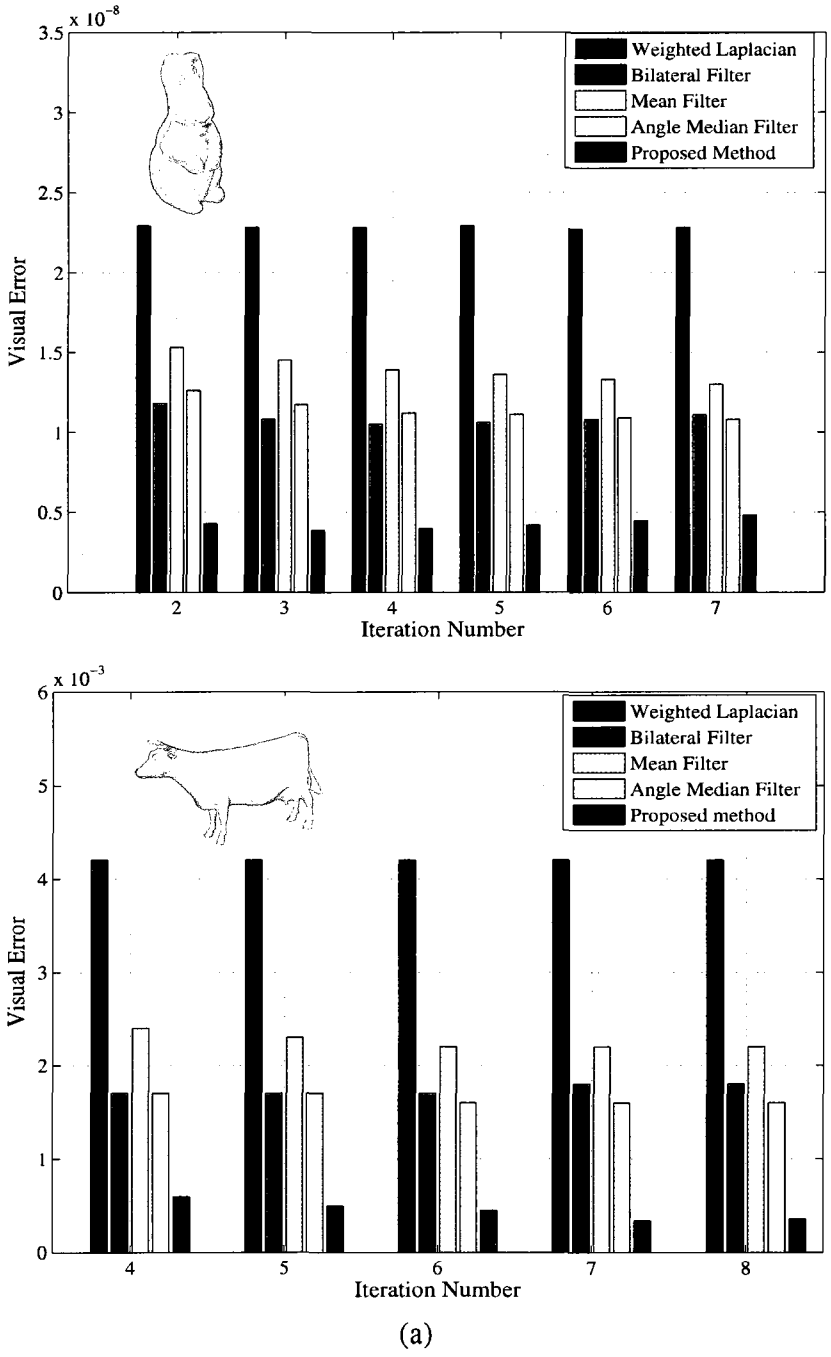
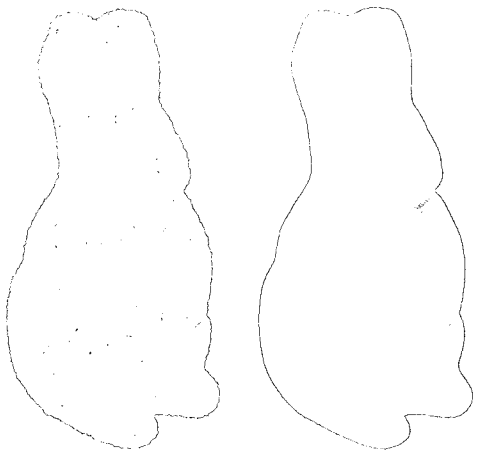


Figure 2.11: Visual error comparison results between the proposed approach and other methods for the (a) rabbit and (b) cow models.



(a) Noisy model (b) 10 iterations

(c) 20 iterations (d) 30 iterations

Figure 2.12: Denoising results for the 3D rabbit model using $\lambda = 0.4$ at higher iteration numbers.

(a) Noisy model (b) 10 iterations

(c) 20 iterations (d) 30 iterations

Figure 2.13: Denoising results for the 3D cow model using $\lambda = 0.8$ at higher iteration numbers.

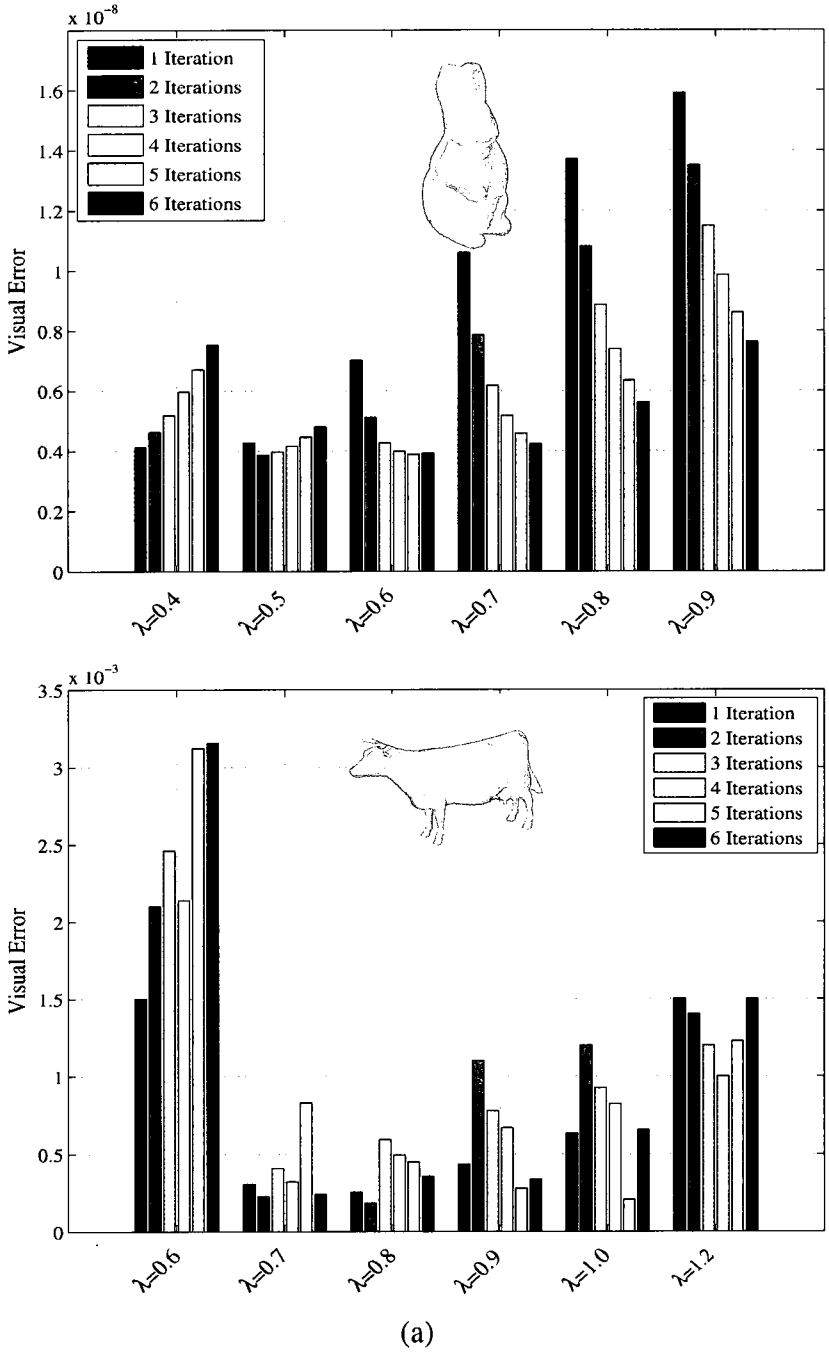


Figure 2.14: Visual error vs. regularization parameter with different number of iterations for the (a) rabbit and (b) cow models.

CHAPTER 3

Mesh Umbrella Operator for 3D Compression

In this chapter we propose a 3D mesh compression technique using the centroidal mesh neighborhood information. The key idea is to apply eigen-decomposition to the mesh umbrella matrix, and then discard the largest eigenvalues/eigenvectors in order to reduce the dimensionality of the new spectral basis so that most of the energy is concentrated in the low frequency coefficients. Extensive experimental results demonstrate the effectiveness of the proposed approach in 3D compression.

3.1 Introduction

Compression of images and shapes has long been the central theme of image processing and computer vision. Its importance is increasing rapidly in the field of computer graphics and multimedia communication because it is difficult to transmit digital information efficiently over the internet without its compression. 3D objects consist of geometric and topological information, and their compressed representation is an important step towards a variety of computer graphics applications including indexing, retrieval, and matching in a database of 3D models [30–32, 48].

In this chapter, we present a novel compression technique for 3D models. The proposed method is inspired by previous works on object compression, and in particular the Laplacian matrix-based

compression technique recently introduced by Karni *et al* [29]. Our proposed approach is based on the so-called mesh umbrella operator [7] which replaces each mesh vertex by the centroid of its neighbors. The primary motivation of the proposed method is to encode a 3D shape into a more compact representation by retaining the largest eigenvalues/eigenvectors of the mesh umbrella matrix. We essentially discard the smallest eigenvalues and the corresponding eigenvectors, hence reducing the dimensionality of the new basis. In other words, most of the energy is concentrated in the low frequency coefficients. To gain further insight into the proposed compression method, some numerical experiments are provided to demonstrate the potential and the much improved performance of the proposed methodology in 3D object compression. This improved performance was evaluated by computing a nonlinear visual metric error between the original 3D model and the compressed model.

The rest of this chapter is organized as follows. In the next section, we introduce the mesh umbrella matrix. In Section 3.3, a spectral 3D mesh compression technique is proposed. In Section 3.4, we provide experimental results to demonstrate a much improved performance of the proposed method in 3D mesh compression.

3.2 Umbrella matrix of a triangle mesh

The mesh umbrella operator is defined as

$$\mathcal{U}(\boldsymbol{v}_i) = \frac{1}{d_i} \sum_{\boldsymbol{v}_j \in \boldsymbol{v}_i^*} (\boldsymbol{v}_j - \boldsymbol{v}_i), \quad \forall i = 1, \ldots, m \tag{1}$$

Applying the umbrella operator to the triangle mesh is equivalent to move each vertex to the centroid of its neighbors. In matrix form, the mesh umbrella operator may be written as

$$\boldsymbol{U} = K\boldsymbol{V} = (\boldsymbol{u}_1 \ \boldsymbol{u}_2 \ \ldots \boldsymbol{u}_m)^T \in \mathbb{R}^{m \times 3},$$

where $\boldsymbol{V} = (\boldsymbol{v}_1\ \boldsymbol{v}_2\ \ldots \boldsymbol{v}_m)^T$ is the $m \times 3$ mesh vertex matrix, and K is an $m \times m$ sparse matrix which we refer to as the mesh umbrella matrix and it is given by

$$K = (k_{ij}) = \begin{cases} -1 & \text{if } \boldsymbol{v}_i = \boldsymbol{v}_j \\ 1/d_i & \text{if } \boldsymbol{v}_i \sim \boldsymbol{v}_j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Let $\boldsymbol{v}_i = (x_i, y_i, z_i)^T \in \mathcal{V}$, then the mesh vertex matrix having as rows the coordinates of the mesh vertices may be written as

$$\boldsymbol{V} = \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_m & y_m & z_m \end{pmatrix} \in \mathbb{R}^{m \times 3},$$

Figure 3.1 illustrates an example of a 2D and a 3D triangle meshes and their sparse umbrella matrices.

3.3 Proposed Method

The eigen-decomposition of the umbrella matrix K yields $KB = B\Lambda$, where $B = (\boldsymbol{b}_1\ \boldsymbol{b}_2\ \ldots \boldsymbol{b}_m)$ is an orthogonal matrix whose columns are the eigenvectors which we refer to as umbrella basis vectors, and $\Lambda = \text{diag}\{\lambda_i : i = 1, \ldots, m\}$ is a diagonal matrix of eigenvalues arranged in increasing order of magnitude. We may express the mesh vertex matrix in the subspace spanned by the umbrella matrix eigenvectors as follows

$$\boldsymbol{V} = B\boldsymbol{C}^T = \sum_{i=1}^{m} \boldsymbol{b}_i \boldsymbol{c}_i^T, \tag{3}$$

where $C = (\boldsymbol{c}_1\ \boldsymbol{c}_2\ \ldots \boldsymbol{c}_m) \in \mathbb{R}^{m \times 3}$ is a matrix of the spectral coefficient vectors, that is $\boldsymbol{C} = B^T\boldsymbol{V}$ is the projection of the mesh vertex matrix onto the umbrella basis vectors. Moreover,

$$\boldsymbol{V} = \underbrace{\sum_{i=1}^{r} \boldsymbol{b}_i \boldsymbol{c}_i^T}_{\text{compressed}} + \sum_{i=r+1}^{m} \boldsymbol{b}_i \boldsymbol{c}_i^T = B_r \boldsymbol{C}_r^T + \sum_{i=r+1}^{m} \boldsymbol{b}_i \boldsymbol{c}_i^T, \tag{4}$$

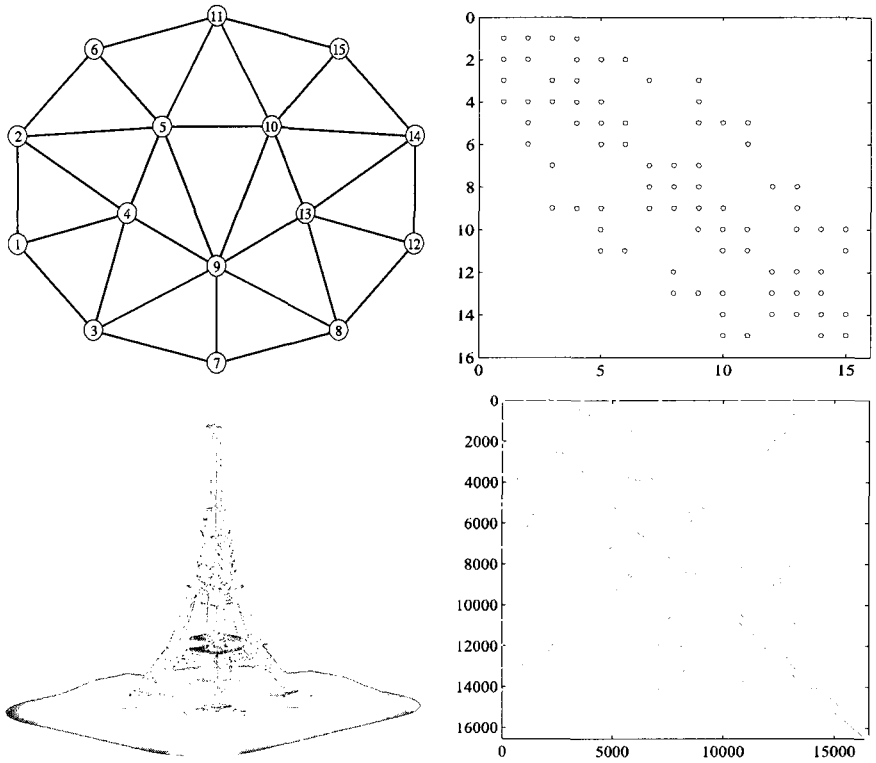


Figure 3.1: 2D/3D triangle meshes (left) and their umbrella matrices (right).

where r is usually chosen to be smaller than m, and hence this yields a compressed version $\boldsymbol{V}_r$ of the original vertex matrix $\boldsymbol{V}$. The matrix $B_r = (\boldsymbol{b}_1\ \boldsymbol{b}_2\ \ldots \boldsymbol{b}_r)$ contains the r spectral basis vectors, and the matrix $\boldsymbol{C}_r = (\boldsymbol{c}_1\ \boldsymbol{c}_2\ \ldots \boldsymbol{c}_r)$ contains the r spectral coefficient vectors. The reconstructed mesh vertex matrix $\hat{\boldsymbol{V}}$ is obtained by minimizing the error $\|B_r\boldsymbol{C}_r^T - \boldsymbol{V}\|^2$ with respect to the matrix $\boldsymbol{C}_r$. The minimization of this error yields $\hat{\boldsymbol{C}}_r = (B_r^T B_r)^+ B_r^T \boldsymbol{V}$ where $(B_r^T B_r)^+$ is the pseudo-inverse of the matrix $B_r^T B_r$, and hence the mesh vertex matrix of the compressed model is estimated by

$$\hat{\boldsymbol{V}} = B_r \hat{\boldsymbol{C}}_r^T = B_r (B_r^T B_r)^+ B_r^T \boldsymbol{V}. \tag{5}$$

Note that $\text{proj}_{B_r}\boldsymbol{V} = B_r(B_r^T B_r)^+ B_r^T \boldsymbol{V}$ gives the projection of the mesh vertex matrix onto the space spanned by the columns of the matrix B_r.

If we rewrite $\boldsymbol{V}$ and $\boldsymbol{C}$ in the form of 3-column matrices, that is

$$\boldsymbol{V} = (\boldsymbol{v}_x\ \boldsymbol{v}_y\ \boldsymbol{v}_z) = \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_m & y_m & z_m \end{pmatrix}$$

and

$$\boldsymbol{C}^T = (\boldsymbol{c}_x \quad \boldsymbol{c}_y \quad \boldsymbol{c}_z) = \begin{pmatrix} c_{x_1} & c_{y_1} & c_{z_1} \\ c_{x_2} & c_{y_2} & c_{z_2} \\ \vdots & \vdots & \vdots \\ c_{x_m} & c_{y_m} & c_{z_m} \end{pmatrix},$$

then the spectral coefficients in the x, y, and z-dimension are given by $\boldsymbol{c}_x = B^T\boldsymbol{v}_x$, $\boldsymbol{c}_y = B^T\boldsymbol{v}_y$, and $\boldsymbol{c}_z = B^T\boldsymbol{v}_z$ respectively as shown in Figure 3.2.

3.4 Experimental Results

This section presents experimental results where the Laplacian matrix-based compression technique [29], and the proposed method [5] are applied to the compression of the original models shown in Figure 3.3.

Figure 3.4(a) through Figure 3.4(d) show the mesh compression results using the Laplacian-based method, and the proposed approach. These results clearly show that the Laplacian-based technique has a poor compression performance in comparison with the proposed method as illustrated in Figure 3.5 where we use the zoom tool to enlarge the view of the 3D bunny model's head in order to clearly show the better performance of our proposed algorithm. In particular, the geometric structures and the fine details around the face and the ears of the 3D bunny model are very well preserved by our method. More experimental results showing the better performance of the proposed algorithm are presented in Figure 3.6 and Figure 3.7.

In all the experiments, we observe that the proposed technique has much better compression capabilities than the Laplacian-based approach while preserving important geometric features of

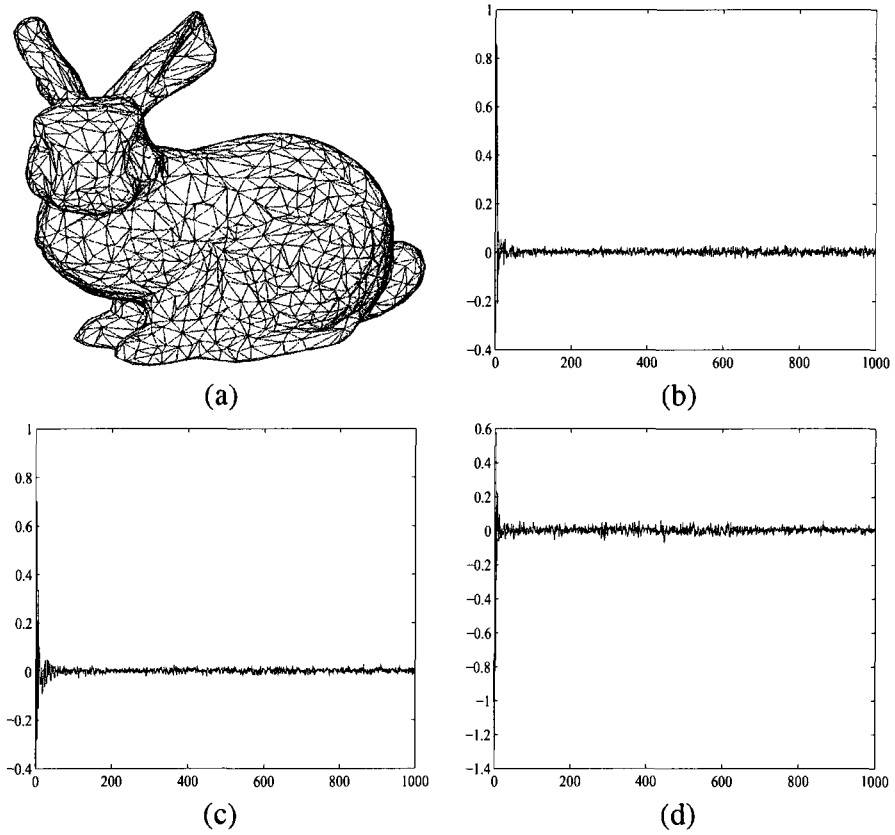


Figure 3.2: (a) 3D bunny model, and its spectral coefficients of in the (b) x-dimension, (c) y-dimension, and (d) z-dimension.

the 3D models in a very fast and efficient way. This better performance is in fact consistent with a large number of 3D models used for experimentation.

To quantify the better performance of the proposed approach in comparison with the Laplacian mesh compression, we propose a nonlinear visual error $D(\mathbb{M}, \widehat{\mathbb{M}})$ defined between the original model $\mathbb{M}$ and the compressed model $\widehat{\mathbb{M}}$ as follows

$$\begin{aligned} D(\mathbb{M}, \widehat{\mathbb{M}}) &= \frac{1}{2m} \sum_{i=1}^{m} \|\boldsymbol{v}_i - \hat{\boldsymbol{v}}_i\|^2 \\ &\quad + \frac{1}{2m} \sum_{i=1}^{m} \|\mathcal{A}(\boldsymbol{v}_i) - \mathcal{A}(\hat{\boldsymbol{v}}_i)\|^2, \end{aligned} \tag{6}$$

where $\{\boldsymbol{v}_i\}_{i=1}^m$ and $\{\hat{\boldsymbol{v}}_i\}_{i=1}^m$ are the mesh vertex sets of $\mathbb{M}$ and $\widehat{\mathbb{M}}$ respectively. $\mathcal{A}$ is a nonlinear

Figure 3.3: Original 3D models used for experimentation: camel, bunny, and femur (upper part of the long bone in the thigh).

diffusion operator [27] defined as

$$\mathcal{A}(\boldsymbol{v}_i) = \frac{1}{d_i} \sum_{\boldsymbol{v}_j \in \boldsymbol{v}_i^\star} (\boldsymbol{v}_i - \boldsymbol{v}_j)\Big(g(|\nabla \boldsymbol{v}_i|) + g(|\nabla \boldsymbol{v}_j|)\Big), \tag{7}$$

where the gradient magnitudes are given by

$$|\nabla \boldsymbol{v}_i| = \sqrt{\sum_{\boldsymbol{v}_j \in \boldsymbol{v}_i^\star} \left\| \frac{\boldsymbol{v}_i}{\sqrt{d_i}} - \frac{\boldsymbol{v}_j}{\sqrt{d_j}} \right\|^2}, \tag{8}$$

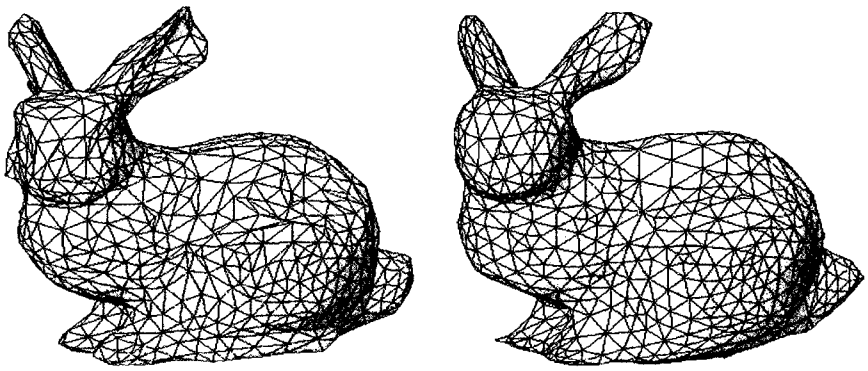
$$|\nabla \boldsymbol{v}_j| = \sqrt{\sum_{\boldsymbol{v}_k \in \boldsymbol{v}_j^\star} \left\| \frac{\boldsymbol{v}_j}{\sqrt{d_j}} - \frac{\boldsymbol{v}_k}{\sqrt{d_k}} \right\|^2}, \tag{9}$$

and g is Cauchy weight function (see Figure 3.8) given by

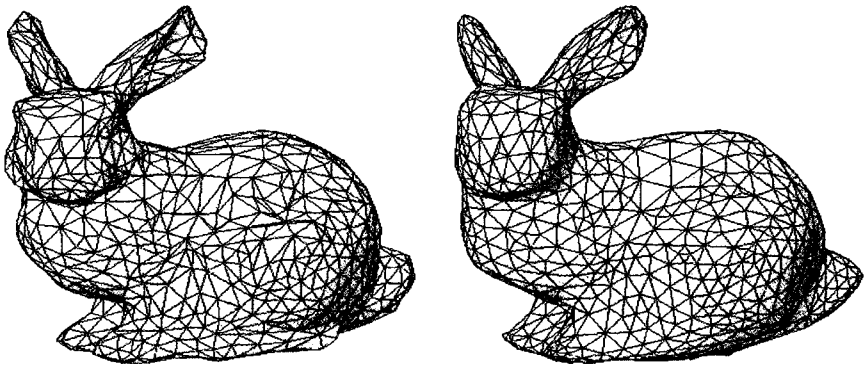
$$g(x) = \frac{1}{1 + x^2/c^2}, \tag{10}$$

with a constant tuning parameter c that needs to be estimated. It can be shown (see [33]) that the 95% asymptotic efficiency on the standard Gaussian distribution is obtained with the tuning constant $c = 2.3849$. This tuning value is used in all the experimental results.

Note that the visual error $D(\mathbb{M}, \widehat{\mathbb{M}})$ requires the use of two neighboring rings as depicted in Figure 3.9. Intuitively, the anisotropic operator $\mathcal{A}$ introduces some smoothing effect which

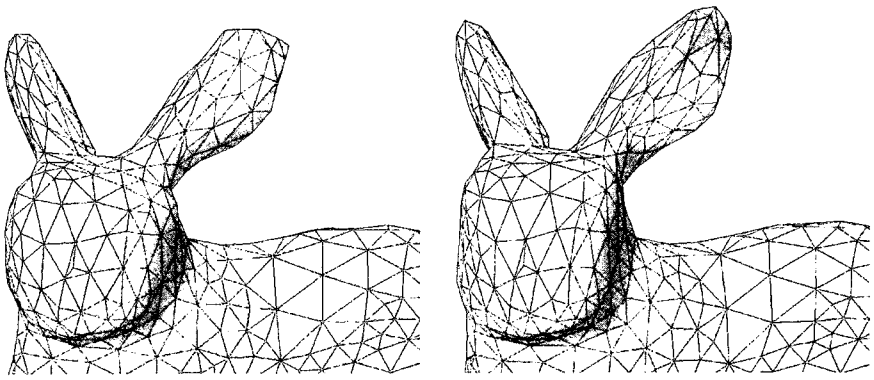


(a) $r = 500$ basis functions (b) $r = 125$ basis functions



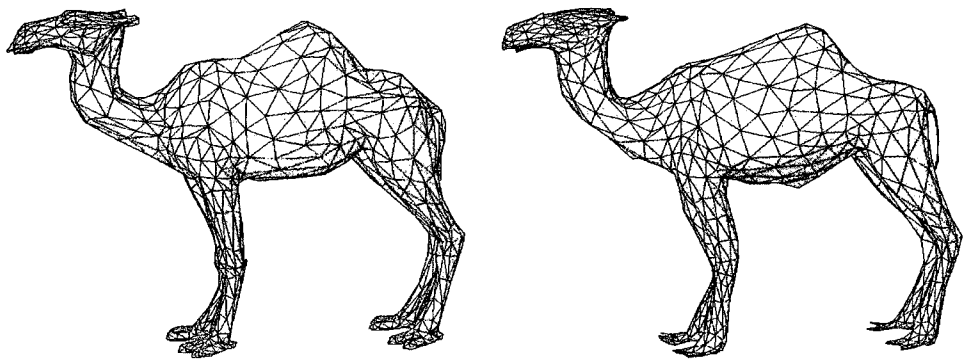
(c) $r = 500$ basis functions (d) $r = 125$ basis functions

Figure 3.4: Spectral compression of the 3D bunny at different resolutions. (a)-(b): Laplacian compression. (c)-(d): Proposed method.

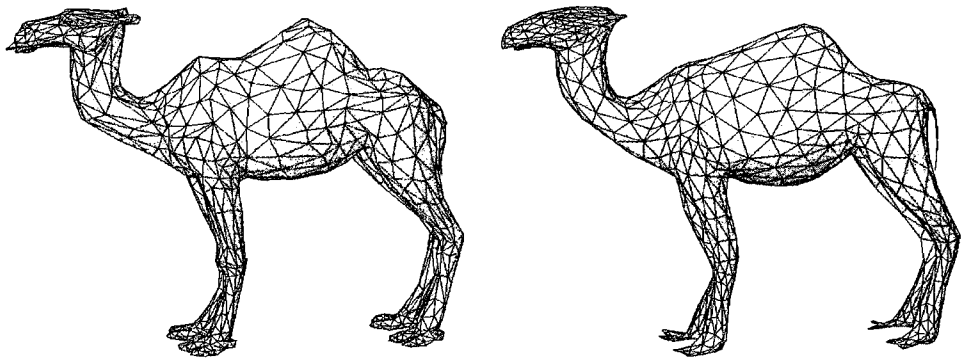


(a) Laplacian compression (b) proposed method

Figure 3.5: Close-up comparison of Laplacian compression and the proposed approach using $r = 125$ basis functions.



(a) $r = 500$ basis functions (b) $r = 125$ basis functions

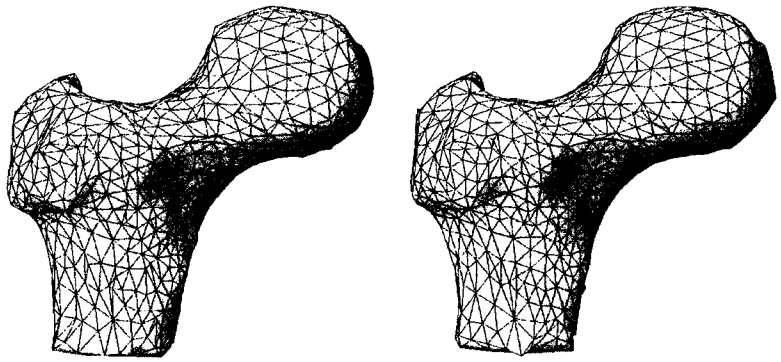


(c) $r = 500$ basis functions (d) $r = 125$ basis functions

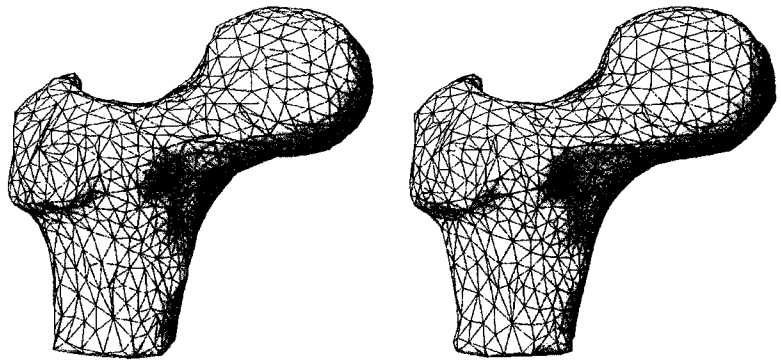
Figure 3.6: Spectral compression of the 3D camel at different resolutions. (a)-(b): Laplacian compression. (c)-(d): Proposed method.

may be explained as follows: in flat regions of a 3D mesh where the vertex gradient magnitudes are relatively small, Eq. (7) is reduced to a linear operator which tends to smooth more but the smoothing effect is unnoticeable. And around the sharp features of the 3D mesh where the vertex gradient magnitudes are large, the nonlinear diffusion operator given by Eq. (7) tends to smooth less and hence leads to a much better preservation of the mesh geometric structures.

The values of visual error metric for all the experiments are depicted in Figure 3.10-3.12 which clearly show that the proposed method gives the best results indicating the consistency with the subjective comparison.



(a) $r = 500$ basis functions (b) $r = 250$ basis functions



(c) $r = 500$ basis functions (d) $r = 250$ basis functions

Figure 3.7: Spectral compression of the 3D femur at different resolutions. (a)-(b): Laplacian compression. (c)-(d): Proposed method

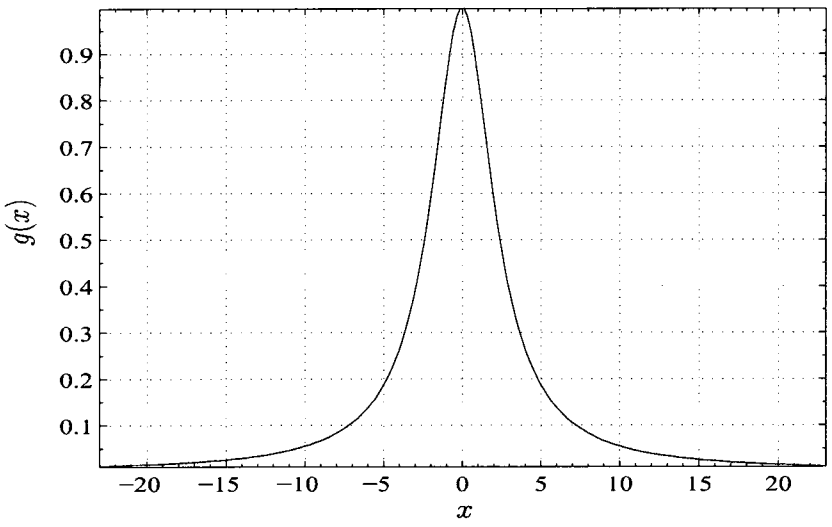


Figure 3.8: Cauchy weight function with $c = 2.3849$.

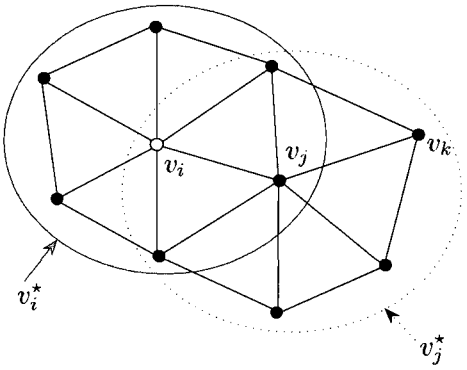


Figure 3.9: Illustration of two neighboring rings.

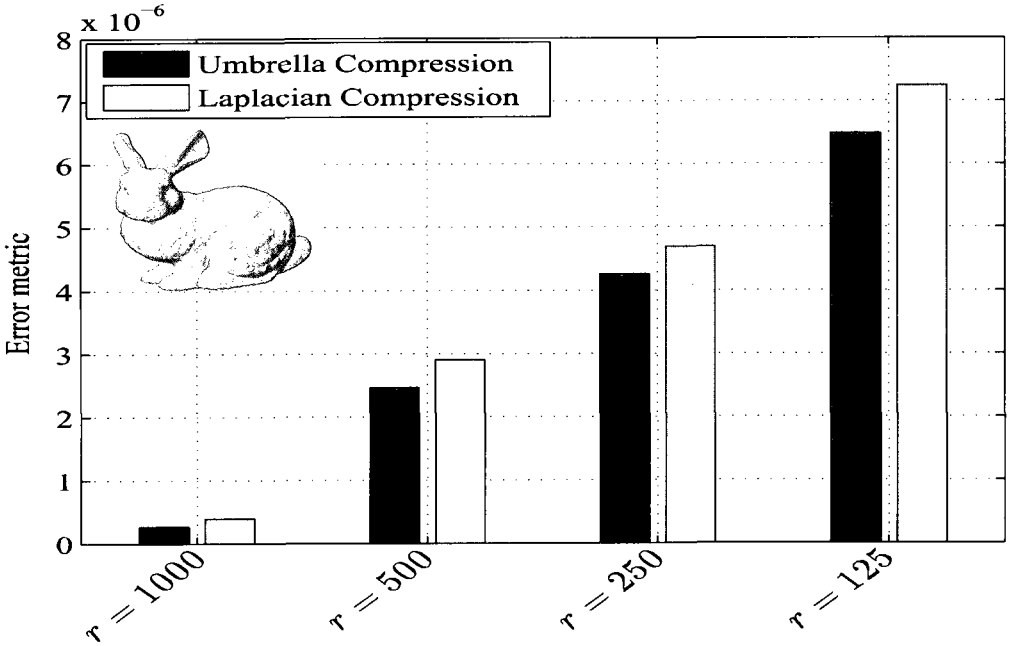


Figure 3.10: Compression error results for the 3-D bunny model.

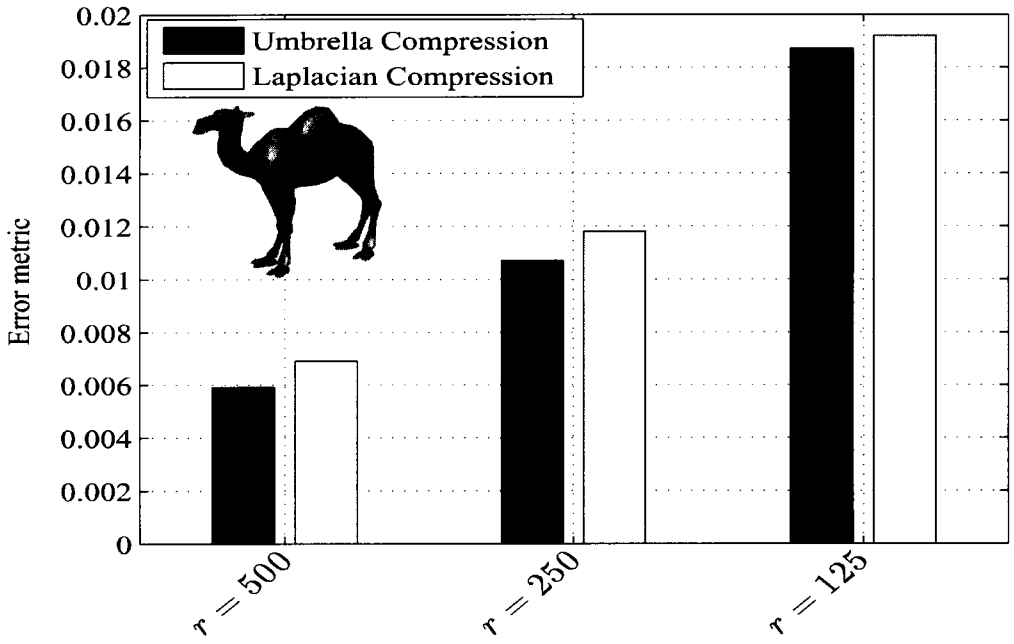


Figure 3.11: Compression error results for the 3-D camel model.

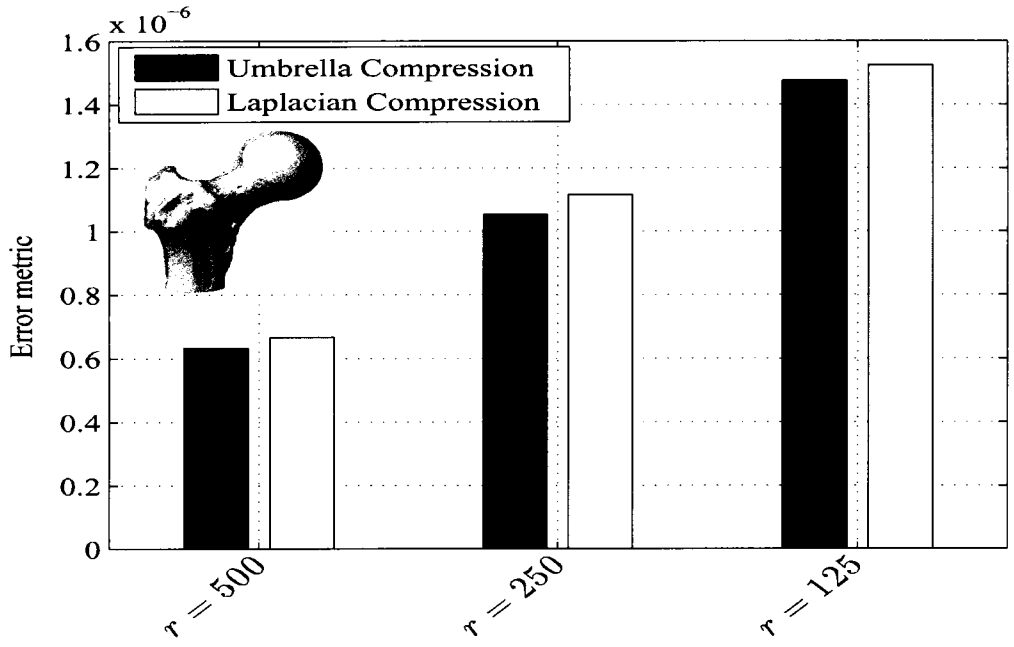


Figure 3.12: Compression error results for the 3-D femur model.

CHAPTER 4

3D Mesh Fingerprinting

4.1 Introduction

The increasing use of 3D models in multimedia applications and the wide demand of online services have opened the doors for users to modify the digital content without leaving any perceptual traces. To tackle this problem, cryptographic hash functions could help in ensuring the authentication and the integrity of data. Cryptographic hash functions play an important role in modern cryptography [34]. Hash functions take an input of arbitrary length to produce an output of fixed length referred to as hash.

The authenticity of the data can be verified by recalculating the hash value from the underlying data and comparing it to the attached hash value. Recently, Venkatesan *et al.* [35] introduced a method for robust image hashing. This technique uses randomized signal processing strategies for a non-reversible compression of images into random binary strings, and is shown to be robust against image changes that are due to compression, geometric distortions, and other attacks. Another robust image hashing technique was proposed in [36], which presents a framework for perceptual image hashing using feature points. Using on the characteristics of the visual system, significant image features are extracted by using a wavelet-based feature detection algorithm [37]. This hash algorithm withstands standard benchmark attacks and common signal processing operations. In [38], a novel algorithm for generating an image hash based on Fourier transform

features and controlled randomization was proposed. This scheme shows its resiliency to content-preserving modifications.

The problem of 3D mesh hashing is relatively new compared to 2D hashing and has received less attention partly because the technology that has been used for image and video analysis cannot be easily adapted to 3D objects. Also, a large number of attacks can be applied to 3D meshes. In [29], the mesh Laplacian matrix was used to encode the 3D shape into a more compact representation by retaining the smallest eigenvalues and associated eigenvectors which contain the highest concentration of the shape information. In [39], an geometric hashing method for object recognition was presented. This method identifies objects in the presence of noise and partial occlusion. In [40], a public authentication of 3D mesh models was presented. The signature is embedded within the 3D mesh model for authentication purposes, and a new hash value is produced and compared with the value decrypted from the retrieved signature.

The primary motivation behind the proposed method is to encode the geometric and topological information of a 3D object into a hash value that may be used for a variety of rightful ownership protection purposes including authentication and integrity. Our approach partitions a 3D triangle mesh into sub-meshes and produces a hash value for each sub-mesh. To gain further insight into the proposed technique, we performed extensive numerical experiments to demonstrate the potential and the much improved performance of the proposed scheme in 3D object authentication.

The layout of this chapter is as follows. Section 4.2 is devoted to the problem formulation, followed by a brief background material about Laplace-Beltrami matrix and entropic spanning trees. Section 4.3 describes the algorithmic steps of the proposed approach. In section 4.4, we present experimental results to show the performance of the proposed method and its robustness against the most common attacks.

4.2 Problem Formulation

The objective of 3D hashing is to design a robust hash function that produces a unique identifier for a 3D model, while satisfying three main requirements [34]. First, given a 3D model $\mathbb{M}$ and a

hash function H, the computation of the hash value $h = H(\mathbb{M})$ must be easy. Second, Given h, it is hard to find a 3D model $\widetilde{\mathbb{M}}$ such that $h = H(\widetilde{\mathbb{M}})$. Third, two different 3D models should not produce the same hash value.

4.2.1 Laplace-Beltrami matrix of a triangle mesh

Let $\boldsymbol{v}_i \in \mathcal{V}$, the Laplace-Beltrami operator $\Delta_m \boldsymbol{v}_i$ is defined as

$$\Delta_m \boldsymbol{v}_i = \frac{3}{\eta} \sum_{\boldsymbol{v}_j \in \boldsymbol{v}_i^*} (\cot \alpha_{ij} + \cot \beta_{ij})(\boldsymbol{v}_j - \boldsymbol{v}_i), \tag{1}$$

where α_{ij} and β_{ij} are the angles $\angle \boldsymbol{v}_i \boldsymbol{v}_{j-1} \boldsymbol{v}_j$ and $\angle \boldsymbol{v}_i \boldsymbol{v}_{j+1} \boldsymbol{v}_j$ respectively (see Figure 4.1), and $\eta = \sum_{\boldsymbol{t}_j \in \mathcal{T}(\boldsymbol{v}_i^*)} area(\boldsymbol{t}_j)$.

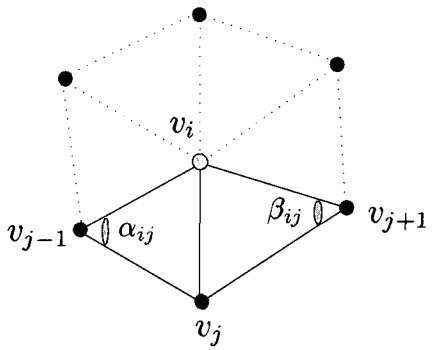


Figure 4.1: Illustration of Laplace-Beltrami angles α_{ij} and β_{ij}.

We may define the Laplace-Beltrami matrix as

$$L = D - W = \begin{cases} d_i - \omega_{ii} & \text{if } \boldsymbol{v}_i = \boldsymbol{v}_j \\ -\omega_{ij} & \text{if } \boldsymbol{v}_i \sim \boldsymbol{v}_j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $W = (w_{ij})$ denotes the weighted adjacency matrix with $w_{ij} = 3(\cot \alpha_{ij} + \cot \beta_{ij})/\eta$, and $D = \text{diag}\{d_i : \boldsymbol{v}_i \in \mathcal{V}\}$ is the degree matrix with diagonal entries $d_i = \sum_{\boldsymbol{v}_j \in \boldsymbol{v}_i^*} \omega_{ij}$.
Figure 4.2 illustrates an example of a 3D triangle mesh and its sparse Laplace-Beltrami matrix.

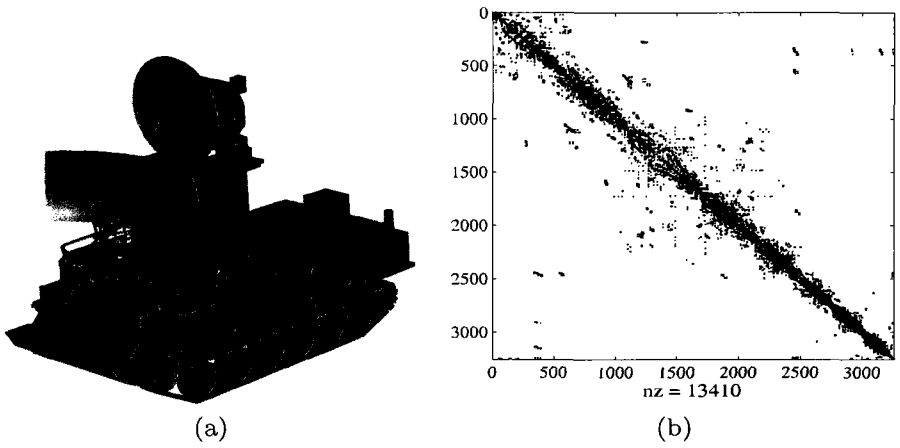


Figure 4.2: (a) 3D tank model and its (b) Laplace-Beltrami matrix.

4.2.2 Minimal spanning tree

A spanning tree is a connected acyclic graph that passes through all the elements of the vertex set $\mathcal{V}$, and it is specified by an ordered list of edges e_{ij} connecting certain pairs $(\boldsymbol{v}_i, \boldsymbol{v}_j), i \neq j$, along with a list of edge adjacency relations [41].

Recently, there has been a concerted research effort in statistical physics to explore the properties of Tsallis entropy, leading to a statistical mechanics that satisfies many of the properties of the standard theory [42]. When a system is composed of two statistically independent subsystems, then Shannon or Rényi entropy of the composite system is just the sum of entropies of the individual systems, and hence the correlations between the subsystems are not accounted for. Tsallis entropy, however, does take into account these correlations due to its pseudo-additivity property [42].
We may define an estimator $\widehat{H}_\alpha$ of Tsallis entropy as follows

$$\widehat{H}_\alpha(\mathcal{V}) = \frac{1}{1-\alpha}\left[\frac{L^\star(\mathcal{V})}{\beta m^\alpha} - 1\right], \tag{3}$$

where $L^\star(\mathcal{V})$ is the total length of the minimal spanning tree [41], α is referred to as an entropic index, and β is a constant playing a role of bias correction [43]. We employ Kruskal's algorithm [41] to compute the minimal spanning tree. Figure 4.3 shows two 3D models and their minimal spanning trees.

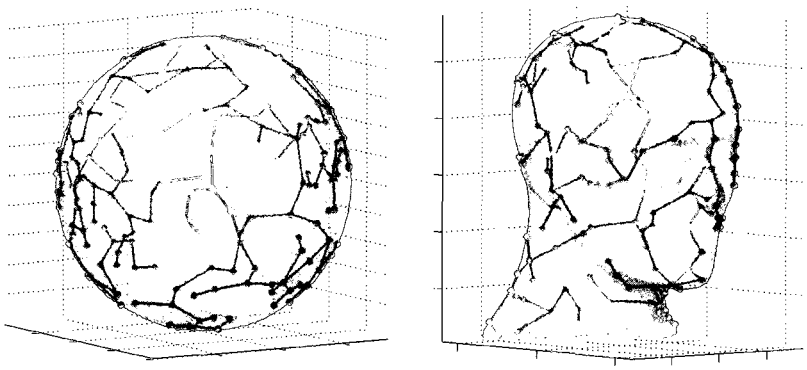


Figure 4.3: 3D models and their minimal spanning trees.

4.3 Proposed Hashing Method

In this section, we describe the main steps of the proposed 3D object hashing methodology.

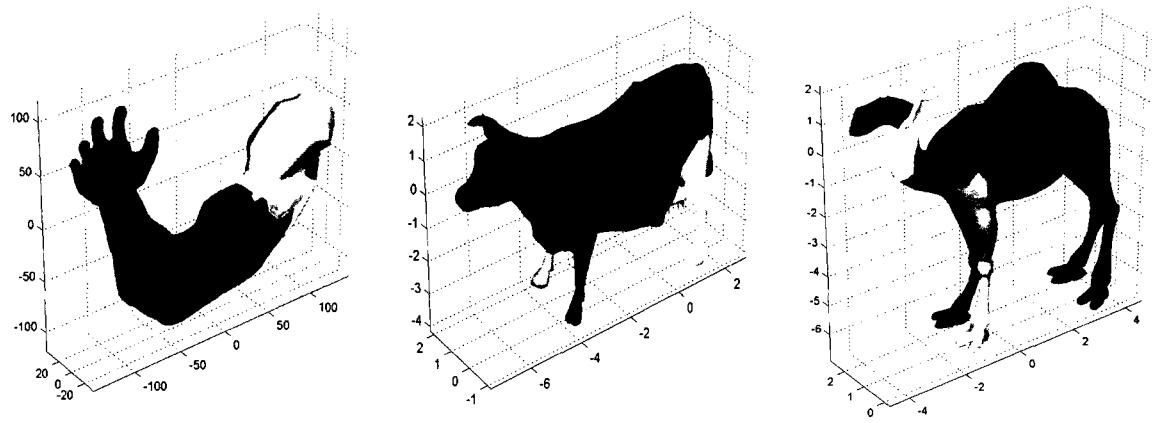
4.3.1 Mesh partitioning

The proposed algorithm is based on the concepts of entropic spanning trees and the eigen-decomposition of the Laplace-Beltrami matrix. However, calculating of the eigenvalues and the eigenvectors of a typically large $m \times m$ Laplace-Beltrami matrix is prohibitively expensive $\mathcal{O}(m^3)$. To circumvent this limitation, we first partition the original 3D mesh into sub-meshes and then apply eigen-analysis to each sub-mesh. To this end, we implemented a 3D mesh partitioning algorithm based on MeTiS [44–46]. Figure 4.4 shows some 3D models partitioned into eight sub-meshes.

4.3.2 Proposed algorithm

The algorithmic steps of the proposed 3D hashing approach may be summarized as follows:

1) Partition a 3D object $\mathbb{M}$ into s sub-meshes: $\mathbb{M} = \cup_{k=1}^{s} \mathbb{M}_k$, where the cardinality of the vertex set $\mathcal{V}_k$ of each sub-mesh $\mathbb{M}_k$ is equal to $|\mathcal{V}_k| = m_k$.



(a) (b) (c)

Figure 4.4: 3D mesh partitioning: each sub-mesh is colored randomly. (a) 3D arm model, (b) 3D cow model, and (c) 3D camel model.

1.1) Apply Kruskal's algorithm to each $\mathcal{V}_k$ in order to compute the entropy value ξ_k of each sub-mesh

$$\xi_k \simeq 2\left[\frac{L^{\star}(\mathcal{V}_k)}{\sqrt{m_k}} - 1\right], \quad k = 1, \ldots, s$$

where the Tsallis entropic index α is set to 1/2.

1.2) Apply eigen-decomposition to the Laplace-Beltrami matrix L_k of each sub-mesh, that is $L_k = B_k \Lambda_k B_k^T$, where $B_k = (\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{m_k})$ is an orthogonal matrix whose columns are eigenvectors which form a spectral basis, and $\Lambda_k = \mathrm{diag}(\lambda_i : i = 1, \ldots m_k)$ is a diagonal matrix of eigenvalues arranged in decreasing order of magnitude.

1.3) Retain the r ($r < m_k$) significant spectral basis vectors which account for most of the energy.

1.4) Compute the hash value μ_k of each sub-mesh according to the formula given by $\mu_k = \sum_{i=1}^{r} \lambda_i^{\xi_k} \|\boldsymbol{b}_i\|^2$

2) Stack the hash values of all the sub-meshes into a single vector $\boldsymbol{h} = (\mu_1, \mu_2 \ldots, \mu_s)$, which we refer to as the hash vector.

4.4 Experimental Results

We tested the performance of the proposed hashing method using a variety of 3D models. Figure 4.5 shows a sample of six 3D objects.

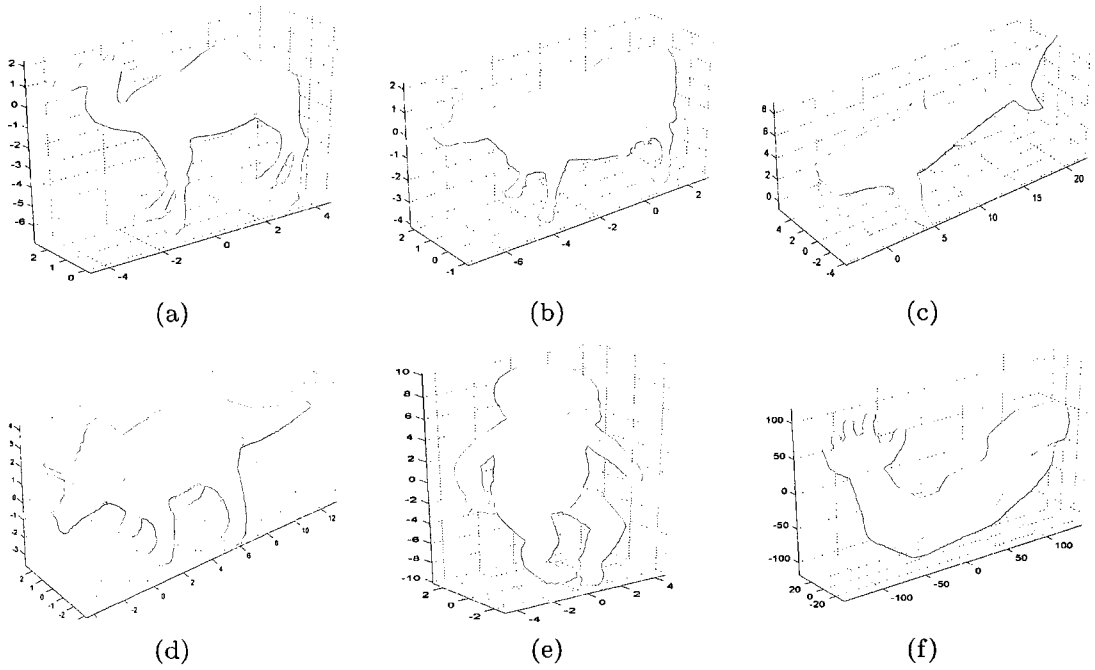


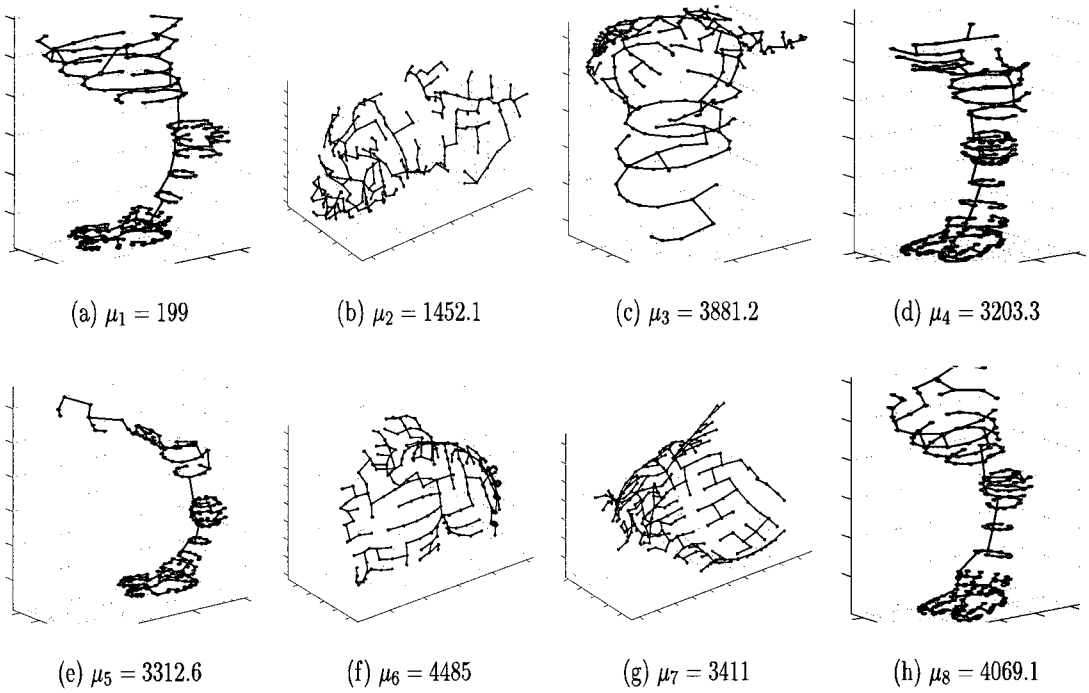
Figure 4.5: 3D models used for experimentation: (a) Camel, (b) Cow, (c) Shark, (d) Triceratops, (e) Baby, (f) Arm.

Figure 4.6(a)-(h) and Figure 4.7(a)-(h) depict the minimal spanning trees for each sub-mesh of the 3D camel and cow models respectively, as well as their corresponding hash values μ_k, where $k = 1, \ldots, 8$.

To test the robustness of the proposed hashing algorithm, we applied several attacks to the 3D models including scaling, rotation, mesh smoothing, mesh simplification, and Gaussian noise, as shown in Figure 4.8 and Figure 4.9. We evaluated the performance of the proposed scheme by computing the normalized correlation ρ between the resulting hash vectors as follows

$$\rho = \frac{|\boldsymbol{h}_1 \cdot \boldsymbol{h}_2|}{\|\boldsymbol{h}_1\| \, \|\boldsymbol{h}_2\|}, \tag{4}$$

where $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ are the hash vectors before and after the attack respectively. The correlation



(a) $\mu_1 = 199$ (b) $\mu_2 = 1452.1$ (c) $\mu_3 = 3881.2$ (d) $\mu_4 = 3203.3$

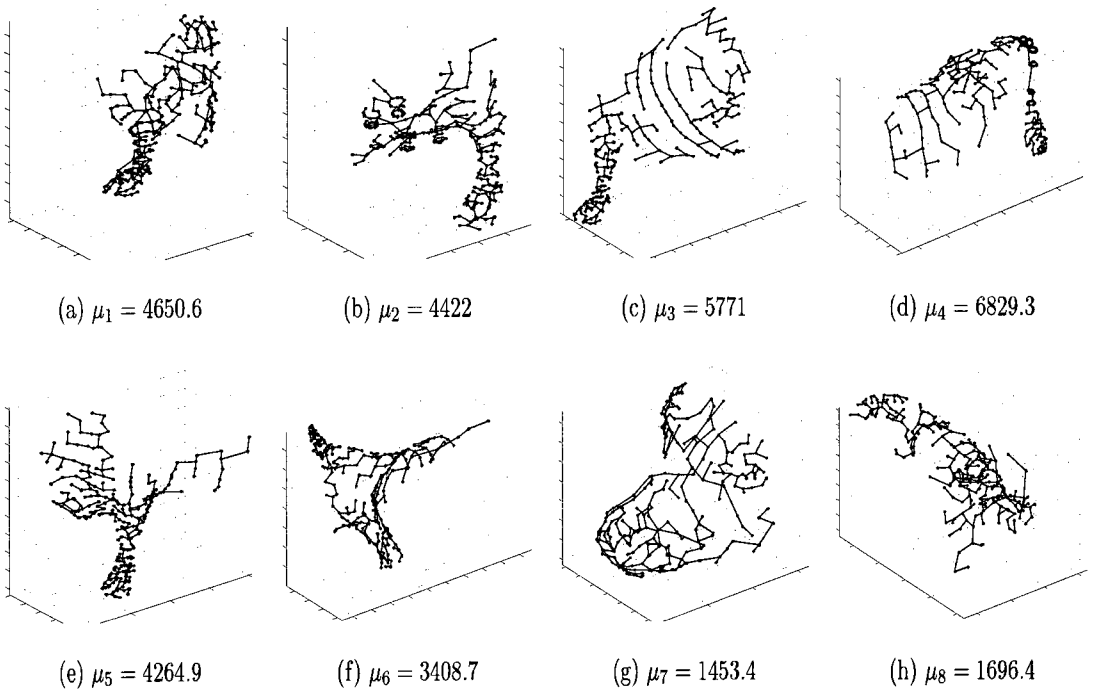
(e) $\mu_5 = 3312.6$ (f) $\mu_6 = 4485$ (g) $\mu_7 = 3411$ (h) $\mu_8 = 4069.1$

Figure 4.6: Minimal spanning trees of the 3D camel sub-meshes and their corresponding hash values: (a) head, (b) neck, (c)-(d) front feet, (e) hump, (f) shoulders, (g)-(h) back.

results shown in Table 4.1 clearly demonstrate the good performance of the proposed method in terms of robustness against the attacks. Moreover, this good performance is in fact consistent with all the 3D models used for experimentation.

4.4.1 Uniqueness

Uniqueness is an important factor that needs to be take into consideration when dealing with hash functions. As mentioned earlier, the hash value produced by our proposed method should be unique. Therefore, we compared the hash vectors between different 3D models using the normalized correlation coefficient to check whether the proposed hash vector fulfills the requirement of uniqueness. The results are listed in Table 4.2. It is apparent that the proposed method shows a very good performance in terms of the ability to distinguish between different 3D models and also to produce different hash values.



(a) $\mu_1 = 4650.6$ (b) $\mu_2 = 4422$ (c) $\mu_3 = 5771$ (d) $\mu_4 = 6829.3$

(e) $\mu_5 = 4264.9$ (f) $\mu_6 = 3408.7$ (g) $\mu_7 = 1453.4$ (h) $\mu_8 = 1696.4$

Figure 4.7: Minimal spanning trees of the 3D cow sub-meshes and their corresponding hash values: (a)-(b) back feet, (c)-(e) front feet, (d) back-tail, (f) neck, (g)-(h) head-horn.

Table 4.1: Normalized hash correlation results for different 3D models.

Attacks	3D Models					
	Camel	Cow	Shark	Triceratops	Baby	Arm
x-axis scaling	0.9674	0.9908	0.7539	0.7755	0.5907	0.7704
y-axis scaling	0.9625	0.9910	0.7565	0.7864	0.6017	0.7784
z-axis scaling	0.9553	0.9221	0.7433	0.7637	0.5916	0.7673
x-axis rotation 45°	0.9534	0.9880	0.7486	0.7807	0.5864	0.7812
y-axis rotation 45°	0.9534	0.9880	0.7486	0.7807	0.5864	0.7812
z-axis rotation 45°	0.9534	0.9880	0.7486	0.7807	0.5864	0.7812
Mesh smoothing	0.9531	0.9886	0.7343	0.7765	0.5922	0.7643
Mesh simplification (70%)	0.7965	0.8448	0.7247	0.7641	0.8862	0.7621
Gaussian Noise ($\sigma = 0.25$)	0.9572	0.9893	0.7606	0.7931	0.6153	0.7886

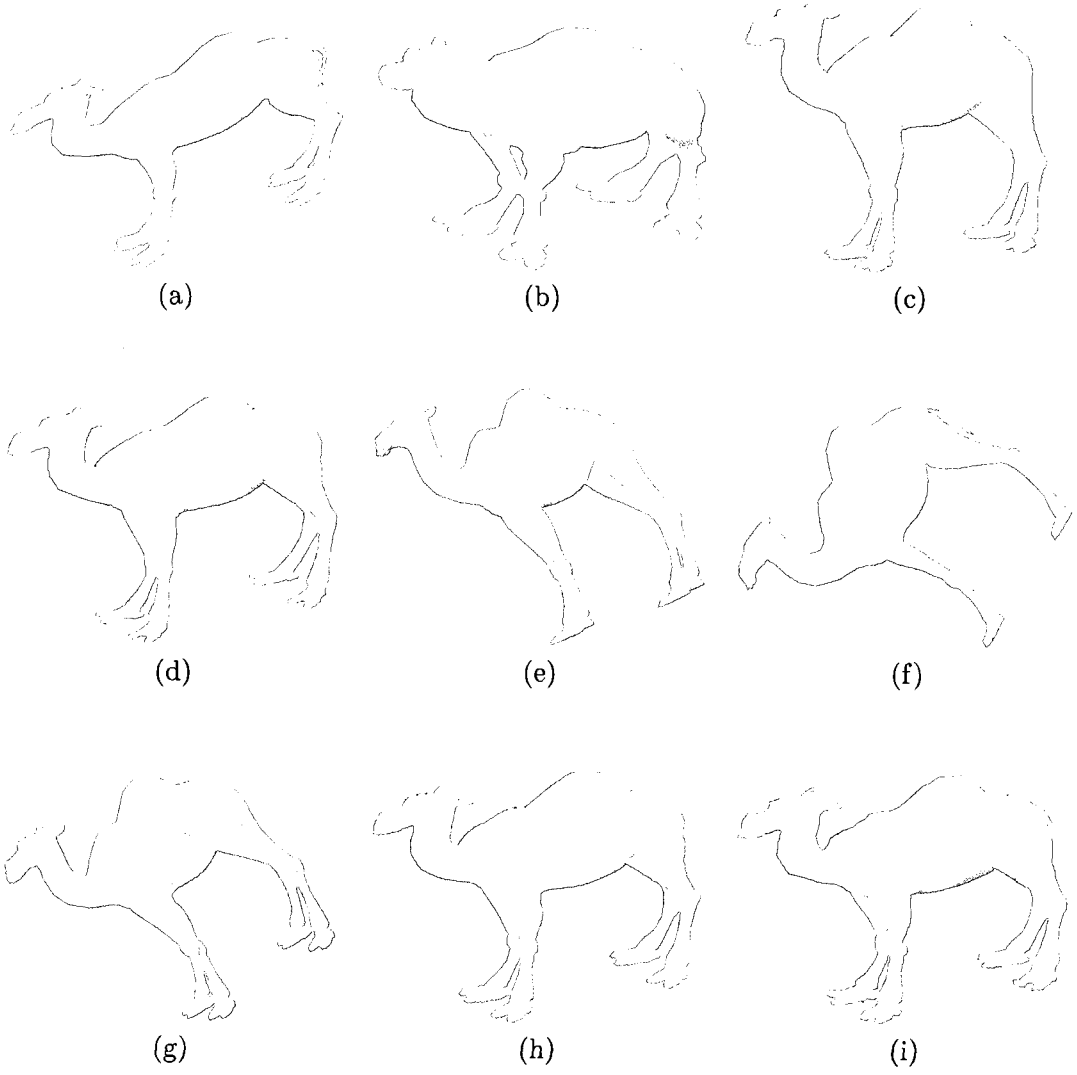


Figure 4.8: 3D camel model under different attacks: (a) x-axis scaling, (b) y-axis scaling, (c) z-axis scaling, (d) Laplacian mesh smoothing using 10 iterations, (e) rotation around x-axis by 45^o, (f) rotation around y-axis by 45^o, (g) rotation around z-axis by 45^o, (h) mesh simplification 70%, and (i) additive Gaussian noise with $\sigma = 0.25$.

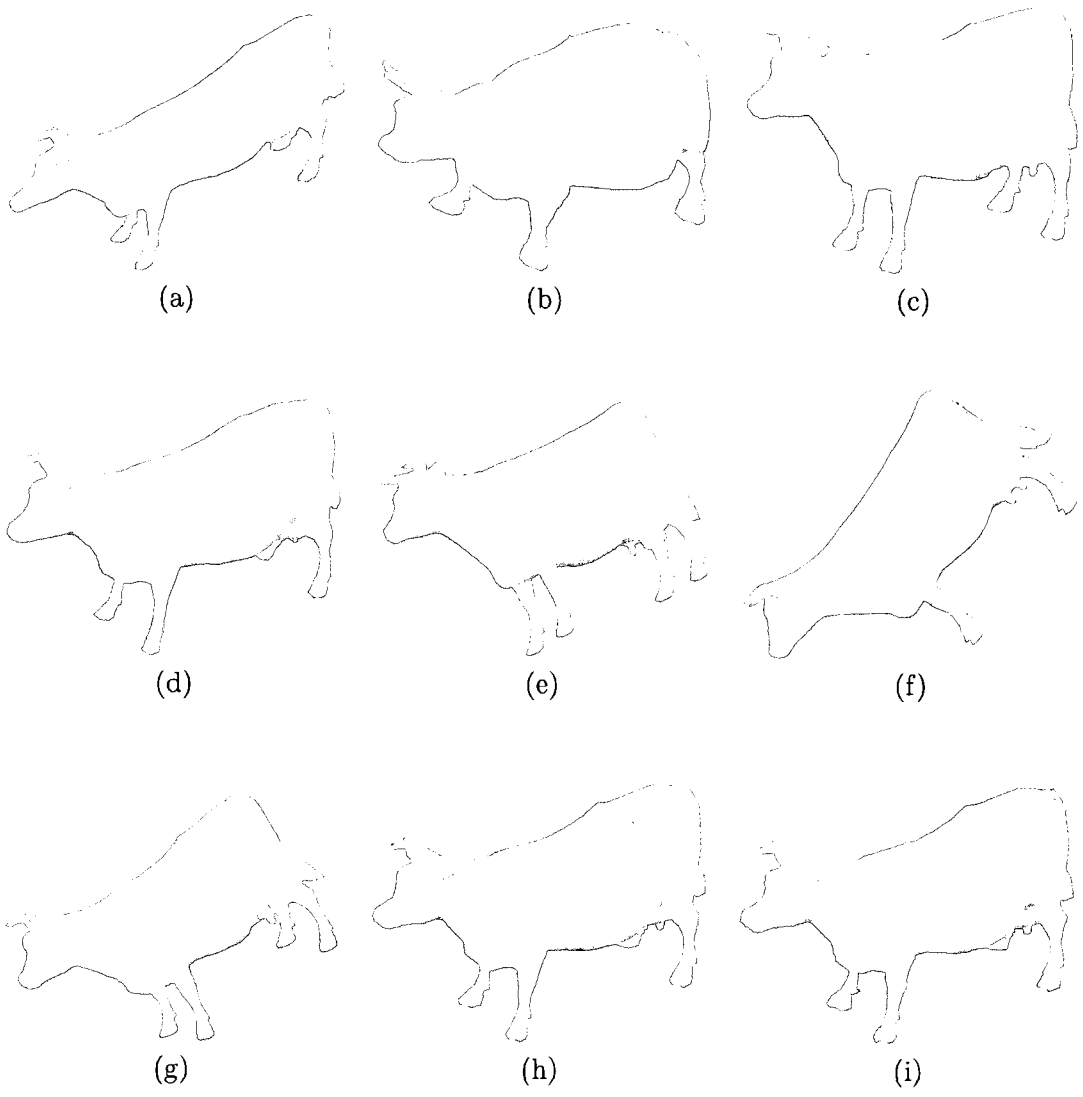


Figure 4.9: 3D cow model under different attacks: (a) x-axis scaling, (b) y-axis scaling, (c) z-axis scaling, (d) Laplacian mesh smoothing using 10 iterations, (e) rotation around x-axis by 45^o, (f) rotation around y-axis by 45^o, (g) rotation around z-axis by 45^o, (h) mesh simplification 70%, and (i) additive Gaussian noise with $\sigma = 0.25$.

Table 4.2: Normalized correlation coefficients results between different 3D hashed models.

	Camel	Cow	Shark	Triceratops	Baby	Arm
Camel	1	0.7987	0.7458	0.7668	0.8112	0.7834
Cow	0.7987	1	0.7842	0.8590	0.8855	0.7921
Shark	0.7458	0.7842	1	0.7816	0.5129	0.7172
Triceratops	0.7668	0.8590	0.7816	1	0.7179	0.7145
Baby	0.8112	0.8855	0.5129	0.7179	1	0.8328
Arm	0.7834	0.7921	0.7172	0.7145	0.8328	1

CHAPTER 5

Statistical 3D Mesh Distributions

In this chapter we introduce several statistical distributions to analyze the topological properties of 3D objects. The proposed statistical measures include the mesh degree, the mesh assortativity, the mesh clustering coefficient, and the mesh geodesic distance distributions. These probabilistic distributions provide useful information about the way 3D mesh models are connected. Illustrating experimental results show the effectiveness of the proposed measures in quantifying the topological features of 3D objects.

5.1 Introduction

Shape analysis of 3D objects has become an active research field with the recent developments in solid modeling and visualization [48]. Nowadays, vast amounts of 3D models are being developed and are distributed freely or commercially on the internet. 3D graphics are commonly used in several multimedia applications such as video gaming, engineering design, virtual reality, e-commerce and scientific visualization. 3D objects consist of geometric and topological information. Topology is the property that determines which parts of the shape of objects are connected to which other parts, while geometry determines where, in a given coordinate system, each part is located. The topological properties of 3D objects in the statistical framework are the focus of this chapter, and the main objective is to convert the 3D mesh data into useful and meaningful information. To convert the 3D mesh data into information, we need appropriate probabilistic tools

and techniques [47, 49–51]. Such statistical methods will help us quantify the mesh topological properties, and to also gain very useful information by presenting the same data graphically. Even with modern 3D computer graphics tools, we cannot draw a meaningful conclusion about a large 3D model or graph using direct examination by the human eye [49]. This examination analysis can be done only by statistical methods for quantifying the connectivity information of large graphs. Generally, graph-theoretic representations of a problem are based towards highlighting aspects such as the connectivity of components in the problem. In recent years, we have witnessed a substantial development in graph theory research, with the attention shifted from the analysis of small graphs and the properties of individual vertices or edges within such graphs to the consideration of large-scale statistical properties of graphs [49]. This change has been driven fast by the availability of powerful computers and communication networks that allow us to gather and analyze data for larger 3D mesh models.

In this chapter, we introduce some statistical distributions to analyze the connectivity information of 3D mesh models. The proposed distributions are defined in terms of the topological properties of the mesh. In particular, we introduce the vertex degree distribution, and the geodesic distance distribution which captures the nonlinearity structure and the intrinsic geometry of a 3D object. We also propose the clustering coefficient distribution which provide useful information about a 3D mesh and the way it is connected. Clustering techniques have been well studied in many applications such as data mining,image segmentation, pattern classification and text categorization in which a group of similar words characteristics are clustered in the same group. Recently a graph based-representation approach has been used [52], and the main idea is to analyze the human language acquisition by representing the set of terms as vertices and edges representing the relationship between them. Also, statistical measures have been recently studied to determine how well the internet topologies are connected in order to reflect many practical and theoretical network characteristics such as robustness of the network under attack.

The rest of this chapter is organized as follows. In Section 5.2 we propose some statistical distributions to study the topological properties of 3D mesh data and we explain in more details the fundamental aspects of each distribution. In Section 5.3, we provide experimental results to

show the usefulness of the proposed statistical measures in 3D graphics.

5.2 Proposed Statistical Measures

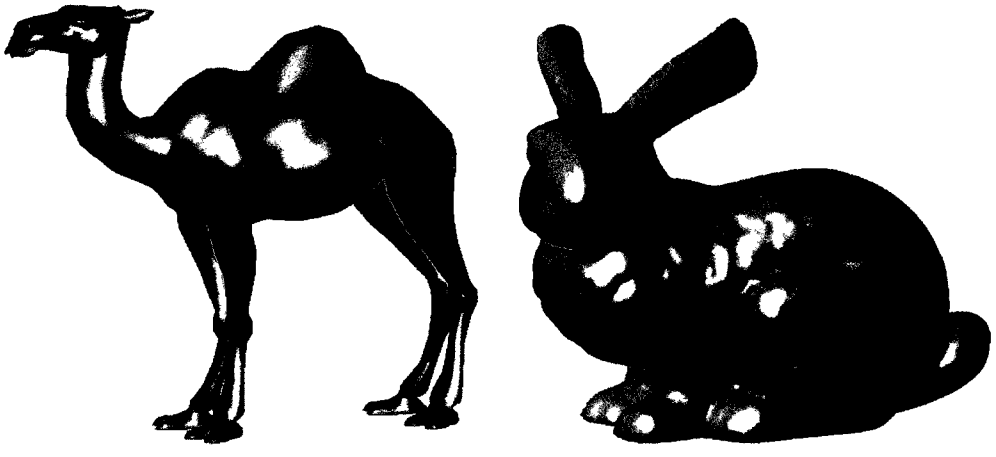
5.2.1 Average vertex degree

The average vertex degree of a triangle mesh $\mathbb{M}$ is the coarsest connectivity characteristic, and it is given by

$$\bar{d} = \frac{1}{m}\sum_{i=1}^{m} d_i = \frac{2|\mathcal{E}|}{m}. \quad (1)$$

The average vertex degree may be used as a measure of connectivity of a triangle mesh. We say that a triangle mesh is "better-connected" if it has a higher average vertex degree. Figure 5.1 shows an example of two triangle meshes with different average vertex degrees. Although the number of edges in the 3D bunny model is about 14.2 times larger than the 3D camel model, we found out, however, that the 3D camel has a much higher average vertex degree and hence it is much better-connected than the 3D bunny.

The average vertex degree, however, has limited utility since triangle meshes with the same average vertex degree may have vastly different topological structures.



(a) (b)

Figure 5.1: (a) 3D camel model, (b) 3D bunny model.

5.2.2 Mesh degree distribution

A more informative characteristic of graph connectivity is the vertex degree distribution, which provides information on the number of vertices in the triangle mesh having a certain degree. The vertex degree distribution is the probability that a randomly selected vertex has a degree equal to k, and it is given by

$$p_k = \frac{m_k}{m}, \tag{2}$$

where m_k is the number of vertices of degree k, that is m_k is the cardinality of the set $\mathcal{V}_k = \{\boldsymbol{v}_i \in \mathcal{V} : d_i = k\}$.

The degree distribution contains more information about the mesh connectivity than the average vertex degree since given a specific form of p_k we can always restore the average vertex degree by

$$\overline{k} = \sum_{k=1}^{k_{\max}} k p_k, \tag{3}$$

where $k_{\max}$ is the maximum vertex degree in the set $\mathcal{V}_k$.

5.2.3 Mesh assortativity distribution

An important concept that provides important information about the correlations of the degree of neighboring vertices is the average neighbor connectivity of each vertex, and it is given by

$$K_{nn}(\boldsymbol{v}_i) = \frac{1}{d_i} \sum_{\boldsymbol{v}_j \in \boldsymbol{v}_i^*} d_j, \quad \forall \boldsymbol{v}_i \in \mathcal{V}. \tag{4}$$

The assortativity function of a mesh is defined as

$$K_{nn}(k) = \frac{1}{m_k} \sum_{\boldsymbol{v}_i \in \mathcal{V}_k} K_{nn}(\boldsymbol{v}_i) \tag{5}$$

If degrees of neighboring vertices are uncorrelated, then the assortativity is a constant. When correlations are present, two main classes of possible correlations can been identified: assortative and disassortative. The triangle mesh has an assortative behavior if $K_{nn}(k)$ increases with k, indicating that large degree vertices are connected with other large degree vertices, and has an

disassortative behavior if $K_{nn}(k)$ is a decreasing function of k. The mesh assortativity distribution is defined as the histogram of the assortativity function.

5.2.4 Mesh clustering coefficient distribution

Clustering refers to the the process of organizing objects into groups whose members are similar in some way. In the triangle mesh framework, clustering may be used to measure how close the neighbors of any vertex are to form a group within the mesh. The clustering coefficient of each vertex is defined as

$$C(\boldsymbol{v}_i) = \frac{2}{d_i - 1}, \quad \forall \boldsymbol{v}_i \in \mathcal{V}, \tag{6}$$

and the higher clustering coefficient of the vertex is, the more interconnected its neighbors are. The average clustering coefficient of all vertices of degree k is given by

$$C(k) = \frac{1}{m_k} \sum_{\boldsymbol{v}_i \in \mathcal{V}_k} C(\boldsymbol{v}_i), \tag{7}$$

and the clustering coefficient distribution is defined as the histogram of $C(k)$.

5.2.5 Mesh geodesic distance distribution

The geodesic distance $g(\boldsymbol{v}_i, \boldsymbol{v}_j)$ between two mesh vertices $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ is the shortest length $L(\gamma) = \int_a^b \|\gamma'(t)\| dt$ of a smooth curve $\gamma : [a, b] \rightarrow \mathcal{M}$ such that $\gamma(a) = \boldsymbol{v}_i$ and $\gamma(b) = \boldsymbol{v}_j$. The primary motivation behind the geodesic distance is of overcoming the limitations of the Euclidean distance which by virtue of its linearity in nature cannot account for nonlinear structures in a 3D object. Unlike the Euclidean distance which is basically a straight line between two points in 3D space, the geodesic distance captures the global nonlinear structure and the intrinsic geometry of the data. The geodesic distance calculation is based on a similar approach used for computing the isometric feature mapping (Isomap) for multidimensional scaling [67] on nonlinear manifolds [53]. The algorithm has two main steps:

(i) Construct a neighborhood graph by connecting a given vertex to its k-nearest neighbors, and link these neighboring vertices by edges with weights equal to the Euclidean distances.

(ii) Compute the geodesic distances (shortest paths) between all pairs of n points in the constructed graph using Dijkstra's or Floyd's algorithm.

Note that Step (i) may be alleviated by choosing a random subset of $\mathcal{V}$ in order to reduce the computational complexity of the geodesic calculation.

5.3 Experimental Results

This section presents experimental results where the proposed statistical distributions are applied to 3D mesh models with different topologies and varying sizes as depicted in Figures 5.2(a)-5.19(a). For example, the 3D camel model contains 2443 vertices and 7326 edges, whereas the 3D bunny model consists of 34834 vertices and 104288 edges.

The Laplacian spectra of each model are depicted in Figures 5.2(b)-5.19(b). Spectral analysis is a powerful tool for investigating 3D structures such as discovering clusters of highly connected vertices. By performing the spectrum analysis, a wide range of critical graph characteristic can be discovered. The largest values are particularly important. Most graphs with high values for these largest eigenvalues have small diameter and contain strong clusters. In our experiments, we discovered using spectral analysis of the Laplacian matrix that the eigenvalues start at most for the lower normalized ranks which explains that all the 3D models are highly clustered on those lower normalized ranks.

By analyzing the mesh degree distributions of the 3D models as shown in Figures 5.2(c)-5.19(c), we can easily see that the probability of the degree increases as the vertex degree increases until it reaches the maximum with vertices of degree 6, then it starts decreasing as the vertex degree increases.

From the mesh assortativity plots shown in Figures 5.2(d)-5.19(d), we observe that the distribution of assortativity depends on the distribution of vertices on a 3D mesh. For example, the 3D camel, cow,feto, caballo, frog,hand,femur,foot and sculpture models have a strong dissortative behavior as illustrated in Figure 5.2(d), Figure 5.4(d), Figure 5.7(d), Figure 5.8(d), Figure 5.9(d), Figure 5.10(d), Figure 5.15(d), Figure 5.17(d) and Figure 5.19(d) but all the remaining models

show an assortative behavior for small vertex degrees, and a dissortative for large vertex degrees.

For the mesh clustering coefficient distributions shown in Figures 5.2(e)-5.19(e), the vertices of degree 3 have the highest coefficient value which is equal to 1, meaning that every neighbor connected to a vertex v_i of degree 3 is also connected to every other vertex within the neighborhood. Moreover, the clustering coefficient starts decreasing as the vertex degree increases until it reaches a lower clustering value for the highest vertices degree.

The mesh geodesic distance distributions as depicted in Figures 5.2(f)-5.19(f) clearly show that the nonlinear structure of each 3D object. For example, the geodesic distributions of the 3D camel and dragon models have a multimodal shape which clearly capture the humps in these models.

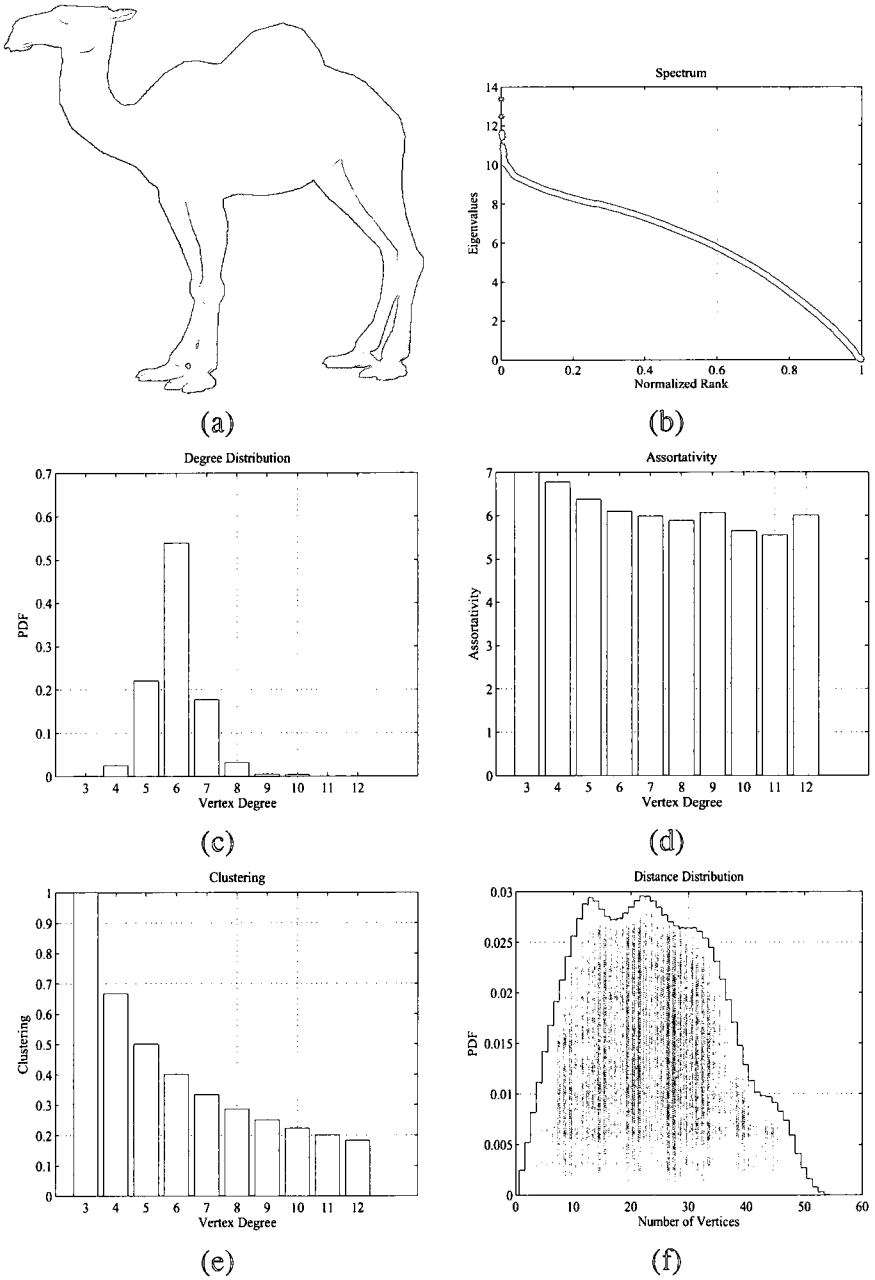


Figure 5.2: (a) 3D camel model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

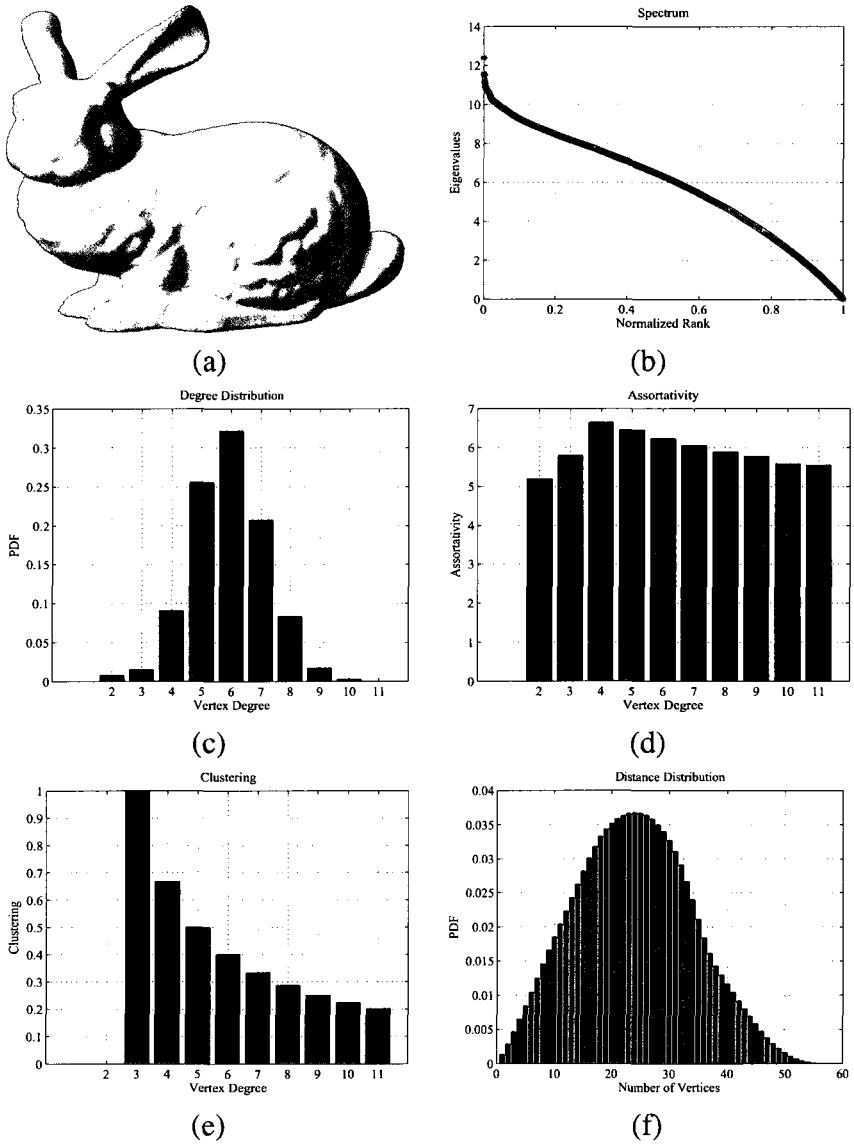


Figure 5.3: (a) 3D bunny model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

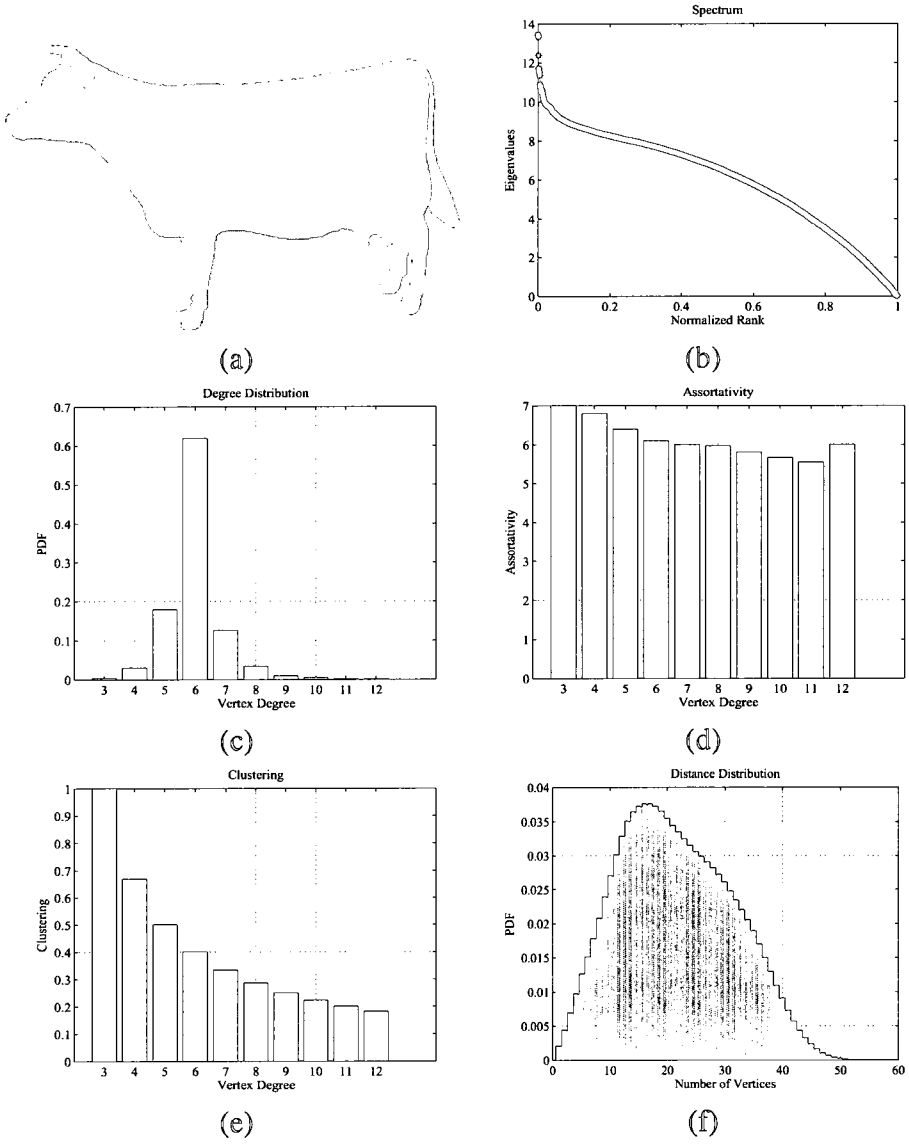


Figure 5.4: (a) 3D cow model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

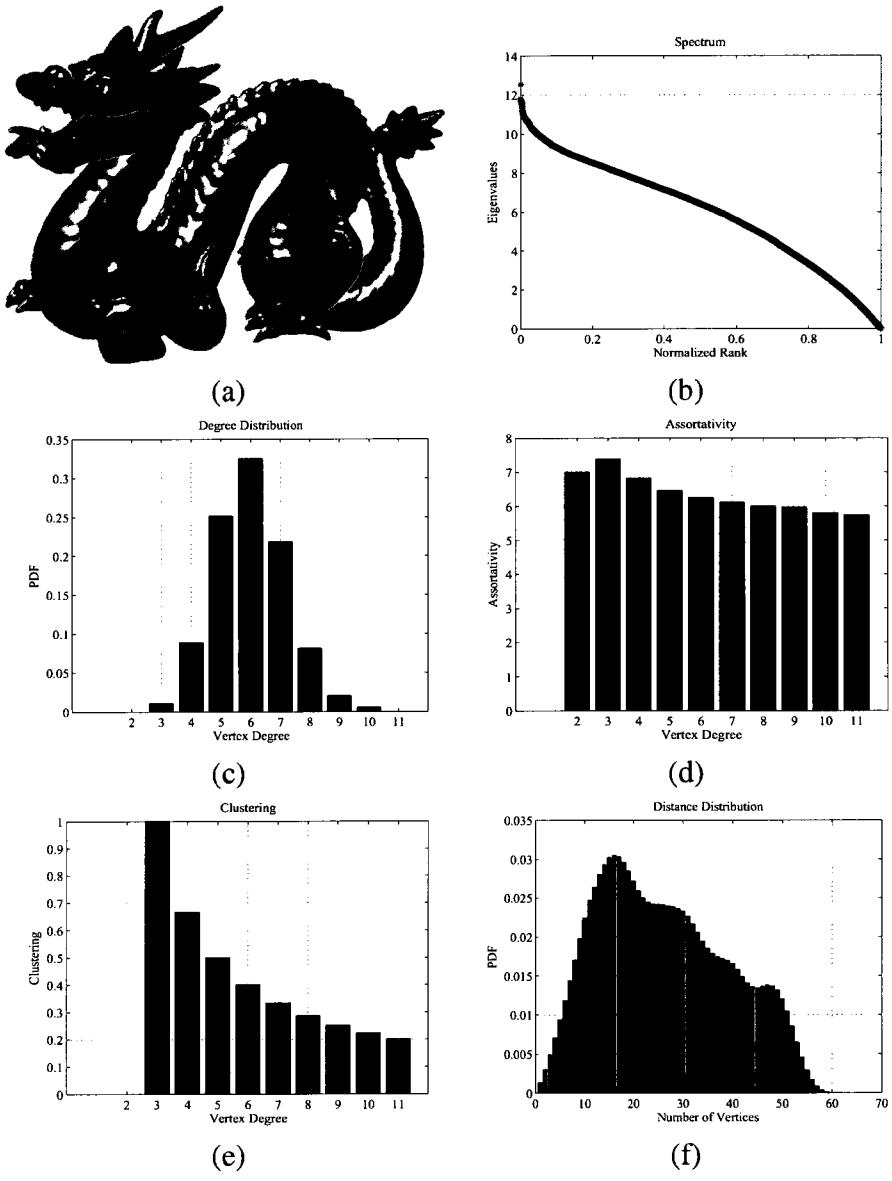


Figure 5.5: (a) 3D dragon model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

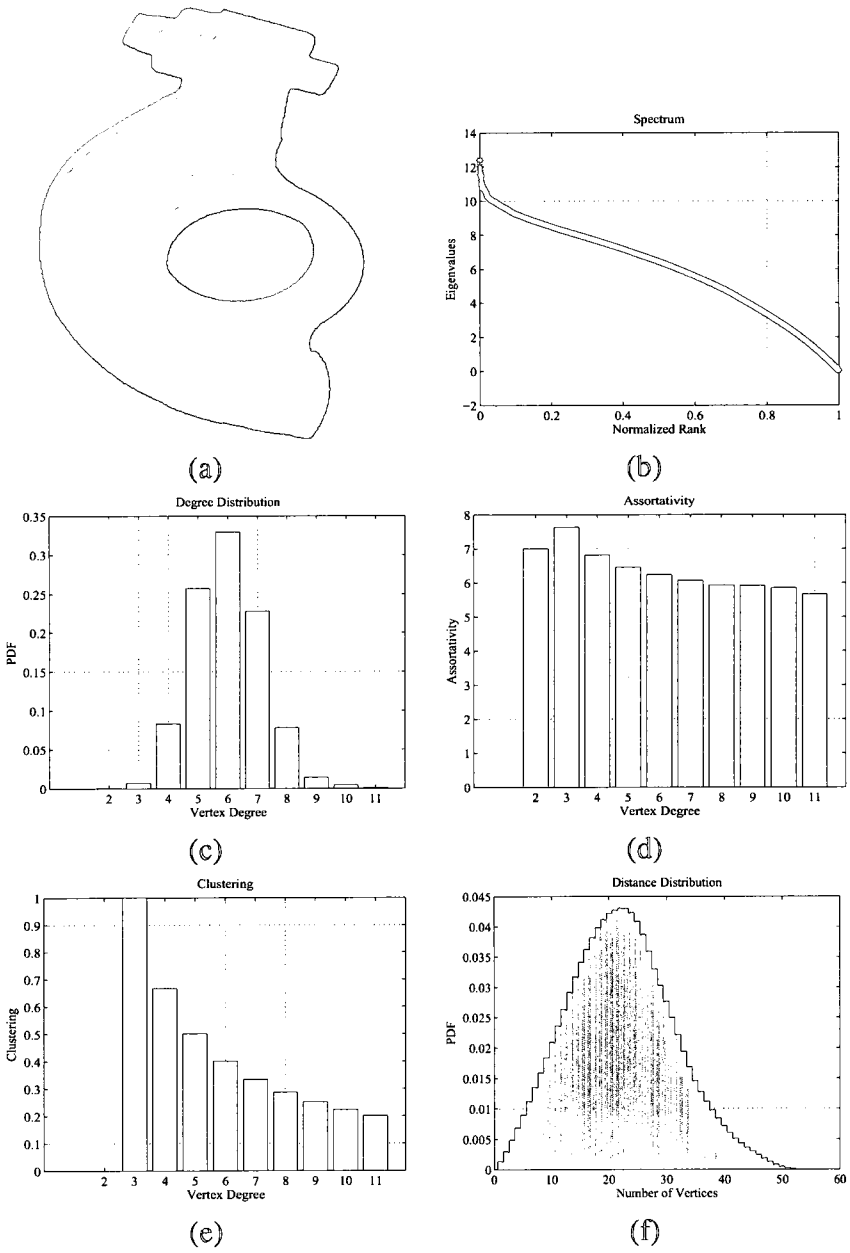


Figure 5.6: (a) 3D mechanical model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

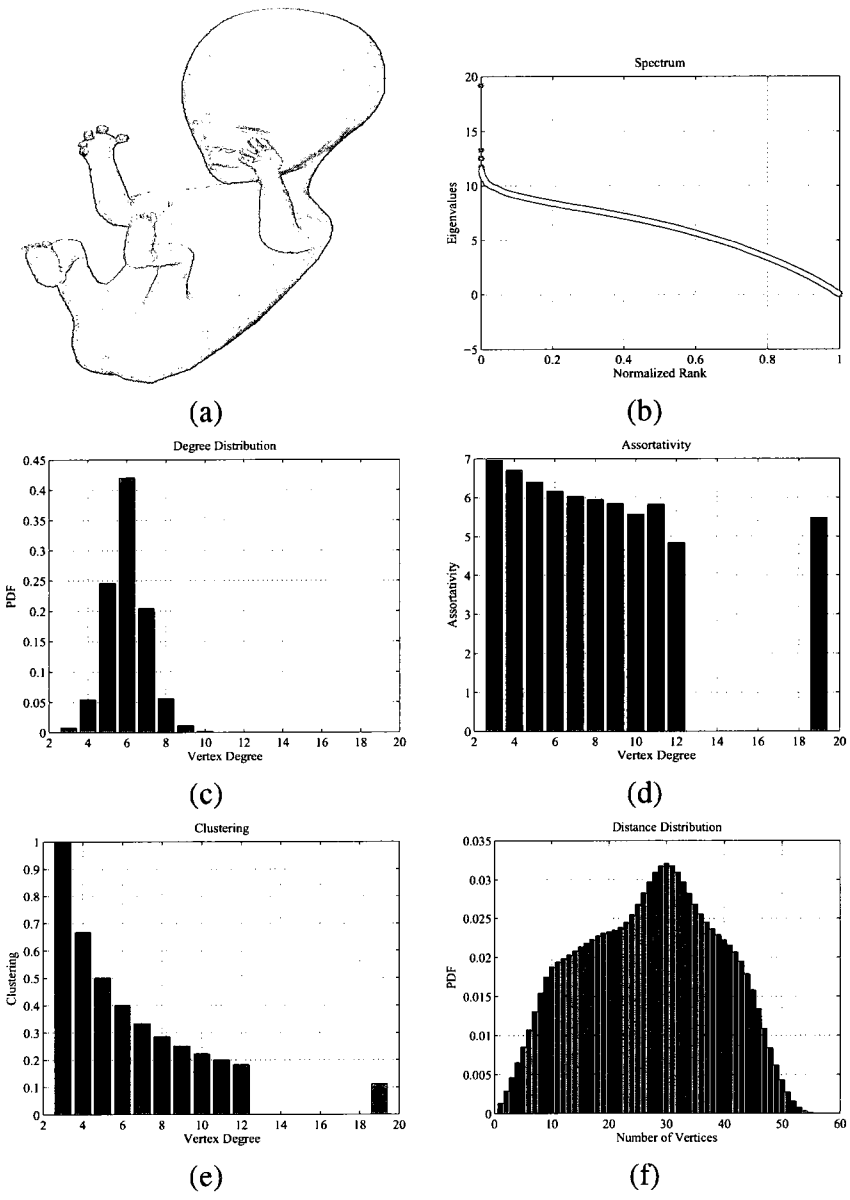


Figure 5.7: (a) 3D feto model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

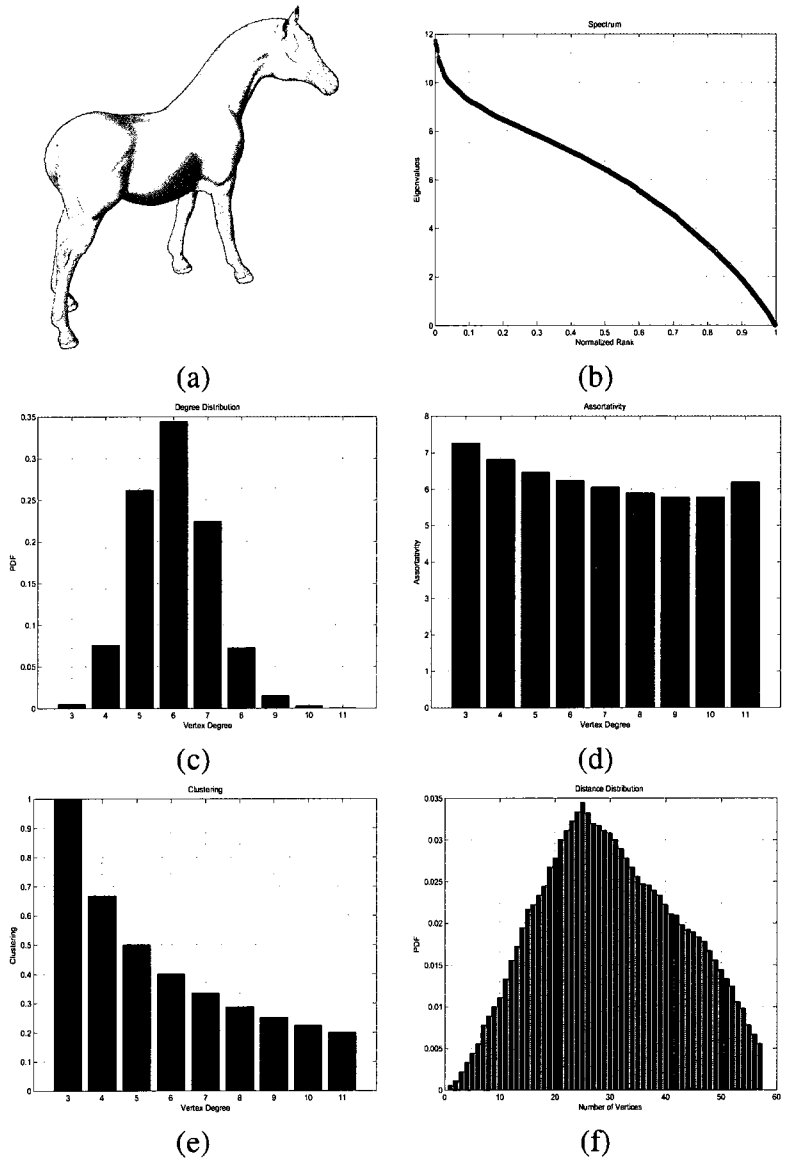


Figure 5.8: (a) 3D caballo model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

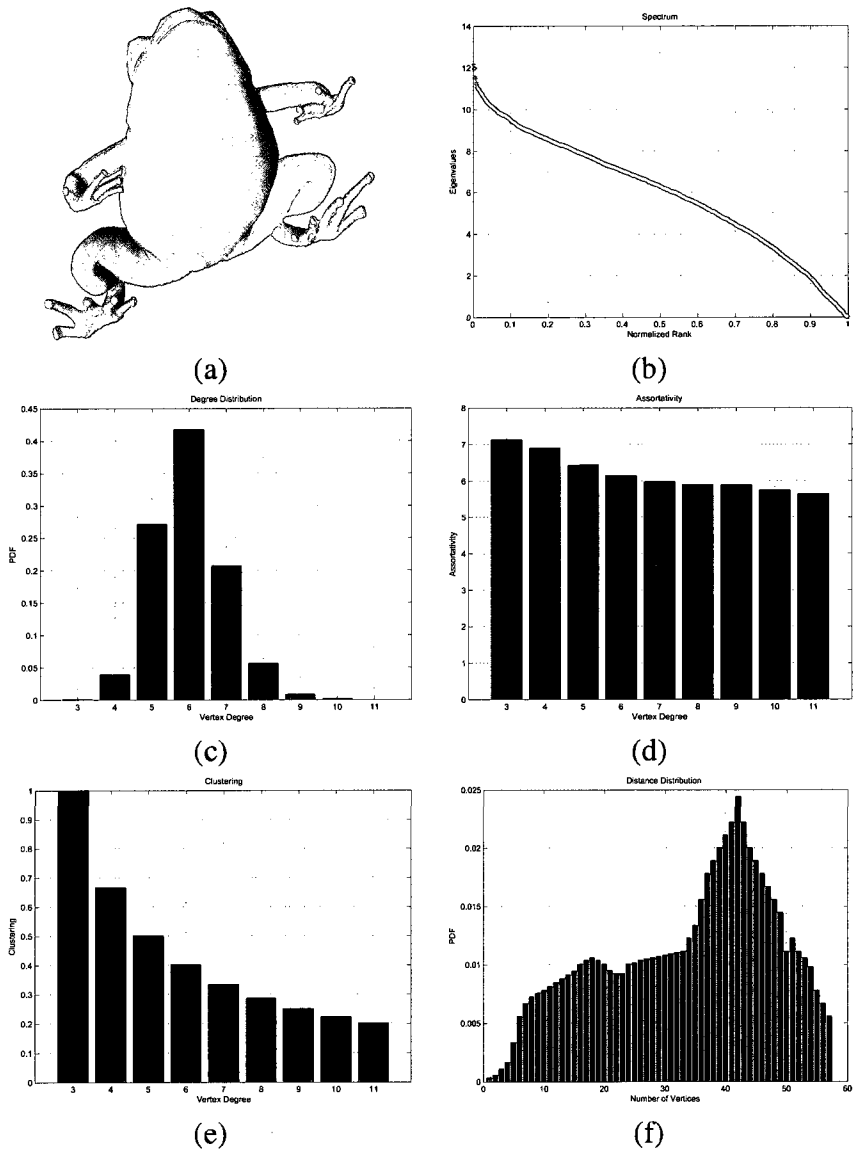


Figure 5.9: (a) 3D frog model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

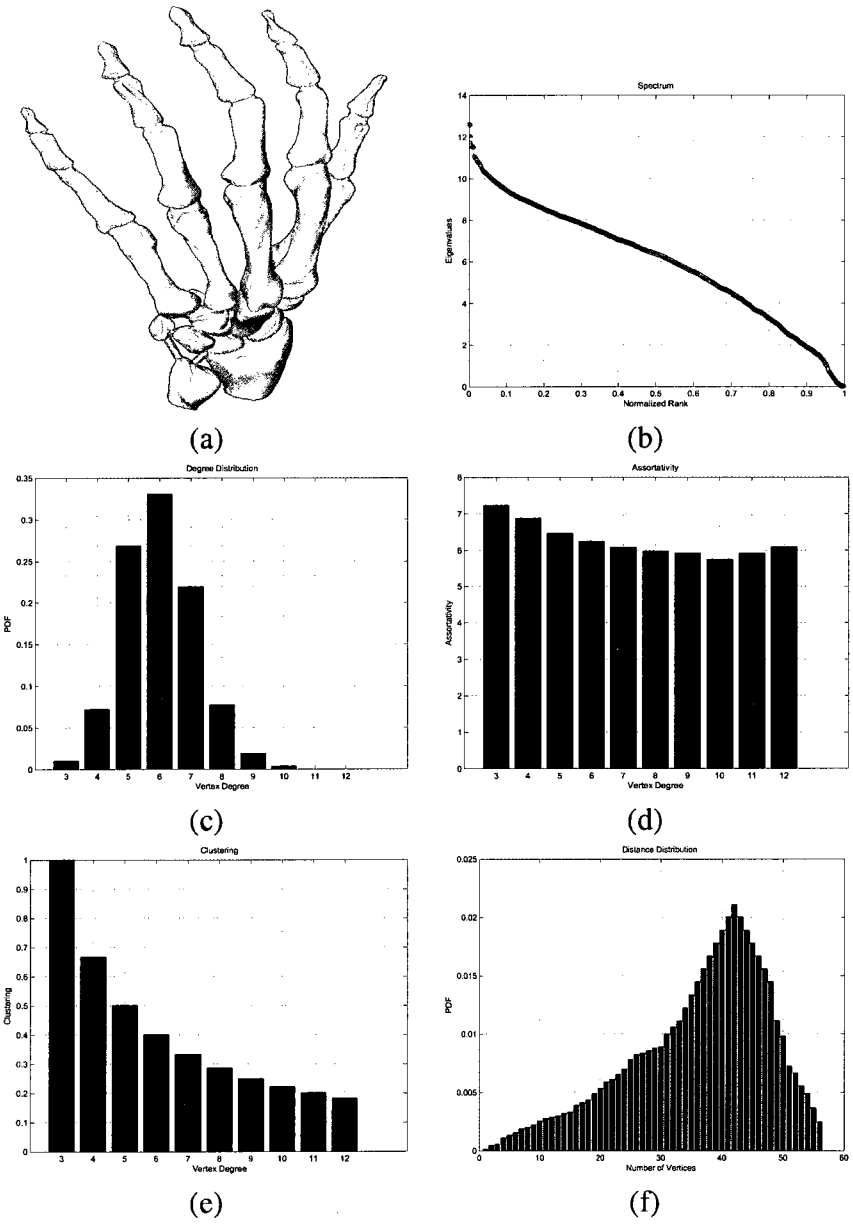


Figure 5.10: (a) 3D hand model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

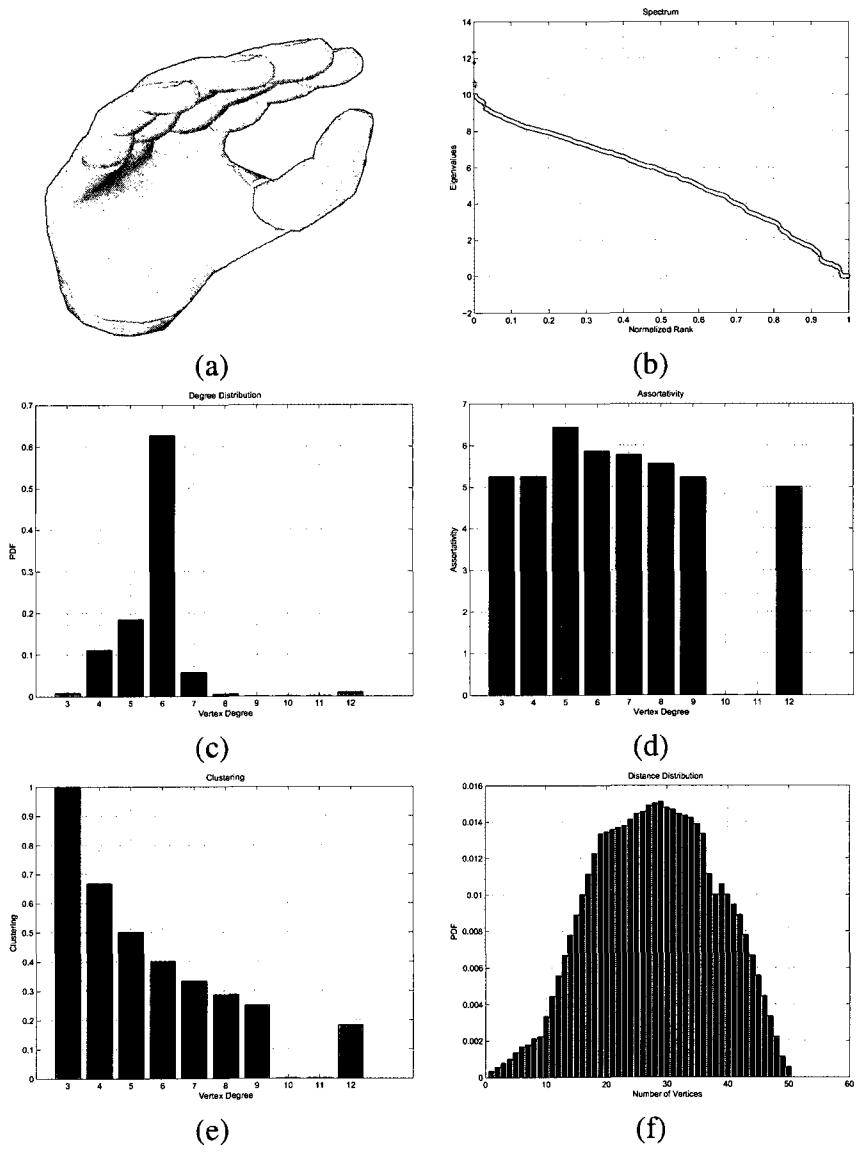


Figure 5.11: (a) 3D hand model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

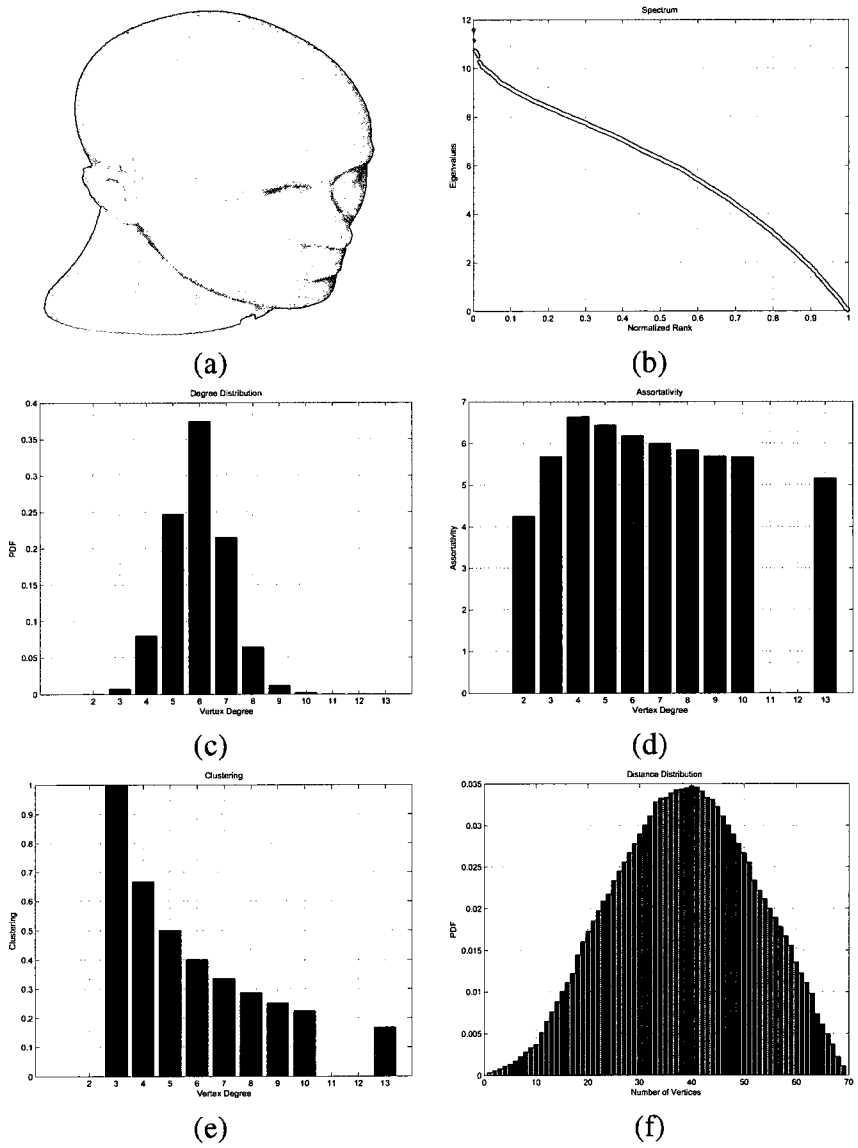


Figure 5.12: (a) 3D mannequin model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

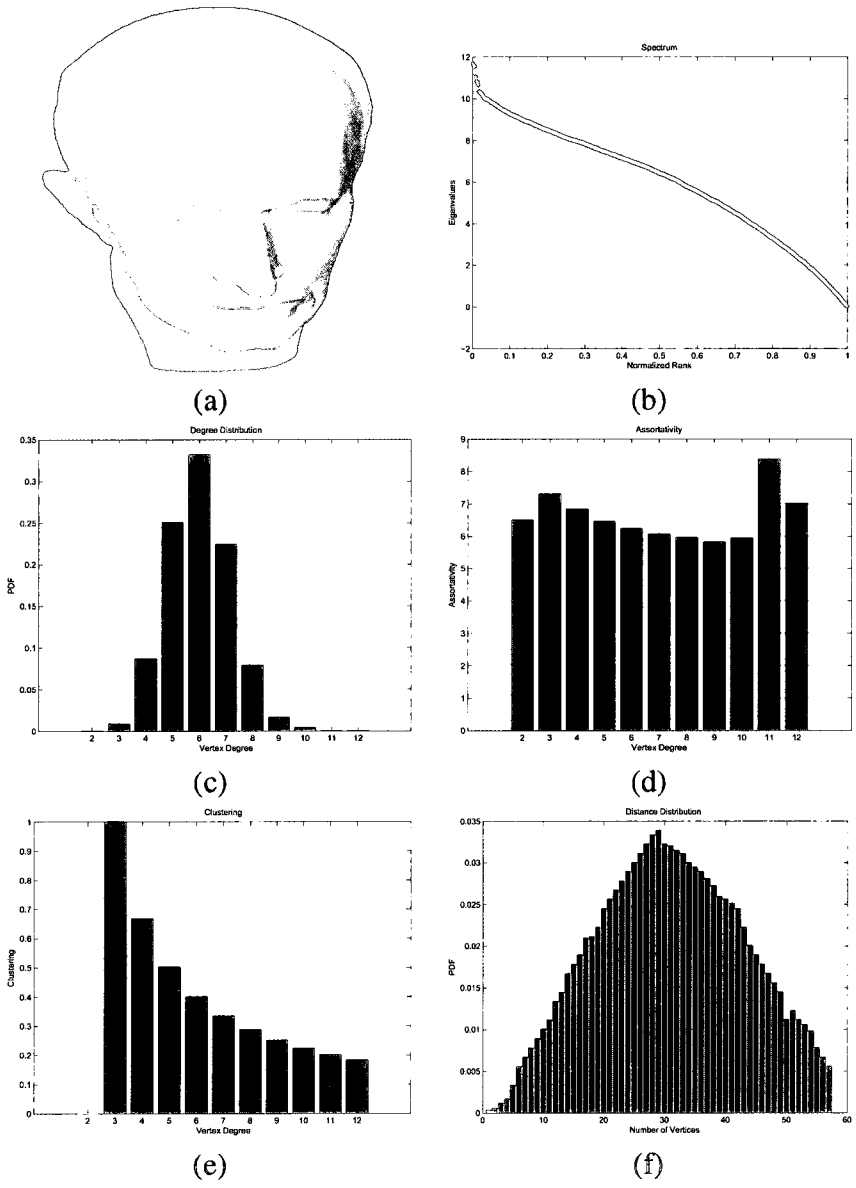


Figure 5.13: (a) 3D maxplanck model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

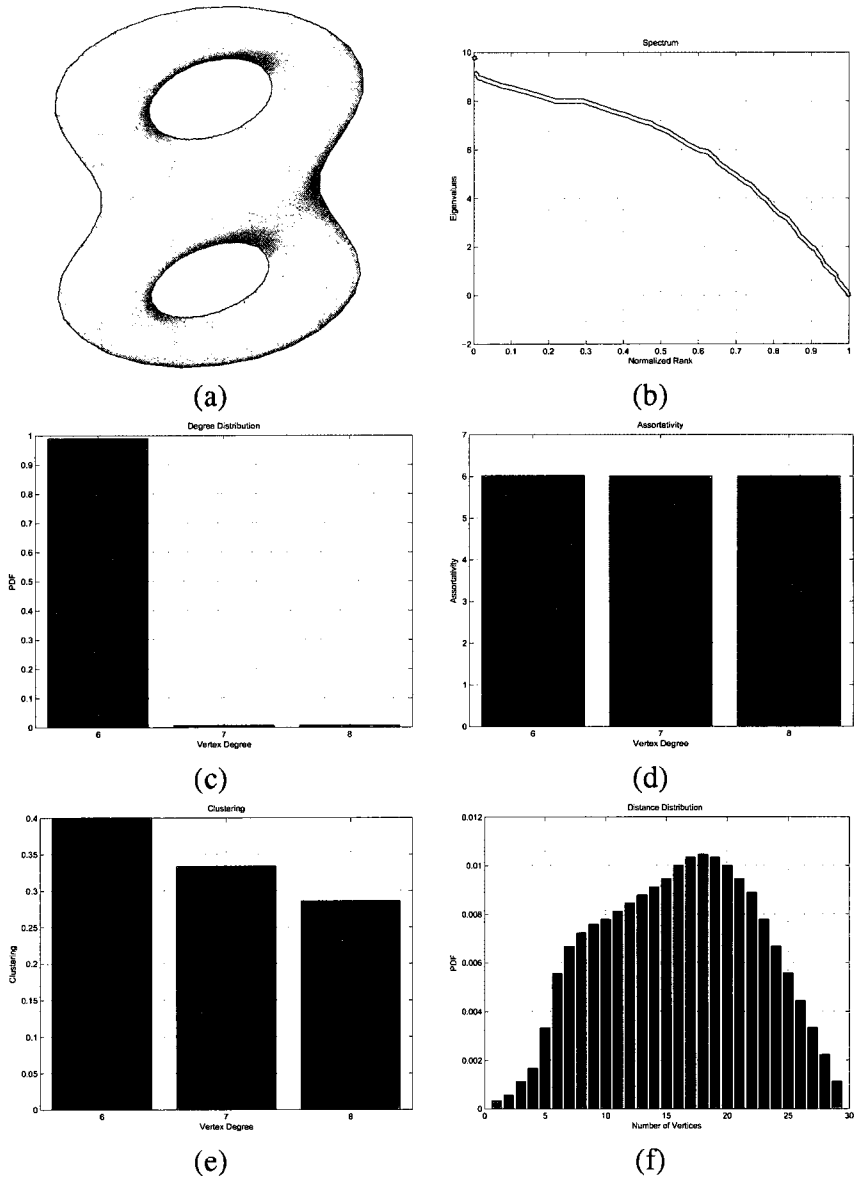


Figure 5.14: (a) 3D eight model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

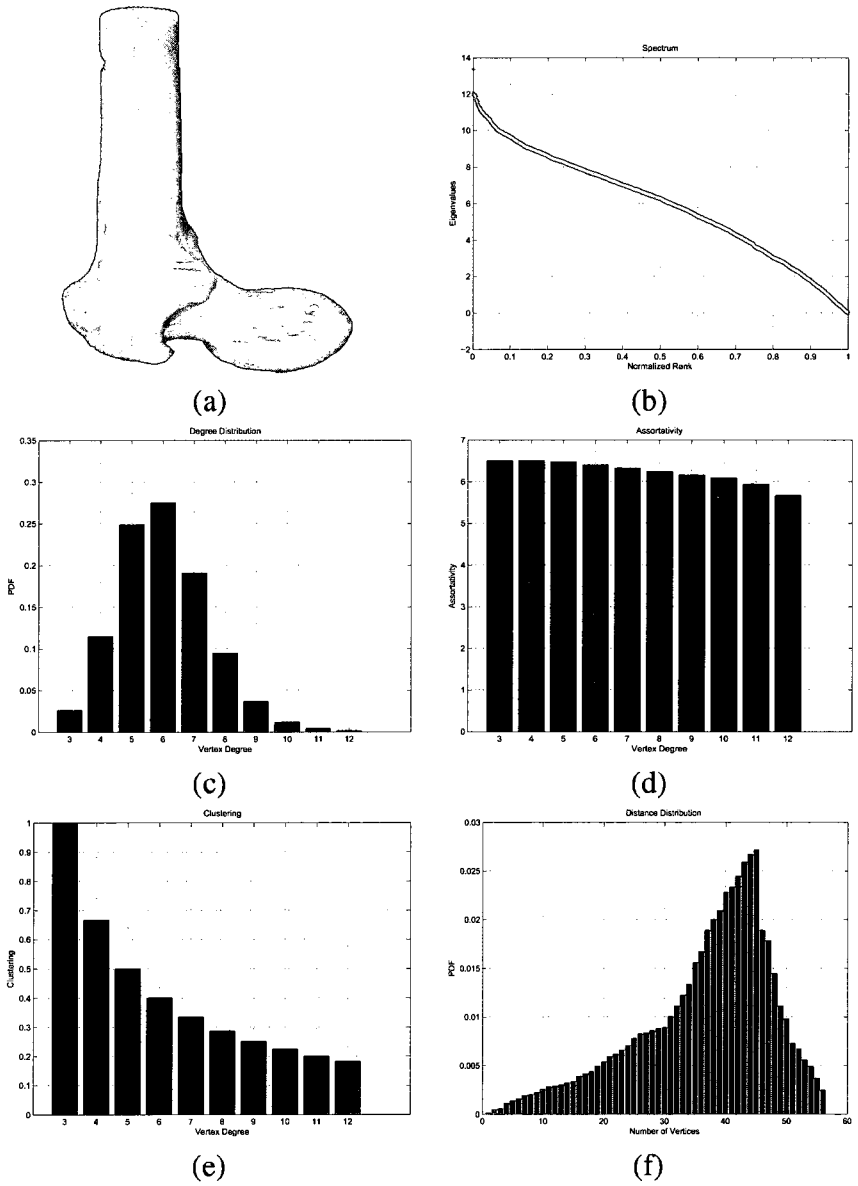


Figure 5.15: (a) 3D femur model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

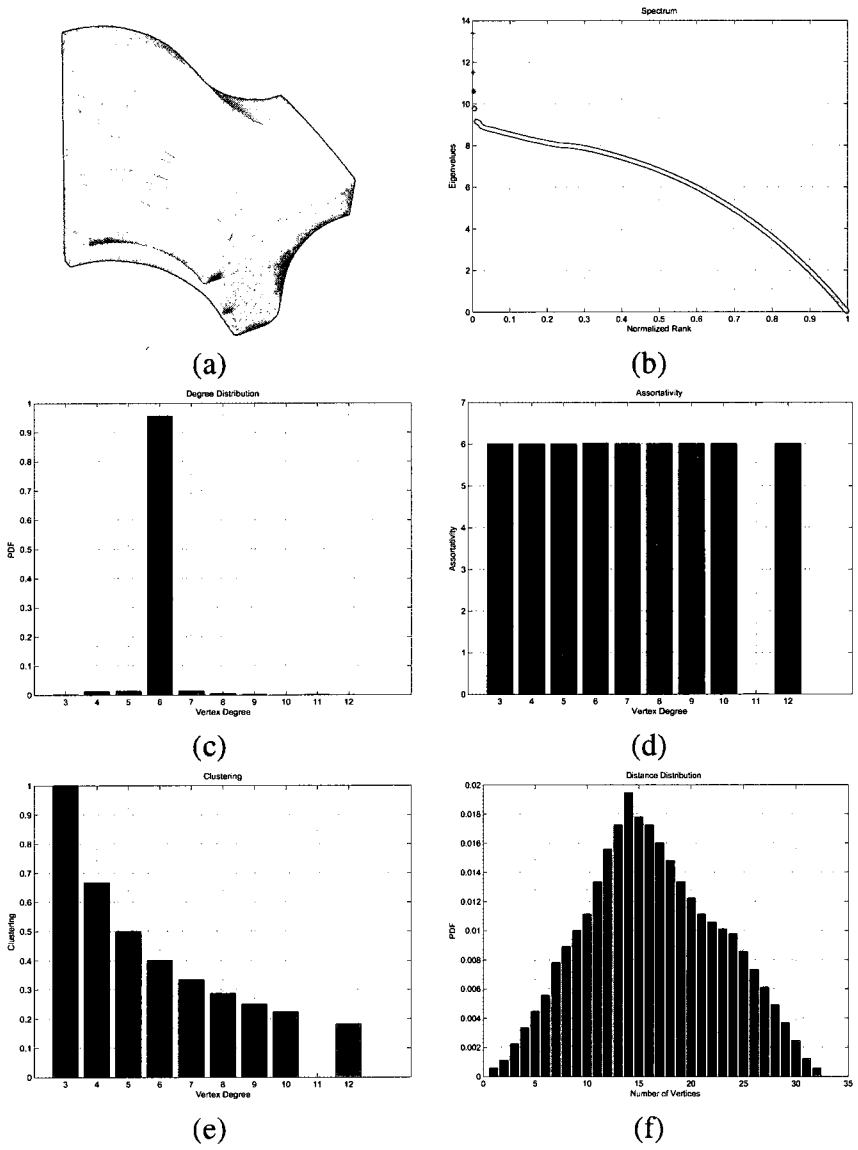


Figure 5.16: (a) 3D findisk model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

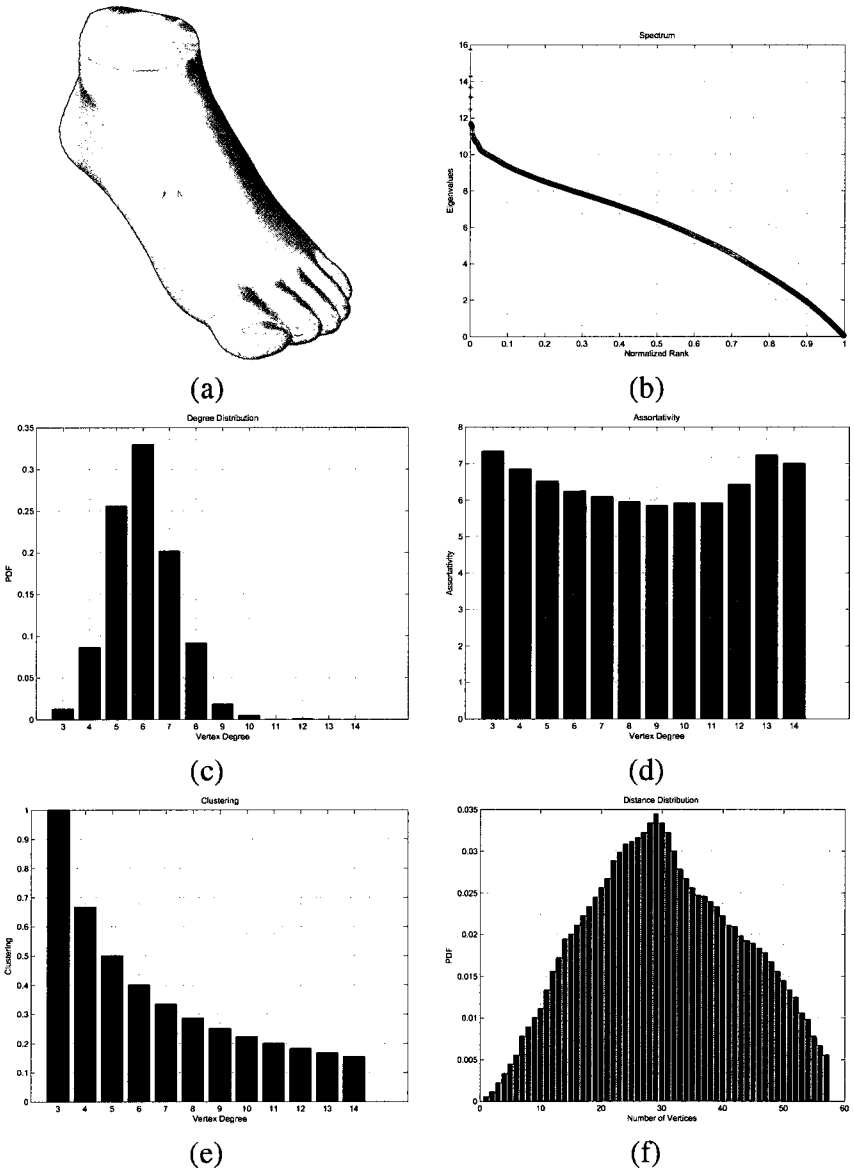


Figure 5.17: (a) 3D foot model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

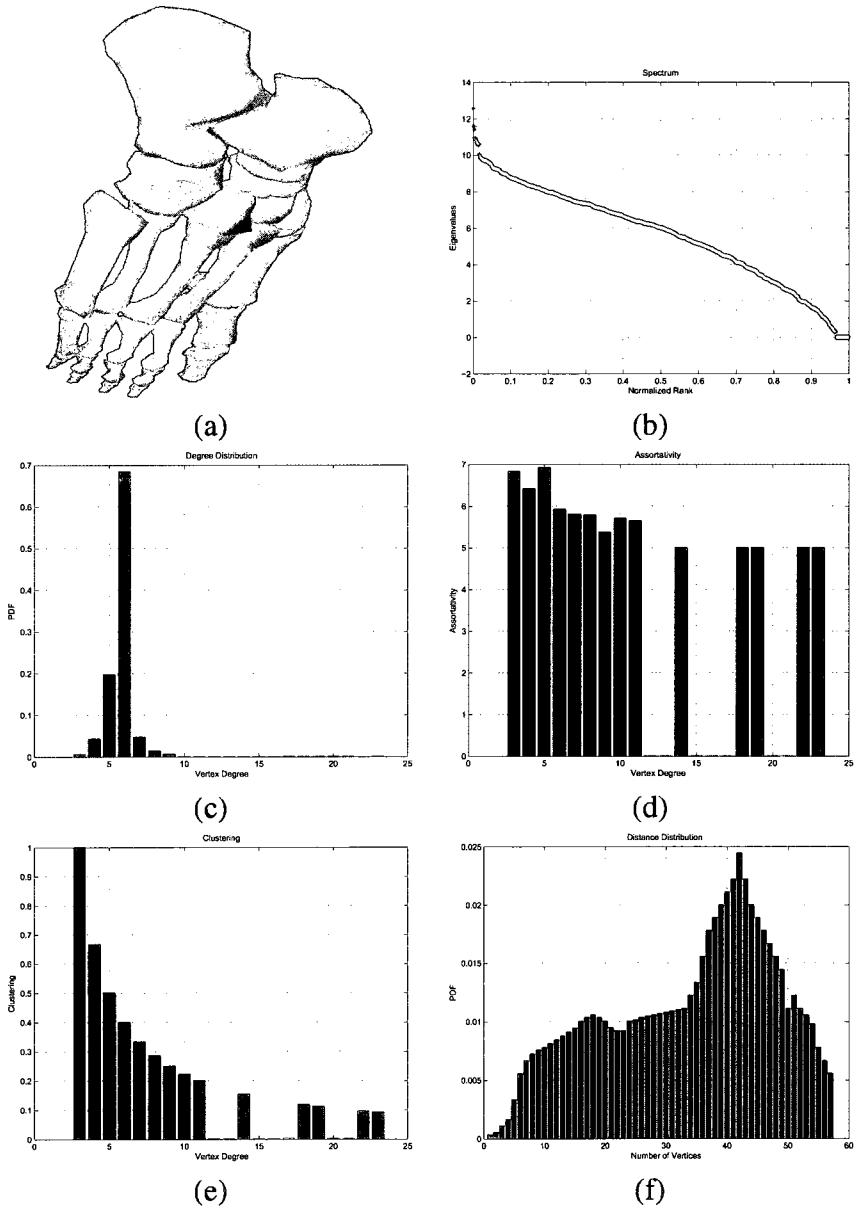


Figure 5.18: (a) 3D footbones model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

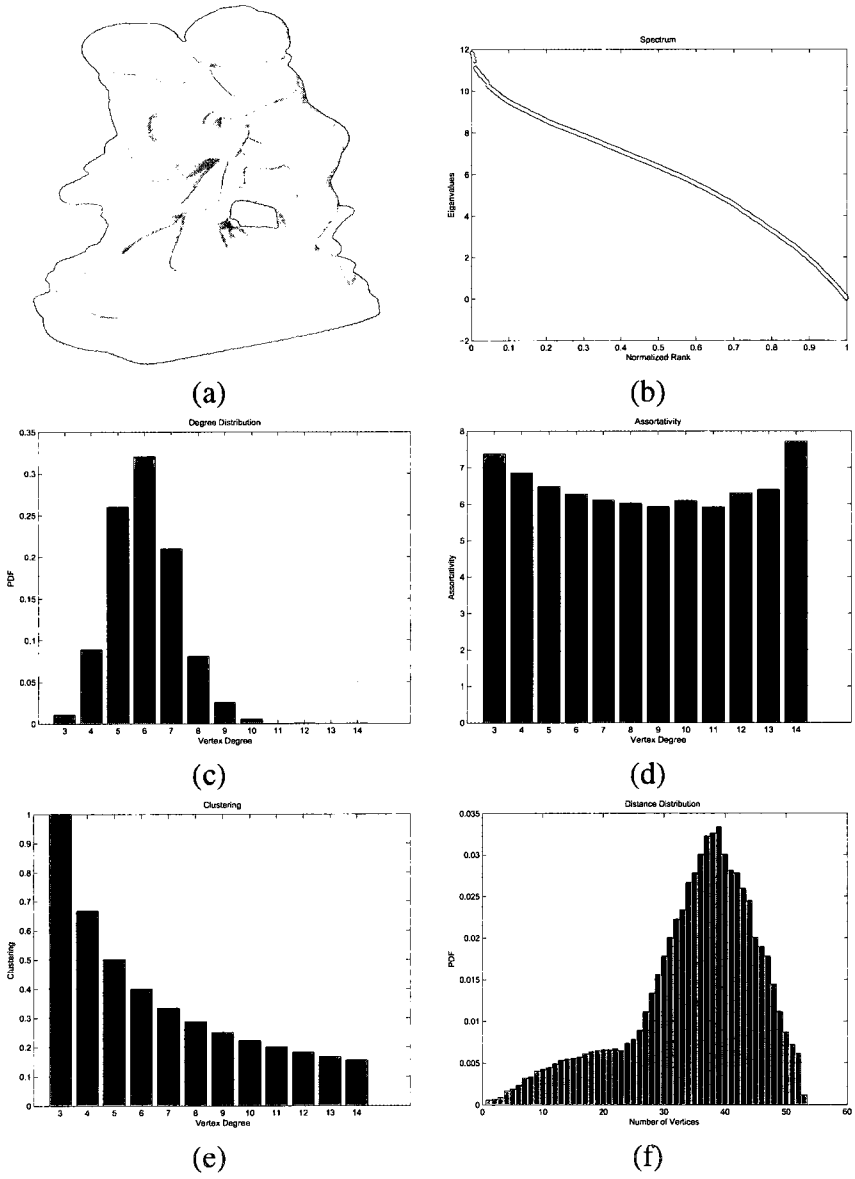


Figure 5.19: (a) 3D sculpture model, (b) spectrum, (c) vertex degree distribution, (d) assortativity distribution, (e) clustering distribution, (f) geodesic distance distribution.

CHAPTER 6

Conclusions and Future Research Directions

This chapter briefly concludes the thesis and highlights the major contributions of this research. This thesis presented a robust 3D mesh denoising method based on kernel density estimation, a 3D mesh compression approach based on the mesh umbrella matrix, a 3D mesh fingerprinting scheme, and statistical 3D mesh distributions. We have demonstrated the effectiveness of the proposed methods through numerical experiments with a variety of 3D models.

In the next section, the contributions made in each of the previous chapters and the concluding results drawn from the associated research work are presented. Suggestions for future research directions related to this thesis are provided in Section 6.2.

6.1 Thesis Contributions

6.1.1 Mesh denoising via multivariate kernel diffusion

In chapter 2, we introduced a simple and fast 3D mesh denoising technique using the concept of multivariate kernel density estimation. The main idea behind our proposed approach is to use a regularized bandwidth matrix of the kernel density in order to avoid over-smoothing and to fully preserve the geometric structure of the 3D mesh data, while effectively removing undesirable noise. The experimental results showed that our proposed technique is robust, accurate, and has a low

computational cost compared to existing methods.

6.1.2 3D mesh compression

In chapter 3, we proposed an new methodology for 3D object compression. The key idea is to apply spectral decomposition to the mesh umbrella matrix. Object compression is then achieved by discarding the largest eigenvalues/eigenvectors. The main advantages of the proposed approach are: (i) the mesh umbrella matrix captures very well the connectivity information of the 3D mesh data, (ii) the approach is simple and computationally inexpensive, and (iii) the simulation results indicate the suitability of the proposed technique for 3D object compression.

6.1.3 3D mesh fingerprinting

In chapter 4, we proposed a robust hashing scheme for 3D models. The approach consists of partitioning a 3D model into sub-meshes, followed by applying eigen-decomposition to the Laplace-Beltrami matrix of each sub-mesh in order to obtain the hash values of all the sub-meshes. The performance of the proposed method was evaluated through extensive experiments which clearly showed satisfactory resiliency against a variety of attacks.

6.1.4 Statistical 3D mesh distributions

In chapter 5, we proposed several statistical measures to analyze the topological properties of 3D models. The proposed statistical measures include the mesh degree, the mesh assortativity, the mesh clustering coefficient, and the mesh geodesic distance distributions. The experimental results clearly showed the effectiveness of the proposed measures in quantifying the topological features of 3D objects.

6.2 Future Research Directions

Several interesting research directions motivated by this thesis are discussed next. In addition to designing robust mesh denoising approaches for 3D object processing, we intend to accomplish the following projects in the near future:

6.2.1 Kernel mesh denoising

Our ongoing efforts are focused on exploring the use of the local structure tensor instead of the covariance matrix. Also, theoretically we hope to develop more rigorous way of finding the optimal regularization parameter of the covariance matrix.

6.2.2 Kernel topological modeling

Another future work direction is to tailor the proposed mesh neighborhood weighting kernel matrix $\mathcal{K}$, which takes into account the connectivity and geometry of an object, to compute skeletons of robust topology for 3D objects by constructing kernel Reeb graphs from the eigenfunctions of $\mathcal{K}$.

6.2.3 Object matching and retrieval

Another possible future work direction is the use of statistical shape distributions as a shape signature for 3D object matching and retrieval. Also, it would be of interest to incorporate topology into the proposed methodology through surface singularities of the global geodesic shape function.

List of References

[1] K. Tarmissi and A. Ben Hamza, "Multivariate kernel diffusion for surface denoising," *Springer Journal of Signal, Image and Video Processing*, in press, available online 2010.

[2] K. Tarmissi and A. Ben Hamza, "Information-theoretic hashing of 3D objects using spectral graph theory," *Elsevier Journal of Expert Systems with Applications*, vol. 36, no. 5, pp. 9409-9414, 2009.

[3] K. Tarmissi and A. Ben Hamza, "Geometric mesh denoising via multivariate kernel diffusion," *Proc. MIRAGE'09 – Lecture Notes in Computer Science*, vol. 5496, pp. 23-33, 2009.

[4] K. Tarmissi and A. Ben Hamza, "Feature-preserving kernel diffusion for surface denoising," *Proc. IEEE Int. Conf. Image Processing*, pp. 2973-2976, 2009.

[5] M. Ouhsain, K. Tarmissi, and A. Ben Hamza, "Spectral compression of 3D triangle meshes" *Proc. Int. Science and Technology Conference*, Malaga, Spain, 2007.

[6] M. Qasaimeh, Y. Zhang, K. Tarmissi, and A. Ben Hamza, "Statistical mesh distributions for 3D object topology," *Proc. IEEE Int. Symp. Signal Processing & Its Applications*, UAE, pp. 1-4, 2007.

[7] G. Taubin, "A signal processing approach to fair surface design," *Proc. ACM SIGGRAPH*, pp. 351-358, 1995.

[8] Y. Ohtake, A.G. Belyaev, and I.A. Bogaevski, "Polyhedral surface smoothing with simultaneous mesh regularization," *Proc. Geometric Modeling and Processing*, pp. 229-237, 2000.

[9] G. Taubin, "Linear anisotropic mesh filtering," IBM Research Report, RC2213, 2001.

[10] S. Petitjean "A survey of methods for recovering quadrics in triangle meshes," *ACM Computing Surveys*, vol. 34, no. 2, pp. 211-262, 2002.

[11] Y. Shen and K.E. Barner, "Fuzzy vector median-based surface smoothing," *IEEE Trans. Visualization and Computer Graphics*, vol. 10, no. 3, pp. 252-265, 2004.

[12] H. Yagou, Y. Ohtake, and A. Belyaev, "Mesh smoothing via mean and median filtering applied to face normals," *Proc. Geometric Modeling and Processing Theory and Applications*, pp. 124-131, 2002.

[13] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral mesh denoising," *Proc. ACM SIGGRAPH*, pp. 950-953, 2003.

[14] T. Jones, F. Durand, and M. Desbrun, "Non-iterative, feature preserving mesh smoothing," *Proc. ACM SIGGRAPH*, pp. 943-949, 2003.

[15] M. Desbrun, M. Meyer, P. Schröder, and A. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow," *Proc. ACM SIGGRAPH*, pp. 317-324, 1999.

[16] M. Desbrun, M. Meyer, P. Schröder, and A. Bar, "Anisotropic feature-preserving denoising of height fields and bivariate data," *Graphics Interface*, pp. 145-152, 2000.

[17] U. Clarenz, U. Diewald, and M. Rumpf, "Anisotropic geometric diffusion in surface processing," *Proc. IEEE Visualization*, pp. 397-405, 2000.

[18] U. Clarenz, U. Diewald, and M. Rumpf, "Processing textured surfaces via anisotropic geometric diffusion," *IEEE Trans. Image Processing*, vol. 13, no.2, pp. 248-261, 2004.

[19] C.L. Bajaj, and G. Xu, "Anisotropic diffusion of subdivision surfaces and functions on surfaces," *ACM Trans. Graphics*, vol. 22, no.1, pp. 4-32, 2003.

[20] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher, "Geometric surface smoothing via anisotropic diffusion of normals," *Proc. IEEE Visualization*, pp. 125-132, 2002.

[21] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher, "Geometric surface processing via normal maps," *ACM Trans. Graphics*, vol. 22, no. 4, pp. 1012-1033, 2003.

[22] K. Hildebrandt and K. Polthier, "Anisotropic filtering of non-linear surface features," *Computer Graphics Forum*, vol. 23, no. 3, pp. 391-400, 2004.

[23] J. Peng, V. Strela, and D. Zorin, "A simple algorithm for surface denoising," *Proc. IEEE Visualization*, pp. 107-112, 2001.

[24] V. Delouille, M. Jansen, and R. von Sachs, "Second-generation wavelet denoising methods for irregularly spaced data in two dimensions," *Signal Processing*, vol. 86, no. 7, pp. 1435-1450, 2006.

[25] X. Le Faucheur, B. Vidakovic, and A. Tannenbaum,"Bayesian spherical wavelet shrinkage: applications to shape analysis," *Proc. SPIE*, vol. 6763, 2007.

[26] A.F. El Ouafdi, D. Ziou, and H. Krim, "A smart stochastic approach for manifold smoothing," *Computer Graphics Forum*, vol. 27, no. 5, pp. 1357-1364, 2008.

[27] Y. Zhang and A. Ben Hamza, "Vertex-based diffusion for 3D mesh denoising," *IEEE Trans. on Image Processing*, vol. 16, no. 4, pp. 1036-1045, 2007.

[28] B.W. Silverman, *Density estimation for statistics and data analysis*, Chapman and Hall, London, 1986.

[29] Z. Karni and C. Gotsman, "Spectral compression of mesh geometry," *Proc. ACM SIGGRAPH*, pp. 279-286, 2000.

[30] Y. Shinagawa, T.L. Kunii, and Y.L. Kergosien, "Surface coding based on Morse theory," *IEEE Comp. Graph. and Appl.*, 11(5), 1991, 66-78.

[31] S. Yamany and A. Farag, "Surface signatures: an orientation independent free-form surface representation scheme for the purpose of objects registration and matching," *IEEE Trans. Pattern Analysis and Machine intelligence*, vol. 24, no. 8, pp. 1105-1120, 2002.

[32] A. Ben Hamza and H. Krim, "Geodesic matching of triangulated surfaces," *IEEE Trans. Image Processing*, vol. 15, no. 8, pp. 2249- 2258, 2006

[33] W.J. Rey, *Introduction to robust and quasi-robust statistical methods*, Springer, Berlin, Heidelberg, 1983.

[34] A.J. Menezes , P.C van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[35] R. Venkatesan, S. M. Koon, M.H. Jakubowski, and P. Moulin, "Robust image hashing," *Proc. IEEE Int. Conf. Image Processing*, Vancouver, Canada, 2000.

[36] V. Monga and B.L. Evans, "Perceptual image hashing via feature points: performance evaluation and tradeoffs," *IEEE Trans. Image Processing*, vol. 15, no. 11, pp. 3452-3465, 2006.

[37] A. Meixner and A. Uhl, "Analysis of a wavelet based robust hash algorithm," *Proc. SPIE Security, Steganography Watermarking of Multimedia Contents*, San Jose, CA, 2004.

[38] A. Swaminathan, Y. Mao, and M. Wu, "Robust and secure image hashing," *IEEE Trans. on Information Forensics and Security*, vol. 1, no. 2, pp. 215-230, 2006.

[39] B. Lamiroy and P. Gros, "Rapid object indexing and recognition using enhanced geometric hashing," *Proc. ECCV*, pp. 59-70, 1996.

[40] H. Wu and Y. Cheung, "Public authentication of 3D mesh models," *Proc.IEEE/WIC/ACM Int. Conf. Web Intelligence*, pp. 940-946, Hong Kong, 2006.

[41] B.Y. Wu and K.M. Chao, *Spanning Trees and Optimization Problems*, Chapman & Hall/CRC, 2004.

[42] C. Tsallis, "Possible generalization of Boltzmann-Gibbs statistics," *Jour. Statistical Physics*, vol. 52, pp. 479-487, 1988.

[43] A.O Hero, B. Ma, O. Michel, and J. Gorman, "Applications of entropic spanning graphs," *IEEE Signal Processing Magazine*, vol. 19, no. 5, pp. 85-95, 2002.

[44] G. Karypis and V. Kumar, "MeTiS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices," *Version 4.0, University of Minnesota*, 1998.

[45] G.L. Miller, S.H. Teng, W. Thurston, and S.A. Vavasis. "Automatic mesh partitioning," *Institute for Mathematics and Its Applications*, vol. 56, 1993.

[46] G.L. Miller, S.H. Teng, W. Thurston, and S.A. Vavasis, "Geometric separators for finite-element meshes," *SIAM Jour. on Scientific Computing*, vol. 19, no. 2, pp. 364-386, 1998.

[47] C. Bettstetter, "On the minimum node degree and connectivity of a wireless multihop network," *ACM Press*, pp. 80-91, 2002.

[48] A.T. Fomenco and T.L. Kunii, *Topological modeling for visualization*, Springer, 1997.

[49] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, pp. 167-256, 2003.

[50] A. Barrat, M. Barthelemy, and A. Vespignani, "Modeling the evolution of weighted networks," *Physical Review E*, vol. 70, pp. 066149-1 - 066149-12, 2004.

[51] P. Mahadevan *el al.*, "The internet AS-level topology: three data sources and one definitive metric," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 17-26, 2006.

[52] H. Fuks and C. Phipps, "Toward a model of language acquisition threshold," *Proc. Conference on Modelling and Simulation*, Montreal, Canada, 2006.

[53] J.B. Tenenbaum, V. de Silva, and J.C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319-2323, 2000.

[54] P. Perona and J. Malik, "Scale space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629-639, 1990.

[55] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, pp. 259-268, 1992.

[56] Y.L. You, W. Xu, A. Tannenbaum, M. Kaveh, "Behavioral Analysis of anisotropic diffusion in image processing," *IEEE Trans. Image Processing*, vol. 5, no. 11, pp. 1539-1553, 1996.

[57] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud, "Deterministic edge-preserving regularization in computed imaging," *IEEE Trans. Image Processing*, vol. 6, pp. 298-311, 1997.

[58] J. Weickert, *Anisotropic diffusion in image processing*, Teubner-Verlab, 1998.

[59] A. Yezzi, "Modified curvature motion for image smoothing and enhancement," *IEEE Trans. Image Processing*, vol. 7, no. 3, pp. 345-352, 1998.

[60] P. Kornprobst, R. Deriche, and G. Aubert, "Image sequence analysis via partial differential equations," *Journal of Mathematical Imaging and Vision*, vol. 11, no. 1, pp. 5-26, 1999.

[61] C. Samson, L. Blanc-Féraud, G. Aubert, and J. Zerubia, "A variational model for image classification and restoration," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, pp. 460-472, 2000.

[62] M. Cetin and W.C. Karl, "Feature-enhanced synthetic aperture radar image formation based on nonquadratic regularization," *IEEE Trans. Image Processing*, vol 10, pp. 623-631, 2001.

[63] Y. Bao and H. Krim, "Smart nonlinear diffusion: a probabilistic approach," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 63-72, 2004

[64] M. Giaquinta and S. Hildebrandt, *Calculus of Variations I: The Lagrangian Formalism*, Springer-Verlag, 1996.

[65] X.T. Li, T.W. Woon, T.S. Tan, Z.Y. Huang, "Decomposing polygon meshes for interactive applications," *Proc. ACM Symposium on Interactive 3D Graphics*, pp. 35-42, 2001.

[66] G.M. Nielson and T.A. Foley, "A survey of applications of an affine invariant norm," *Mathematical Methods in Computer Aided Geometric Design*, Academic Press, Boston, pp. 445-467, 1989.

[67] T.F. Cox and M.A. Cox, *Multidimensional scaling*, second edition, Monographs on Statistics and Applied Probability, vol. 88, 2001.

[68] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, "Stability of persistence diagrams," *Proc. Symp. Computational Geometry* , pp. 263-271, 2005.

[69] V. Guillemin and A. Pollack, *Differential topology*, Prentice-Hall, 1974.

[70] M. do Carmo, *Differential geometry of curves and surfaces*, Prentice-Hall, 1976.

[71] F.R. Chung, *Spectral Graph Theory*, American Mathematical Society, 1997.

[72] M. Ankerst, G. Kastenmüller, H. Kriegel, and T. Seidl, "3D shape histograms for similarity search and classification in spatial databases," *Proc. Int. Sympo. Advances in Spatial Databases*, pp. 207-226, 1999.