Carnegie Mellon University

CARNEGIE INSTITUTE OF TECHNOLOGY

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF **Doctor of Philosophy**

TITLE: Geometric Modeling and Shape Analysis for Biomolecular Complexes Based on Eigenfunctions

PRESENTED BY: Tao Liao

ACCEPTED BY THE DEPARTMENT OF : Mechanical Engineering

APPROVED BY THE COLLEGE COUNCIL :

Geometric Modeling and Shape Analysis for Biomolecular Complexes Based on Eigenfunctions

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Department of Mechanical Engineering

Tao Liao

B.S., Automotive Engineering, Tsinghua University

M.S., Automotive Engineering, Tsinghua University

Carnegie Mellon University

Pittsburgh, PA

August 2015

Abstract

Geometric modeling of biomolecules plays an important role in the study of biochemical processes. Many simulation methods depend heavily on the geometric models of biomolecules. Among various studies, shape analysis is one of the most important topics, which reveals the functionalities of biomolecules. To enable the geometric modeling and shape analysis for various biomolecular complexes, we develop: (a) an efficient multi-scale modeling method for the biomolecular complexes accelerated using the CPU- and GPU-based parallel computation; (b) a structure-aligned surface parameterization method; (c) an adaptive and anisotropic T-mesh generation algorithm; (d) a shape correspondence analysis method for biomolecules based on volumetric eigenfunctions; and (e) a shape analysis approach based on geometric operators from the second fundamental form of the surface.

Various algorithms for the analysis of biomoleucles have been developed based on their surface or volumetric meshes. We introduce an efficient computational framework to construct multi-scale models, which reflect the geometries of the biomolecular complexes represented by atomic resolution data in the Protein Data Bank (PDB). A multi-level summation of Gaussian kernel functions is employed to generate implicit models for biomolecules. The coefficients in the summation are designed as functions of the structure indices, which specify the structures at a certain level and enable a local resolution control on the biomolecular surface. To improve the efficiency of Gaussian density map construction, an error-bounded atom elimination method is introduced to reduce the atom number. Moreover, a method called neighboring search is adopted to locate the grid points close to the expected biomolecular surface, and reduce the number of grids to be analyzed. For a specific grid point, a KD-tree or bounding volume hierarchy is applied to search for the atoms contributing to its density computation, and faraway atoms are ignored due to the decay of Gaussian kernel functions. In addition to density map construction, three modes are also employed and compared during mesh generation and quality improvement: CPU

sequential, multi-core CPU parallel and GPU parallel. With all these techniques, quality triangle and tetrahedral meshes can be quickly generated for large biomolecules.

Compared with the commonly used triangle meshes, quadrilateral meshes are known to be more accurate in the Finite Element Analysis (FEA). Besides, the quadrilateral meshes can also be used as the control meshes of spline or subdivision surfaces, enabling more mathematical tools for the analysis of biomolecules. An important method to build the quadrilateral meshes is the surface parameterization. In this thesis, we present a structure-aligned approach for surface parameterization using eigenfunctions from the Laplace-Beltrami operator. Several methods are designed to combine multiple eigenfunctions using isocontours or characteristic values of the eigenfunctions. The combined gradient information of eigenfunctions is then used as a guidance for the cross field construction. Finally, a global parameterization is computed on the surface, with an anisotropy enabled by adapting the cross field to non-uniform parametric line spacings. By combining the gradient information from different eigenfunctions, the generated parametric lines are automatically aligned with the structural features at various scales, which are insensitive to local detailed features on the surface when low-mode eignfunctions are used.

Another issue for geometric modeling is to build a T-spline representation for the biomolecular surface, which enables the study of biomolecules using isogeometric analysis. In this thesis, an algorithm is developed to build the adaptive and anisotropic control mesh of the T-spline surface, namely T-mesh, for the biomolecular surface. The adaptation is achieved by adapting the parametric line spacings to different surface resolutions in the cross field-based parameterization. Moreover, an anisotropy defined from an input scalar field can also be achieved. From the parameterization results, we extract adaptive and anisotropic T-meshes for the further T-spline surface construction. Finally, a gradient flow-based method is developed to improve the T-mesh quality, with the anisotropy preserved in the quadrilateral elements. The effectiveness of the presented algorithm has been verified using several large biomolecular complexes.

Based on the geometric models, various shape analysis can be performed for biomolecules. Tracking the deformation and comparing biomolecular shapes are essential in understanding their mechanisms. In this thesis, a new spectral shape correspondence analysis method is introduced for biomolecules based on volumetric eigenfunctions. The eigenfunctions are computed from the joint graph of two given shapes, avoiding the sign flipping and confusion in the order of modes. An initial correspondence is built based on the distribution of a shape diameter, which matches similar surface features in different shapes and guides the eigenfunction computation. A two-step scheme is developed to determine the final correspondence. The first step utilizes volumetric eigenfunctions to correct the assignment of boundary nodes that disobey the main structures. The second step minimizes the distortion induced by deforming one shape to the other. As a result, a dense point correspondence is constructed between the two given shapes, based on which we approximate and predict the shape deformation, as well as quantitatively measure the detailed shape differences.

Geometric operators are powerful mathematical tools for shape analysis of biomolecules. In this thesis, we introduce two new geometric operators based on the second fundamental form of the surface, namely the secondary Laplace operator (SLO) and generalized Giaquinta-Hildebrandt operator (GGHO). Surface features such as concave creases/regions and convex ridges can be captured by eigenfunctions of the SLO, which can be used in surface segmentation with concave and convex features detected, such as segmenting protein pockets. Moreover, a new geometric flow method is developed based on the GGHO, providing an effective tool for sharp feature-preserving surface smoothing.

In summary, several new geometric modeling methods are introduced in this thesis, which build different types of meshes to represent the biomolecular surface, including the triangle, tetrahedral, quadrilateral meshes and the T-mesh. Based on the geometric models, new methods for the shape analysis of biomolecules are developed based on surface or

volumetric eigenfunctions from different geometric operators, such as the shape correspondence analysis and pocket segmentation.

Acknowledgements

First of all, I would like to give the most sincere gratitude to my advisor Dr. Yongjie Jessica Zhang. I thank her for all the patient instructions and inspiring guidance throughout my doctorial study. Not only the academic skills but also the enterprising spirit learned from her will benefit me for the rest of my life.

I would like to thank my committee, Dr. Ge Yang, Dr. Kenji Shimada and Dr. Levent Burak Kara for their great support and insightful comments and I have learned a lot from them. Besides, I thank all our collaborators, Dr. Ge Yang, Dr. Guoliang Xu, Dr. Pete Kekenes-Huskey, Xinge Li and Hao-Chih Lee. It was great experience to work with them.

I want to thank all my labmates, Dr. Wenyan Wang, Dr. Xinghua Liang, Dr. Jin Qian, Lei Liu, Kangkang Hu, Xiaodong Wei, Yicong Lai, Aishwayar Pawar, and Arjun Kumar, in the Computational Biomodeling Lab. I feel lucky to study in this great research group, or family. Thank them all for the friendship and support for both my research and daily life.

I would like to thank my parents and my sister for their love and selfless support in all my life. I hope that this work makes them be proud of. Without them I could never finish my Ph.D. study. Moreover, I want to thank my friend Yiming Jing, who always cares about me like a brother. Thank Dr. Luoting Fu. The professional skills and spirit of innovation I learned from him are of great help for me in my doctorial study. I also want to thank my friends Xiaoxin Su and Xiaorui Wang. They gave me the warmest support and help for planning my future career.

This work was supported by Y. Zhang's NSF CAREER Award OCI-1149591, PECASE Award N00014-14-1-0234 and CIT Dean's Fellowship.

To my parents.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Biological functionalities of the biomolecules depend heavily on their geometries. Geometric modeling and shape analysis play an important role in various applications such as drug design and pathological study. Due to the complicated structures of biomolecular complexes, efficient and adaptive modeling algorithms are of great help for the analysis. On the other hand, eigenfunctions of various operators usually reveal specific geometric features, which have been utilized in different fields. It is worthy exploring the properties of eigenfunctions in the shape analysis of biomolecules. In this thesis, new algorithms for the geometric modeling of biomolecules are introduced, and novel shape analysis methods are developed based on eigenfunctions of different operators.

The biomolecular structures are usually represented in atomic resolution data, and implicit or explicit model of biomolecules needs to be built to represent their geometries. Different types of meshes have been utilized in various computations such as estimating electrostatic potentials and diffusion-based calcium signaling [18, 43, 116, 115, 141, 36, 136]. As progressively larger biomolecular complexes are studied, we need to handle huge amount of computation, which brings a great challenge for both modeling and simulation.

To efficiently represent complex biomolecules, a multiscale modeling method that controls the local resolution of the specified hierarchical structure is required, which can significantly reduce the mesh size and thus lighten the compuational cost for simulations. In addition, with the rapid development of parallel computers, multi-core CPU and GPU-based computation brings in new directions for the acceleration of modeling in various fields. In this thesis we also aim to apply these parallel computation techniques to multiscale modeling for biomolecules.

The biomolecular surfaces can be represented using various types of meshes. Compared with the commonly used triangle meshes, quadrilateral meshes are well known for higher computational accuracy. Besides, the quadrilateral meshes can be used for the control meshes of spline or subdivision surfaces, which enables more mathematical tools for the geometric modeling. One of the commonly used quadrilateral mesh generation method is the surface parameterization, which is of great importance for many applications, such as quadrilateral meshing [12], texture mapping and synthesis [128, 66]. An important issue for surface parameterization or quadrilateral generation is how to align parametric lines with the feature directions. Some simplification-based techniques [85, 95, 119] were developed to generate very coarse quadrilateral domain meshes with a good user control. Although feature alignment was achieved in a certain degree [85], it is difficult to control the simplification process to preserve surface features. Using the harmonic field [58, 122], features can be captured, but feature alignment is limited due to the difficulty in generating the field and placing singularities. In recent years, methods based on the cross field have been introduced [58, 13, 51, 96, 99]. Generally, the captured features in the cross field are represented by the principal curvatures, which are sensitive to the local detailed features and may fail in capturing structural features of an object at desired scales.

In the context of isogeometric analysis, the T-spline surface provides a powerful basis for computation in different applications [46, 8]. Due to the high efficiency of the T-spline control mesh, namely T-mesh, the multi-resolution feature on the biomolecular surface

can be represented efficiently. Basically, the T-meshes are quadrilateral meshes allowing T-junctions in the connectivity, which can also be built based on the surface parameterization. Because of the T-junctions, the adaptation can be achieved much easier compared with the general quadrilateral meshes, without sacrificing the alignment of parametric lines.

Since the interactions between biomolecules depend heavily on their surface shapes, shape analysis plays an important role for the study of biomolecules. Based on the surface and volumetric models generated using our geometric modeling methods, various operations can be performed for the shape analysis of biomolecules. Due to the flexibility of peptides, biomolecular shapes usually deform during the biochemical process, and the functionality of many biomolecules depends on their shape deformations. Therefore, measuring the shape differences and tracking the deformation of biomolecular shapes have become important issues in various applications. Although these operations can be performed based on the chemical structures of biomolecules, methods based purely on geometry still need to be explored, which enable the researchers to study the behavior of biomolecules simply based on their outline, and improve the efficiency of experimental study. To realize this, shape correspondence analysis provides a proper basis, which can reveal the shape differences or deformations explicitly based on the connections of corresponding nodes.

Behaviors of biomolecules are usually determined by special surface features such as pockets. Geometric operators are powerful mathematical tools to detect surface features. For example, eigenfunctions of the Laplace-Beltrami operator (LBO) reflect the structural feature of the object and are insensitive to the surface curvature. Compared with the first fundamental form of the surface which defines the LBO, the second fundamental form is more sensitive to the curvature-based surface features, and the applications of their eigenfunctions still need to be explored. In this thesis, the properties of these eigenfunctions are studied, and their applications such as detection and segmentation of protein pockets are explored. Besides, these operators also provide a new basis for the geometric flow operations for the purposes such as surface smoothing [72].

1.2 Problem Statement

In this thesis, we study five main problems:

- **CPU- and GPU-based parallel computation for biomolecular modeling.** Given a PDB/PQR file storing the atom centers and radii of a biomolecule, our method aims at building a quality tetrahedral mesh efficiently. There are three main steps for the modeling process: Gaussian density map computation; adaptive tetrahedral mesh extraction; and quality improvement. The speed of the modeling process depends heavily on the atom number in the biomolecule. An error-bounded atom elimination method is proposed to reduce the atom number by ignoring the low contributing atoms. Besides, a neighboring search algorithm is introduced together with the KD-tree technique, which significantly improves the computational efficiency of the Gaussian density map generation. The multi-core CPU- and GPU-based parallel computation is applied for all the three steps to accelerate the entire modeling process.

- **Structure-aligned surface parameterization.** One of the important issues for quadrilateral mesh generation based on the surface parameterization is the alignment of parametric lines. For many applications, an alignment to the main structure of the object is preferred, because capturing the detailed surface features usually results in more singularities in the mesh, which are undesired for the analysis. The gradients of eigenfunctions of Laplace-Beltrami operators reflect the structural feature of the object. However, an eigenfunction usually can only capture part of the structure. Therefore, an algorithm is proposed to combine the structural information from multiple eigenfunctions, and build a guidance for the parameterization that captures the intact structure. To improve the efficiency of resulting quadrilateral meshes, an improved parameterization method is proposed that adapts the parametric lines to the input field by adapting the cross field.

- **Adaptive and anisotropic T-mesh generation for multi-resolution biomolecular surfaces.** Taking the advantage of the surface parameterization, we develop a method for adaptive T-mesh construction for multi-resolution biomolecular surfaces. The biomolecular surface is divided into patches with different resolutions, and different parametric line spacings are set for each patch, with T-junctions on the patch boundaries. To make a strongly-balanced structure in the T-mesh, constraints are introduced for the parameterization at the patch boundaries. Moreover, an anisotropy can be introduced in the parameterization defined from an input guidance field. A quality improvement method is specified for the T-mesh, which can preserve the anisotropy in the T-mesh compared with other existing methods.

- **Shape correspondence computation based on volumetric eigenfunctions.** The existing shape analysis algorithms for biomolecules are usually based on the surface meshes, which is sensitive to the surface topology change. In this thesis, we introduce a novel point correspondence computation method based on the volumetric eigenfunctions. The volumetric model of the biomolecule consists of voxels in a rectilinear grid, and eigenfunctions are computed by solving an eigenproblem of the graph Laplacian of the volumetric model. The traditional spectral shape analysis methods were usually limited by the perturbations from large deformations, which destroys the invariance of eigenfunctions to shape deformations. Based on the joint graph, a novel method for the eigenfunction computation is introduced in this thesis, which increases the similarity of eigenfunctions in the shapes of two deformed biomolecules by setting additional constraints from the shape diameter. Besides, local neighborhoods are preserved by minimizing the distortion in the correspondence-based deformation. With the shape correspondence, various applications such as approximating, predicting and quantitative comparison of different shapes can be performed.

- **Geometric operator exploration based on the second fundamental form of surfaces.** The traditional Giaquinta-Hildebrandt operator (GHO) based on the second fundamental form is very sensitive to the flat regions on the surface, which makes it difficult to apply its eigenfunctions in shape analysis. Therefore, a new geometric operator, the secondary Laplacian operator (SLO) is introduced in this thesis. Different from the commonly used LBO derived from the first fundamental form, eigenfunctions from the SLO reveal the curvature information on the surface. Using different parameters, the eigenfuncitons of SLO emphasizes different surface features, which can be applied for surface segmentation for different purposes. Moreover, a generalized Giaquinta-Hildebrandt operator (GGHO) is introduced analogy to the GHO, and a new geometric flow method is proposed based on it, which can be used to strengthen or weaken geometric features during surface smoothing.

1.3 Contributions

There are two targets for the methods introduced in this thesis. Contributions have been made in each of them.

1. Constructing different types of quality models to represent the biomolecular surface.
    - An efficient multi-scale modeling method for biomolecular complexes is introduced with local surface resolution control, which is accelerated by CPU- and GPU-based parallel computation;
    - A new guidance for surface parameterization is developed based on eigenfunctions of the Laplace-Beltrami operator, which aligns parametric lines to the main structure of the object;
    - An anisotropic parameterization method is introduced, which reflects the anisotropy from an input field with different parametric line spacings; and

- A new parameterization method is developed for T-mesh generation from multi-resolution biomolecular surface, which adapts parametric line spacings to the local surface resolution.

2. Introducing new mathematical tools based on eigenfunctions of various geometric operators, which enable different operations for the shape analysis of biomolecules.

   - A new shape correspondence analysis method for biomolecules is developed based on the volumetric eigenfunctions from the joint graph of different biomolecular shapes, which enables the operations such as approximating and predicting the shape deformation and quantitative comparison of different shapes; and

   - Two new geometric operators, the SLO and GGHO, are introduced based on the second fundamental form of the surface. The SLO yields eigenfunctions sensitive to the curvature-related surface features, which can be applied for surface segmentation for various purposes. A new geometric flow method is developed based on the GGHO, which preserves and strengthens the surface features during smoothing.

1.4 Publication

The related primary publications include:

- T. Liao, H.-C. Lee, G. Yang, Y. J. Zhang. Shape correspondence analysis for biomolecules based on volumetric eigenfunctions, Molecular Based Mathematical Biology, submitted, 2015.

- T. Liao, X. Li, G. Xu, Y. Zhang. Secondary Laplace operator and generalized Giaquinta-Hildebrandt operator with applications on surface segmentation and

smoothing. A Special Issue of SIAM Conference on Geometric & Physical Modeling in Computer Aided Design, 2015. DOI: 10.1016/j.cad.2015.07.009

- T. Liao, G. Xu, Y. J. Zhang. Atom simplfication and quality T-mesh generation for multi-resolution biomolecular surfaces, Lecture notes in Computational Sciences and Engineering, Vol. 107, 2015.

- T. Liao, W. Wang, Y. Zhang. Adaptive and anisotropic T-mesh generation from cross field. Workshop on Structured Meshing: Theory, Application and Evaluation, the 27th Conference on Computer Animation and Social Agents (CASA 2014). Houston, TX. May 26-28, 2014.

- T. Liao, G. Xu, Y. Zhang, Structure-aligned guidance estimation in surface parameterization using eigenfunction-based cross field, Graphical Models 76 (2014) 691-705.

- T. Liao, Y. Zhang, P. Kekenes-Huskey, Y. Cheng, A. Michailova, A. McCulloch, M. Holst, J. McCammon, Multi-core CPU or GPU-accelerated multi-scale modeling for biomolecular complexes, Molecular Based Mathematical Biology 1 (2013) 164-179.

In addition, I participated some application projects of bimolecular modeling. The co-authored paper is listed below:

- H-C Lee, T. Liao, Y. J. Zhang, G. Yang. Shape Factor Analysis: Structure-Preserving Dimension Reduction on Biological Shape Spaces. Bioinformatics, 2015. Submitted

- P. Kekenes-Huskey, T. Liao, A. Gillette, J. Hake, Y. Zhang, A. Michailova, A. McCulloch, J. McCammon, Molecular and subcellular-scale modeling of nucleotide diffusion in the cardiac myofilament lattice, Biophysical Journal 105 (2013) 2130-2140.

1.5 Outline of Dissertation

Following the introduction, Chapter 2 gives a background literature review. Chapter 3 introduces an efficient biomolecular modeling method based on the Gaussian density map. Chapter 4 proposes a structure-aligned surface parameterization method based on eigenfunctions of the LBO. Chapter 5 develops a new parameterization method for the construction of T-mesh for multi-resolution biomolecular surfaces. Chapter 6 presents our new shape correspondence analysis method based on volumetric eigenfunctions. Chapter 7 introduces new geometric operators from the second fundamental form of the surface, and their applications for surface segmentation and smoothing. Finally, Chapter 8 draws the conclusion and points out the future work.

Chapter 2

Literature Review

2.1 Geometric Modeling for Biomolecular Complexes

There are three important biomolecular surfaces [21, 22]: the Van der Waals surface (VdW), the Solvent Accessible Surface (SAS), and the Solvent Excluded Surface (SES). For the VdW, atoms are represented as rigid spheres with Van der Waals radii, and the biomolecular surface is defined as the envelope of these spherical surfaces. The SAS and SES can be defined by assuming a probe rolling around the biomolecule and keeping contact with the atoms. Then the SAS is formed by tracing the trajectory of the probe center. The topological boundary of the union of all possible probes is called the SES, with no intersection with atoms. A lot of research has been conducted in approximating the SES, including the alpha-shapes [28, 1], the beta-shapes [56, 105], the MSMS [106], the advancing front and generalized Delaunay approaches [59], NURBS approximation [6], and PDE-based methods [133, 149, 44]. The biomolecules were also represented as implicit models. The Gaussian kernel functions were applied in constructing density maps for the biomolecules [10, 38, 62, 147, 140]. In [37], the atomic resolution Gaussian density map was built and then filtered using an ideal filter to obtain a smooth biomolecular surface.

Fast computing is critical in biomolecular modeling [124, 129]. A variety of algorithms were developed in improving the modeling efficiency. For example, the Fast Fourier Transform was used in [149, 44] to get better performance of the PDE transform, and a Non-uniform Fast Fourier Transform (NFFT) was adopted to improve the polynomial-form summation of the kernel functions [7]. The programmable GPU has brought in a new direction for vast data processing in geometric modeling [126, 68, 94, 54, 118].

2.2 Surface Parameterization and T-mesh Generation

In the context of isogeometric analysis [46, 8, 125, 49, 120], the T-spline surface provides a powerful basis for computation in different applications, which can also greatly benefit the analysis of biomolecules. Due to the high efficiency of the T-spline control mesh, namely T-mesh, the multi-resolution feature on the biomolecular surface can be represented efficiently. Various methods have been developed for T-mesh generation [130, 131, 89, 40, 69]. In recent years, the cross field-based global parameterization method provides new clues for T-mesh generation, as it has been more and more commonly used in surface quadrangulation [12, 89].

An important issue for surface parameterization is how to align parametric lines with the feature directions. Some simplification techniques [85, 95, 119] were developed to generate very coarse domain meshes with a good user control. Although feature alignment was achieved in a certain degree [85], it is difficult to control the simplification process to preserve surface features. Using the harmonic field [58, 122], features can be captured, but feature alignment is limited due to the difficulty in generating the field and placing singularities. Methods based on the cross field have been introduced [58, 13, 51, 96, 99]. The smooth cross field could be generated from a prescribed set of singularities [53], but it was more common to guide the field using principal curvatures, which are sensitive to the

local detailed features and may fail in capturing structural features of an object at desired scales.

2.3 Shape Comparison and Correspondence Analysis

In previous literature, various methods have been introduced to measure the difference between different shapes [71, 60, 70]. The shape of each object can be characterized by the statistical information of the surface features. Various methods have been developed to describe and compare different shapes. In [41], the outlines of different biomolecular shapes were described and compared based on the multi-resolution Reeb graph (MRG). A similar but improved method was developed in [70], which handled surfaces with arbitrary genus based on the shape skeleton. Spectral analysis is another important way to compare different shapes, which measures differences between shapes based on the spectrum of the Lapace-Beltrami operator (LBO) [103, 104, 32]. These methods target at scoring the similarities or dissimilarities between shapes for retrieval or classification, which usually cannot give details of the differences between shapes.

Different from the shape matching methods, shape correspondence analysis methods provide a means to explicitly show details about the similarity or difference between shapes [50]. Among different methods, the spectral graph theory has been widely used [123, 19]. Since the eigenfunctions of the graph Laplacian or Laplace-Beltrami operator are supposed to be invariant to the shape deformations, they can be used to match the corresponding points in deformed shapes. Characterized by the eigenfunctions, points in the flexible shapes were assigned to each other based on methods such as the Gaussian proximity matrix [108, 111, 16], which were applied for tracking the movement of articulated models [86]. However, the invariance of eigenfunctions may be affected by the perturbations such as expansion and compression, which may influence the accuracy of correspondence [101, 92]. Rigid and nonrigid transformations were adopted in [86, 79, 80, 47] to reduce the

influence of these perturbations. In [81], a joint graph was used to compute eigenfunctions based on an initial correspondence obtained according to important features on the cerebral cortex surfaces, which improved the similarity of eigenfunctions on different surfaces and avoided the problems such as flipping and improper ordering. In [79, 80], a fast computation method was introduced, which significantly reduced the computational time for the matching of large models. Many algorithms only care about the correspondence of some points at important locations, namely the landmarks [50]. However for the shape analysis of biomolecules, a dense point correspondence is preferred to perform the applications such as measuring the change of a certain area on the surface.

2.4 Eigenfunctions and Geometric Flow Method Based on Geometric Operators

Geometric operators are the basis for many algorithms in surface processing. Based on the first fundamental form of the surface, the Laplace-Beltrami operator (LBO) is defined and its eigenfunctions are well-known for their property of capturing shape behavior and structural feature of an object [65, 102, 104, 113]. They vary along the object surface and are invariant to different poses, which makes them ideal for applications in pose-invariant Reeb graph construction [113], shape matching [65] or registration [86], the Shape-DNA [104], and surface quadrangulation or parameterizaton [150, 74, 26, 45, 76]. Another important application of the LBO eigenfunctions is surface segmentation. The distribution of eigenfunctions is used together with isocontours [102, 127, 78] and point clustering [98, 142, 112] to segment surface into several components. In practice, different methods were employed to improve the performance of LBO eigenfunctions [42, 132]. To detect concavities, eigenfunctions from a concavity-aware Laplacian [127] were used to generate a single segmentation field.

Besides computing eigenfunctions, geometric operators are also used in various geometric flows for surface fairing and smoothing [23, 63, 64, 144, 139, 148]. The mean curvature and Gaussian curvatures are usually used to design geometric operators [139, 57]. Generally, geometric flows smooth surfaces and remove noise by moving nodes in the normal direction, while the tangential movement regularizes the elements. The tangential movement can also help strengthen surface features, which was seldom considered in the existing methods.

Chapter 3

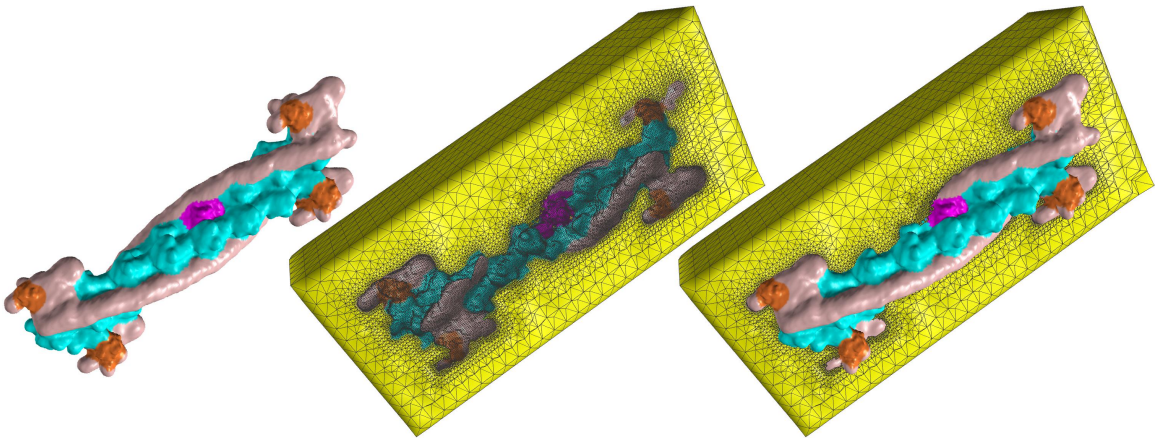
Multi-core CPU or GPU-accelerated Multiscale Modeling for Biomolecular Complexes

Biomolecules are usually represented by the atomic resolution data, such as PDB files. For many studies, geometric models are required to represent the biomolecular surface, which wraps all the atoms inside. In this chapter, we introduce an efficient approach for multi-scale modeling of biomolecular complexes, accelerated by CPU- and GPU-based parallel computation. Quality triangle and tetrahedral meshes are generated, which can be directly used for analysis.

3.1 Introduction

Numerical analysis has been essentially important for the study of biomolecular complexes in a wide range of applications such as estimating electrostatic potentials and diffusion-based calcium signaling [18, 43, 116, 115, 141, 36, 136]. As progressively larger biomolecular complexes (see Fig. 3.1) are studied, we need to handle huge amount of computation,

which brings a great challenge for both modeling and simulation. To efficiently represent complex biomolecules, a multiscale modeling method that controls the local resolution of the specified hierarchical structure is required, which can significantly reduce the mesh size and thus lighten the compuational cost for simulations. In addition, with the rapid development of parallel computers, multi-core CPU and GPU-based computation brings in new directions for the acceleration of modeling in various fields. In this chapter we aim to apply these parallel computation techniques to multiscale modeling for biomolecules. In the following, let us first briefly review previous work on biomolecular modeling and efficient computation.



(a) (b) (c)

Figure 3.1: Multi-scale model for protein 2W4U. High resolution is shown for the actin domain (cyan) with parameters $P_R = 0.8$, $P_C = 0.5$, and $P_D = 1.0$; chain K (magenta) and troponin C (orange) are further emphasized with $P_R = 2.0$, $P_C = 1.0$, and $P_D = 1.0$; and the rest of the protein (pink) is blurred with $P_R = 0.5$, $P_C = 0.3$, and $P_D = 0.8$. (a) Biomolecular surface; (b) exterior tetrahedral mesh; and (c) exterior mesh with embeded protein. P_R, P_C and P_D are parameters to control the local resolution of residuals, chains and domains, respectively.

There are three important biomolecular surfaces [21, 22]: the Van der Waals surface (VdW), the Solvent Accessible Surface (SAS), and the Solvent Excluded Surface (SES). For the VdW, atoms are represented as rigid spheres with Van der Waals radii, and the biomolecular surface is defined as the envelope of these spherical surfaces. The SAS and SES can be defined by assuming a probe rolling around the biomolecule and keeping con-

tact with the atoms. Then the SAS is formed by tracing the trajectory of the probe center. The topological boundary of the union of all possible probes is called the SES, with no intersection with atoms. A lot of research has been conducted in approximating the SES, including the the alpha-shapes [28, 1], the beta-shapes [56, 105], the MSMS [106], the advancing front and generalized Delaunay approaches [59], NURBS approximation [6], and PDE-based methods [133, 149, 44]. The biomolecules were also represented as implicit models. The Gaussian kernel functions were applied in constructing density maps for the biomolecules [10, 38, 62, 147, 140]. In [37], the atomic resolution Gaussian density map was built and then filtered using an ideal filter to obtain a smooth biomolecular surface.

Fast computing is critical in biomolecular modeling [124, 129]. A variety of algorithms were developed in improving the modeling efficiency. For example, the Fast Fourier Transform was used in [149, 44] to get better performance of the PDE transform, and a Non-uniform Fast Fourier Transform (NFFT) was adopted to improve the polynomial-form summation of the kernel functions [7]. The programmable GPU has brought in a new direction for vast data processing in geometric modeling [126, 68, 94, 54, 118].

In this chapter, we propose an efficient multi-scale modeling framework for biomolecules based on the multi-level summation of Gaussian kernel functions [75]. The modeling process contains three steps: Gaussian density computation, adaptive tetrahedral mesh generation, and quality improvement. A special method called neighboring search is applied for efficiency improvement, together with the KD-tree structure and the bounding volume hierarchy (BVH). The multi-core CPU and GPU-based parallel computation techniques are employed in all these three steps. The main contributions in this chapter include:

1. Structure specified parameters are adopted in the multi-level summation of Gaussian kernel functions, enabling a local resolution control for the hierarchical structure of the complicated biomolecules;

2. An error-bounded biomolecule simplification method is introduced, which improves the computational efficiency by reducing the atom number;

3. Neighboring search is applied to locate the grid points close to the biomolecular surface, and thus reduce the number of grids to be analyzed;

4. KD-tree and BVH are employed to quickly search contributing atoms around a grid point. Faraway atoms are ignored due to the decay of Gaussian kernel functions; and

5. Multi-core CPU and GPU-based computation are employed in the entire modeling process, which significantly accelerate the modeling process.

The remainder of this chapter is organized as follows. Section 3.2 describes the multi-level summation of Gaussian kernel functions. Section 3.2.3 explains the Gaussian density map construction, promoted by the neighboring search, KD-tree structure, BVH, and parallel computation. Sections 3.3 and 3.4 talk about mesh generation and quality improvement, respectively. Section 4.5 shows the results. Finally, section 3.6 draws conclusions and points out the future work.

3.2 Gaussian Density Map of Biomolecular Complexes

3.2.1 Multi-level Summation of Gaussian Kernel Functions

As shown in Fig. 3.2, biomolecules usually have a complicated hierarchical structure, including the atomic, residual, and chain scales. A variety of methods have been developed to create multi-resolution models for the biomolecules [5, 147, 37, 133, 149]. However, most of the previous works only considered the overall resolution control, and the local resolution control was seldom studied. In this chapter, we improve the multi-level summation of Gaussian kernel functions and enable a local resolution control on the specific hierarchical structure in biomolecules.

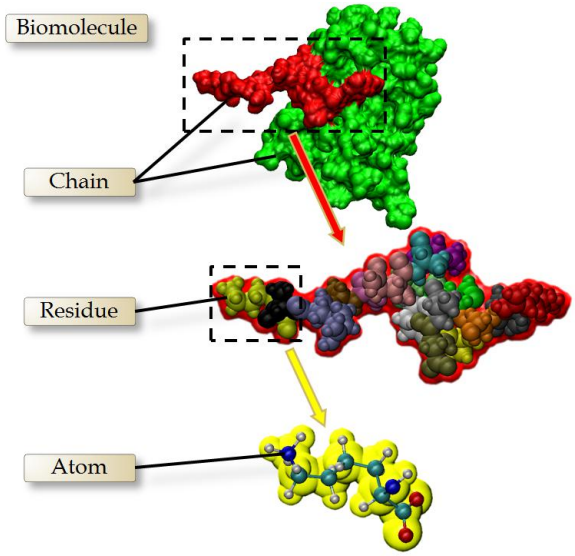


Figure 3.2: Hierarchical structure of the biomolecules.

Gaussian kernel functions were introduced in biomolecular modeling by Blinn *et al.* in 1982 [10]. The biomolecular surface is generated as a level set (isocontour) of the volumetric electron density map [3, 147]. Zhang *et al.* [147] improved the kernel functions to make the distance between the generated surface and the VdW (Van de Waals surface) surface as uniform as possible, resulting in smoother molecular surfaces. The kernel function is defined as

$$G_{i_A}(\mathbf{x}) = e^{\kappa\left(\|\mathbf{x}-\mathbf{x}_{i_A}\|^2 - r_{i_A}^2\right)}, \tag{3.1}$$

where κ is the decay rate, controlling how fast the Gaussian kernel function decays. x_{i_A} and r_{i_A} are the center and radius of the i_A^{th} atom, respectively. A multi-level summation of Gaussian kernel functions was applied to control the resolution of biomolecule models [147]. Lower level structures are classified into groups according to higher level structures. As the basic unit in the biomolecules, atoms are represented by $N_A = \{N_A^{(0)}, ..., N_A^{(n)}\}$. $N_R^{(i)}(i = 1, 2, ..., n_R)$ are subsets of N_A, representing the sets of residues. We have

$$\bigcup_{i=1}^{n_R} N_R^{(i)} = N_A, \quad and \quad N_R^{(i)} \bigcap_{1 \leqslant i \neq j \leqslant n_R} N_R^{(j)} = \varnothing. \tag{3.2}$$

The elements of $N_R := \{N_R^{(i)}\}_{i=1}^{n_R}$ are further grouped into subsets $N_C^{(i)}(i = 1, 2, ..., n_C)$, representing peptides:

$$\bigcup_{i=1}^{n_C} N_C^{(i)} = N_R, \quad and \quad N_C^{(i)} \bigcap_{1 \leqslant i \neq j \leqslant n_C} N_C^{(j)} = \varnothing. \tag{3.3}$$

Similarly, structures with a higher level (e.g., domains) can be represented as the subsets of peptides. The density distribution of a higher level structure is obtained through the summation of lower level density. For example, small proteins are made up of several peptides, so the density map is generated by a two-level summation of Gaussian kernel functions

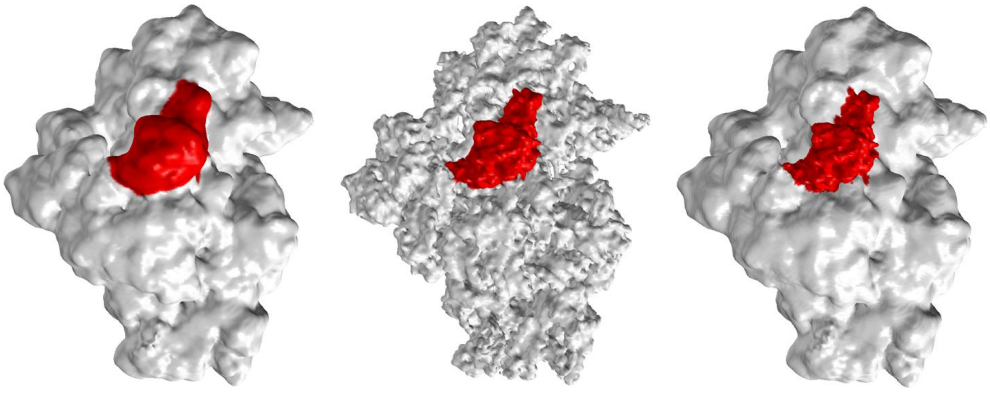
$$G(\mathbf{x}) = \sum_{i_C} \left(\sum_{i_R} \left(\sum_{i_A} G_{i_A}(\mathbf{x}) \right)^{P_R} \right)^{P_C}, \tag{3.4}$$

where i_C, i_R and i_A are the indices of the peptide, residue and atom, respectively. P_R and P_C are constant coefficients that control the local resolution of the model. G_{i_A} is defined in Eqn. 3.1.

To specify the structures at different levels in the biomolecule, non-uniform coefficients are selected based on the structure indices. For example, the coefficient for the residue level is defined as

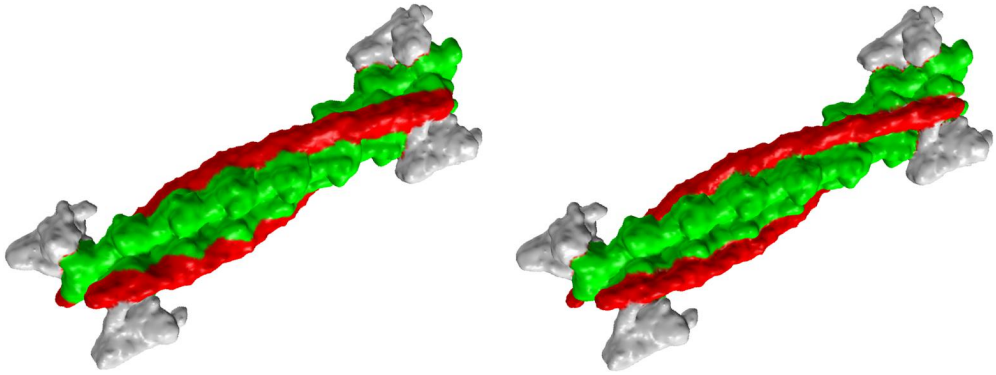
$$P_R = P_R(i_C, i_R), \tag{3.5}$$

where i_C and i_R are indices for the peptide and residue, respectively. The indices can specify the concerned peptide or residue and control the resolution on the surface locally. In Fig. 3.3(a), suppose we only want to emphasize the peptide B (marked in red) for Ribosome 30S (1J5E). If we vary the coefficients uniformly, all the atomic-level details on the biomolecular surface will be strengthened (Fig. 3.3(b)). However, varying only the coefficients for the structures contained by chain B results in a local resolution control on the certain peptide (Fig. 3.3(c)). Similarly, structures at any level (domain, chain or



(a) (b) (c)

Figure 3.3: Surface construction of 1J5E from a two-level summation of Gaussian kernels. (a) Uniformly blurred surface, $P_R = 0.05$ and $P_C = 0.5$; (b) all details on the model are strengthened, $P_R = 0.5$ and $P_C = 1.0$; and (c) only details on chain B are strengthened, $P_R = 0.5$ and $P_C = 1.0$ for chain B while $P_R = 0.05$ and $P_C = 0.5$ for the remaining structure.



(a) (b)

Figure 3.4: Resolution control of 2W4U at the domain level, $P_R = 0.7$, $P_C = 0.4$. (a) Blurred domain boundary for tropomyosin (red) and actin (green), $P_D = 0.25$ for the entire protein; and (b) more detailed features are preserved along the boundary of tropomyosin and actin, $P_D = 0.8$ for these two domains while $P_D = 0.25$ for the remaining structure.

residue) can be emphasized by adjusting the corresponding coefficients, enabling a flexible local resolution control at multiple scales.

Fig. 3.4 shows a macromolecular complex representing the thin filament subunit in muscle fibers (PDB 2W4U). This protein contains 32 peptides, and these chains are grouped into several higher level structures called protein “domains", including tropomyosin (red) and actin (green). A three-level (residue, chain and domain) summation

is utilized here to construct the density map

$$G(\mathbf{x}) = \sum_{i_D} \left(\sum_{i_C} \left(\sum_{i_R} \left(\sum_{i_A} G_{i_A}(\mathbf{x}) \right)^{P_R} \right)^{P_C} \right)^{P_D}, \tag{3.6}$$

where i_D is the domain index, and P_D is the coefficient corresponding to the domain level. Domain boundaries are blurred with a small coefficient P_D, resulting in a smooth biomolecular surface (Fig. 3.4(a)); when P_D is set to be a large value, more detailed features are preserved along the domain boundaries (Fig. 3.4(b)). Although P_D controls the transition region between different domains, the surface resolution inside each domain is mainly decided by the lower level coefficients P_C and P_R. To have a full control of the multi-scale biomolecular models, the number of levels in the summation of Gaussian kernel functions should be consistent with the level of biomolecular structures. Sometimes higher level structures consist of multiple chains, such as the the secondary and tertiary structures, are considered in the simulation. We can simply increase the level of summation to control the resolution at their scales.

Discussion 1: A two-level summation has two coefficients: P_C and P_R. P_C controls the boundary resolution of each chain, and P_R controls the surface resolution inside the chain. In a three-level summation, there is one more coefficient (P_D), which controls the boundary resolution of each domain. This local resolution control is very important in some applications such as the diffusional distribution simulation of the calcium ions in ventricular myocytes. Usually, the active site is a small region around certain residues. To simplify the implicit model, we can use a small P_D value to blur the domain boundary, and adjust the P_C and P_R values to make a smooth surface for the domain while showing more residue-level details at the active sites.

Discussion 2: Various kernel functions have been applied in building implicit models for biomolecules. For example, piece-wise constant kernels were adopted in [149], in which the density map must be filtered to obtain a smooth biomolecular surface. Compared

with other smooth kernel function, such as the radial basis function (RBF), the Gaussian kernel function defined in Eq. 3.1 can reflect the radius of atom without changing the decay rate.

3.2.2 Error-bounded Biomolecule Simplification

Error-bounded Atom Elimination

Let S_A be the set of all the atoms, S_R be the set of remaining atoms after eliminating the low-contributing atoms, and $G_R^-(\mathbf{x}, i)$ be the Gaussian density from all the atoms in S_R except Atom i. During the atom elimination, the change in Gaussian density around the biomolecular surface should be sufficiently small. In a rectilinear grid, a region Ω_S around the surface consists of a set of grid points $\mathbf{x}$ with Gaussian density $g_l < G(\mathbf{x}) < g_u$, here we choose $(g_l, g_u) = (0.9, 1.1)$. Due to the decay of Gaussian kernel functions, eliminating Atom i only influences the Gaussian density map in its neighboring region Ω_i, where all its grid points $\mathbf{x}$ satisfy

$$\|\mathbf{x} - \mathbf{x}_i\|^2 \leq -\frac{\ln \epsilon_r}{\kappa_i} + r_i^2, \tag{3.7}$$

and ϵ_r is a pre-defined threshold (e.g. $\epsilon_r = 10^{-6}$). The Gaussian density error e_G^i of ignoring Atom i can be defined as

$$e_G^i = \max_{\mathbf{x} \in \Omega_i \cap \Omega_S} \left| \frac{G(\mathbf{x}) - G_R^-(\mathbf{x}, i)}{G(\mathbf{x})} \right|. \tag{3.8}$$

In a rectilinear grid, the overall contribution C_i of Atom i to the surface is defined as

$$C_i = \left(\sum_{\mathbf{x} \in \Omega_i \cap \Omega_S} \left(\frac{G^-(\mathbf{x}, i)}{G(\mathbf{x})} \right)^2 \right)^{-1/2}, \tag{3.9}$$

where $G(\mathbf{x})$ and $G^-(\mathbf{x}, i)$ are the Gaussian density with and without considering Atom i, respectively. We sort all the atoms based on their overall contributions, and keep eliminating

the one with the lowest contribution until we find an atom with the Gaussian density error defined in Eq. (3.8) greater than ϵ_G, where ϵ_G is an input threshold. After atom elimination, we build the Gaussian density map with the remaining atoms and extract a triangle mesh to represent the simplified surface. The surface error can be measured as

$$e_G^M = \max_i \left| \frac{G(\mathbf{x}) - 1}{G(\mathbf{x})} \right|, \tag{3.10}$$

where $\mathbf{x}$ is a vertex on the biomolecular surface mesh. See Algorithm 1 for the details.

Algorithm 1 Atom elimination

$S_R := S_A$;
for each grid point $\mathbf{x}$ **do**
  Compute Gaussian density value $G(\mathbf{x})$;
end for
Identify the region Ω_S around the biomolecular surface, where $g_l < G(\mathbf{x}) < g_u$;
for each atom $i \in S_A$ **do**
  Identify its neighboring region Ω_i using Eq. (3.7);
  Compute the overall contribution C_i to the surface in $\Omega_i \cap \Omega_S$;
end for
Sort all the atoms in S_A based on C_i;
for the atom with the lowest contribution in S_R **do**
  Set this atom to be Atom i;
  Compute the Gaussian density error e_G^i using Eq. (3.8);
  if $e_G^i > \epsilon_G$ **then**
    STOP;
  else
    $S_R := S_R \setminus \{i\}$;
    Continue;
  end if
end for
Output the surface error e_G^M according to Eq. (3.10).

During Gaussian density map construction, a neighboring search and BVH-based GPU parallel method is employed for efficient computation [75]. Fig. 3.5 shows the simplification results of 2O53 with different ϵ_G values and coefficients (P_R, P_C). As ϵ_G increases, the Gaussian density error on the surface e_G^M tends to increase faster in concave and valley areas than the rest of the surface. As shown in Tab. 3.1, as the surface resolution is improved

with larger P_R, less atoms can be eliminated. This is because more atoms are exposed on the high resolution surface than the low resolution one.

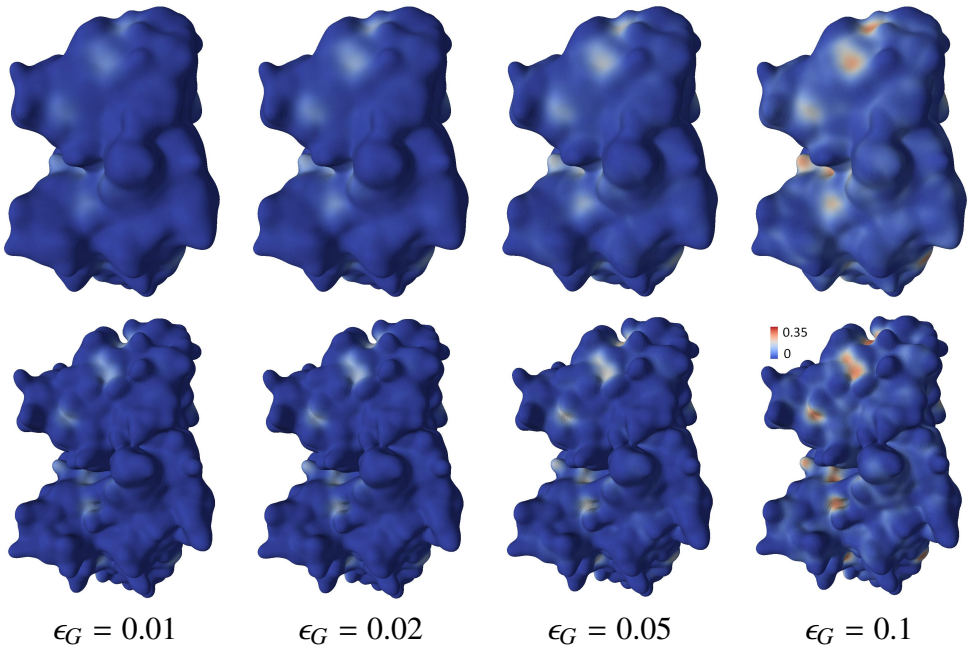


Figure 3.5: Biomolecular surfaces for the simplified protein 2O53 with $P_C = 1.0$ and $\kappa = 0.3$. Top row: $P_R = 0.25$; Bottom row: $P_R = 0.5$. The color represents the distribution of e_G^M.

Table 3.1: Atom Elimination Results for 2O53.

		$\epsilon_G = 0.0$	$\epsilon_G = 0.01$	$\epsilon_G = 0.02$	$\epsilon_G = 0.05$	$\epsilon_G = 0.1$
Case 1	N_R	4,912 (100%)	3,316 (67.5%)	3,011 (61.3%)	2,810 (57.2%)	2,557 (52.1%)
	e_G^M	0.0	0.021	0.032	0.057	0.221
	Optimized e_G^M	—	0.019	0.030	0.053	0.215
	T_G	0.92	0.55	0.52	0.47	0.42
Case 2	N_R	4,912 (100%)	3,662 (74.6%)	3,513 (71.5%)	3,362 (68.4%)	3,102 (63.2%)
	e_G^M	0.0	0.023	0.028	0.060	0.210
	Optimized e_G^M	—	0.021	0.024	0.057	0.208
	T_G	0.98	0.81	0.79	0.76	0.72

Note: Case 1 - $P_R = 0.25$; Case 2 - $P_R = 0.5$; N_R - the number of remaining atoms; atoms; e_G^M - the surface error defined in Eq. (3.10); and T_G - time for Gaussian density map computation (unit: second). The percentage of remaining atoms is shown in parentheses.

Optimization of Remaining Atoms

To reduce the surface error of the simplified biomolecule, the parameters of the remaining atoms can be optimized. Let N_R be the number of remaining atoms. The atoms in S_R are sorted, and their parameters are optimized using a gradient flow method. The objective function is defined as

$$\mathbf{F}(\mathbf{f}) = \int_{\mathbf{R}^3} (G(\mathbf{x}) - G_R(\mathbf{f}, \mathbf{x}))^2 \mathrm{d}\mathbf{x}, \tag{3.11}$$

and in a rectilinear grid we have

$$\mathbf{F}(\mathbf{f}) = \sum_{l=1}^{N_G} (G(\mathbf{x}_l) - G_R(\mathbf{f}, \mathbf{x}_l))^2, \tag{3.12}$$

where N_G is the number of grid points, and $\mathbf{f} = (x_1, y_1, z_1, r_1, \kappa_1, ..., x_{N_R}, y_{N_R}, z_{N_R}, r_{N_R}, \kappa_{N_R})^T$. For the k^{th} iteration of the gradient flow-based optimization, we have

$$\mathbf{f}^{k+1} = \mathbf{f}^k - \tau \nabla \mathbf{F}, \tag{3.13}$$

where τ is the step length, and

$$\nabla \mathbf{F} = \left(\frac{\partial \mathbf{F}}{\partial x_1}, \frac{\partial \mathbf{F}}{\partial y_1}, \frac{\partial \mathbf{F}}{\partial z_1}, \frac{\partial \mathbf{F}}{\partial r_1}, \frac{\partial \mathbf{F}}{\partial \kappa_1}, ..., \frac{\partial \mathbf{F}}{\partial x_{N_R}}, \frac{\partial \mathbf{F}}{\partial y_{N_R}}, \frac{\partial \mathbf{F}}{\partial z_{N_R}}, \frac{\partial \mathbf{F}}{\partial r_{N_R}}, \frac{\partial \mathbf{F}}{\partial \kappa_{N_R}} \right)^T.$$

As $\mathbf{F}$ is minimized step by step, the error in the Gaussian density map is minimized. In each step, the parameters of the remaining atoms are updated until the change in the parameters is less than a threshold ϵ_f,

$$\left\| \mathbf{f}^{k+1} - \mathbf{f}^k \right\|_\infty < \epsilon_f. \tag{3.14}$$

During the iteration in Eq. (3.13), the variation in $\mathbf{f}$ should be bounded. Let $\mathbf{x}_i^0$, r_i^0 and κ_i^0 be the original location, radius and decay rate of Atom i. If $\left\| \mathbf{x}_i^k - \mathbf{x}_i^0 \right\| > \rho_x$ (e.g., $\rho = 0.5$),

then

$$\mathbf{x}_i^k = \mathbf{x}_i^0 + \rho_x \frac{\mathbf{x}_i^k - \mathbf{x}_i^0}{\left\|\mathbf{x}_i^k - \mathbf{x}_i^0\right\|},$$

where ρ_x defines a sphere around $\mathbf{x}_i^0$ and the updated atom center should be inside the sphere. Similarly, the modifications in the radius $|r_i^k - r_i^0|$ and the decay rate $|\kappa_i^k - \kappa_i^0|$ are bounded to ρ_r and ρ_κ (e.g., $\rho_r = 0.5$ and $\rho_\kappa = 0.2$), respectively. As shown in Tab. 3.1, the surface error e_G^M can be improved after the optimization of remaining atoms.

3.2.3 Efficiency Improvement in Density Map Construction

Computational efficiency is critical in biomolecular modeling, especially for large biomolecular complexes with a complicated structure. When computing the Gaussian density for each grid point, if we consider the contribution from all the atoms in the biomolecule, the time complexity will be $O(MN)$, in which M is the number of atoms and N is the number of grid points. To improve the computational efficiency, the neighboring search algorithm is applied to reduce the number of grid points to be analyzed; and the KD-tree structure and a bounding volume hierarchy (BVH) are used to quickly find the contributing atoms for the Gaussian density at a grid point. Moreover, the multi-core CPU and GPU-assisted parallel computation are also employed to further accelerate the speed.

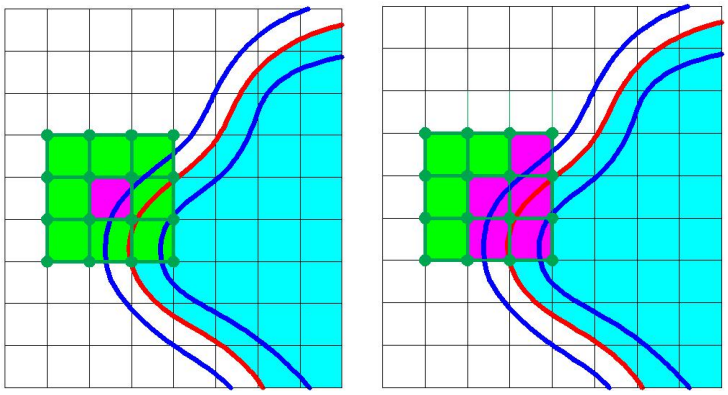
Neighboring Search

Biomolecular surface is extracted from the Gaussian density map as an isosurface. The grid cell intersecting with the isosurface is called a boundary cell. Vertices on the biomolecular surface are located either on the edges of (marching cubes [82]) or inside (dual contouring [48]) the boundary cell. Therefore, the biomolecular surface is only determined by the Gaussian density values at the grid points in boundary cells, while an accurate density calculation is not necessary for faraway grids. This property has been used in the accelerated isocontouring method [4], which detects the boundary cells to achieve a fast surface extraction.

Table 3.2: Ratio of analyzed grid points in 2KFX ($513 \times 513 \times 513$).

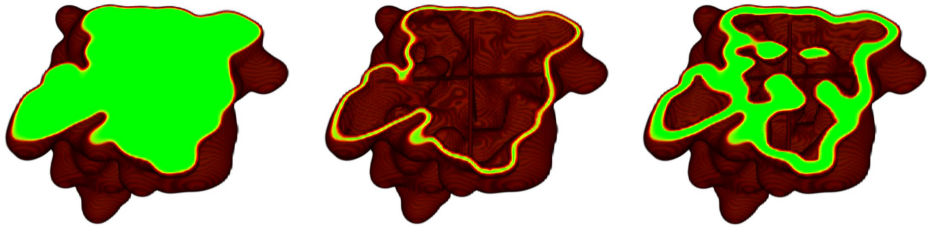
Lower threshold d_l	Upper threshold d_u	Analyzed ratio k_A
0.9	1.1	0.026
0.6	1.4	0.048
0.3	1.7	0.092
0.1	2.0	0.12

In this chapter, a similar idea is adopted and an algorithm called *neighboring search* [2] is applied to reduce the number of grid points to be analyzed. This algorithm can find out all the grid points near the expected isosurface, as long as the biomolecular surface is a manifold. Therefore, we only need to calculate the Gaussian density at these grid points, while the density value for the other grids is simply set as a constant. k_A is the ratio of the analyzed grid points over the total grid points. k_A is controlled by a pair of predefined parameters: a lower threshold d_l and an upper threshold d_u. Fig. 3.6 shows an example of neighboring search. The blue curves represent the isosurfaces of d_l and d_u. The red curve represents the target surface. The green cells are the 1-ring neighbors of a given activated grid cell (magenta). The initially activated cell can be found by searching along a line passing through the center of the grid. For each neighboring cell, the Gaussian density values are computed at the eight vertices. If the density indicates that the cell intersects with the band between the blue curves, the cell is activated. More cells can be activated around the newly activated cells iteratively until there are no more updates. As a result, all the grid points close to the surface are analyzed, while faraway points are ignored. Using a flood fill algorithm, those faraway points are marked as interior or exterior grids. Fig. 3.7 shows the neighboring search results for 2KFX. In (a), the green region represents the inside volume; and in (b) and (c), the green regions represent the band between d_l and d_u. k_A varies with different d_l and d_u, and also depends on the resolution of the grids. It is generally a very small value, see Table 3.2.



(a) (b)

Figure 3.6: The neighboring search algorithm. (a) The activated grid cell (magenta) and its 1-ring neighbors; and (b) the newly activated cell (magenta) in the 1-ring neighbors.



(a) (b) (c)

Figure 3.7: Neighboring search results for 2KFX. (a) The interior domain (green); (b) one detected band using $d_l = 0.9$, $d_u = 1.1$ and $k_A = 0.026$; and (c) another detected band using $d_l = 0.1$, $d_u = 2.0$ and $k_A = 0.12$.

Discussion: Neighboring search can effectively improve the complexity of Gaussian density computation from $O(MN)$ to $O(k_A MN)$, in which k_A is output-dependent. This method is based on the assumption that the biomolecular surface is a manifold and all the grid points close to the expected surface are neighboring to each other. But sometimes biomolecules have isolated components. If any isolated component is missed in the initial search, the modeling result would be incorrect. However, if the biomolecular structure is known beforehand, the initial search route can be easily modified to include all the components.

KD-tree Structure and Bounding Volume Hierarchy

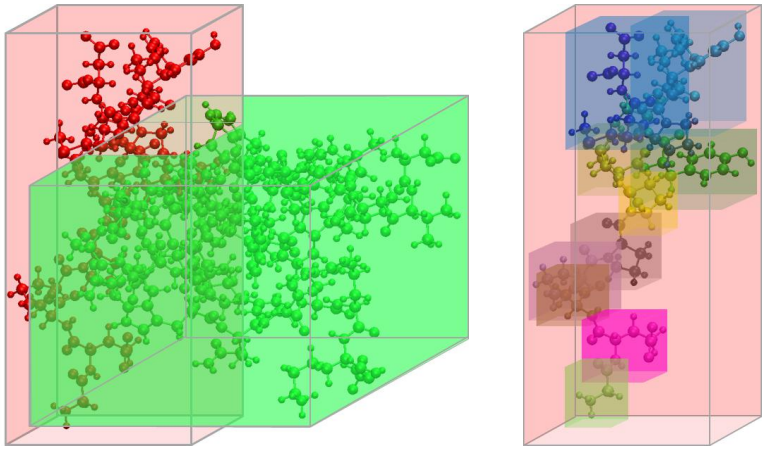
A Gaussian kernel function decays quickly as it moves away from the atom center. For one grid point, it is reasonable to ignore faraway atoms, which contribute little to the Gaussian density map. Two methods are used here to quickly find the nearby atoms: the KD-tree structure and the bounding volume hierarchy (BVH).

The KD-tree structure is a widely used space-partitioning data structure, which has been employed in biomolecular modeling [54, 55, 109]. Differently, in this chapter a KD-tree is used to quickly search atoms around grid points. The cost to build this data structure is $O(M \log M)$. Searching atoms within a certain range around a grid point becomes quite efficient due to the binary tree structure. The error of ignoring faraway atoms can be controlled by a bounding radius R_{kd} in the KD-tree search. Suppose an atom is ignored when its density contribution is less than ε, then R_{kd} should satisfy $e^{\kappa(R_{kd}^2 - r^2)} < \varepsilon$. We have

$$R_{kd} > \sqrt{\frac{\ln \varepsilon}{\kappa} + r^2}, \tag{3.15}$$

where r is the radius of the atom; and κ is the decay ratio in Eqn. 3.1. Pratically, the maximum atom radius is usually 2.0Å and the decay ratio is 1.0. When $\varepsilon = 10^{-6}$, the bounding radius $R_{kd} = 4.3$ ensures that no atom around a grid point has a density contribution smaller than ε. For each analyzed grid point, it takes $O(\log M)$ to find the nearby atoms and compute the Gaussian density.

The bounding volume hierarchy is a tree structure on a set of geometric objects. All geometric objects are wrapped in bounding volumes that form the leaf nodes of the tree. This technique has been widely used in computer visulization, and also employed in protein structure representation [30, 83, 109]. In this chapter, edges of the bounding volumes are along the coordinate axes. The smallest box that contains all the atom centers belonging to a structure is called the minimum bounding box of this structure. The minimum bounding box is generally expanded by a certain ratio. The margin of the bounding box is constraint



(a) (b)

Figure 3.8: The bounding box system. (a) The bounding boxes of two peptides; and (b) the bounding box of one peptide is subdivided into bounding boxes of residues.

with at least $R_{BVH} = 4.3$ to make sure all the contributing atoms can be searched (similar with Eqn. 3.15). The bounding box of a higher level structure contains bounding boxes of lower structures. Fig. 3.8 shows an example of the BVH for a protein with an atom-residue-chain structure. In Fig. 3.8(a), the space is subdivided into the bounding boxes (red and green) of two peptides. The bounding boxes of peptides are further subdivided into the bounding boxes of residues (Fig. 3.8(b)). The subdivision allows overlaps and gaps between the children boxes. Searching nearby atoms follows a top-down order in the BVH tree structure. The time complexity for the BVH searching can be $O(\log M)$, but in practice the efficiency would be a bit worse because the structures usually have many overlaps. Moreover, as the atom order is organized following the hierarchical structure, the construction of the BVH tree can be easily conducted.

Parallel Gaussian Density Computation

To further accelerate the speed, multi-core CPU and GPU-based parallel computation are applied in the Gaussian density calculation. In each step of the neighboring search, the grid points to be analyzed are distributed to different threads, and the returned Gaussian density value is used as the input of the next step. The workload for various grid points can be very different because the number of contributing atoms can vary a lot. For the 8-core CPU

mode, the grid points are randomly distributed to different cores and the computation keeps at full load for most of the time. Differently, the workload imbalance can seriously influence the performance of the GPU-based computation. Therefore for GPU computation, the contributing atom number of the previously analyzed grids are recorded, and the workload of the newly analyzed grid points is estimated using these numbers. Grid points with similar estimated workload are distributed to the same GPU blocks to achieve a better balance.

As shown in Fig. 3.9, a variety of proteins are tested using neighboring search, the KD-tree structure and the BVH. A 2-level summation of Gaussian kernel functions are used, with $P_R = 0.5$ and $P_C = 1.0$. Time costs are listed in Table 3.3. The rectilinear grid for each protein is large enough to contain all the atom centers inside, and the margin on each dimension is 4.0 Å. For all the proteins, we fix the resolution of the grid to be 0.25 Å. The CPU-based results in Table 3.3 are obtained under 8-core parallel computation. All the results are generated with an Intel E5-1620 CPU, a Nvidia GeForce GTX680 graphic card and 16GB memory. The algorithms are implemented in C++ by using OpenMP for the multi-core CPU computation and CUDA 5.0 for the GPU computation. T_0 is the time cost of CPU computation without any acceleration algorithms. Neighboring search, KD-tree structure and BVH are compared in CPU-based implementations.

Generally, as the atom number M increases, the speedups for neighboring search, KD-tree and BVH all become more significant. For KD-tree and BVH, the time complexity of Gaussian density map generation is improved to $O(N \log M)$, compared with the original $O(NM)$. Therefore, it is easy to understand that the speedups of KD-tree and BVH increase as M gets larger. For neighboring search, the corresponding time complexity is $O(k_A NM)$. At a certain resolution, k_A decreases as M gets larger because generally the surface area grows much slower than the grid size increases. Therefore, the speedup of neighboring search also increases as M gets larger. Besides atom number, the efficiency speedups are also affected by the biomolecular structure. The performance of BVH is directly related to the hierarchical structure of the biomolecules, while the KD-tree is not. For example,

Table 3.3: Efficiency comparison for neighboring search, the KD-tree and BVH. (Grid resolution: 0.25Å; Unit: second)

PDB ID	M	N	T_0	$T_{CPU(NS)}$	$T_{CPU(KD)}$	$T_{CPU(BVH)}$	$T_{CPU(NS+KD)}$	$T_{GPU(NS+BVH)}$
1BOR	832	$147 \times 149 \times 154$	3.89	0.76(×5.14)	1.81(×2.15)	2.08(×1.87)	0.65(×6.01)	0.20(×19.44)
1NEQ	1,187	$196 \times 179 \times 175$	8.83	1.08 (×8.21)	2.48(×3.55)	2.73(×3.23)	0.93 (×9.53)	0.25(×35.33)
1A63	2,065	$275 \times 176 \times 183$	21.60	2.56(×8.45)	5.61(×3.85)	6.57(×3.29)	1.50 (×14.40)	0.33(×65.45)
1A7M	2,809	$188 \times 260 \times 218$	34.93	3.37(×10.36)	8.91(×3.92)	9.87(×3.54)	2.03(×17.23)	0.51(×68.49)
1BEB	4,972	$232 \times 246 \times 324$	107.38	8.85(×12.13)	18.77(×5.72)	20.65(×5.20)	2.99(×35.94)	0.92(×116.71)
1VNG	8,808	$319 \times 258 \times 307$	262.19	20.36(×12.88)	36.42(×7.20)	35.43(×7.40)	6.19(×42.37)	1.20(×218.49)
1GTP	69,980	$915 \times 591 \times 345$	15,496.44	820.79(×18.88)	2,060.70(×7.52)	501.83(×30.88)	162.01(× 95.65)	11.33(×1,367.74)
2KXH	70,200	$281 \times 284 \times 256$	2,307.54	159.50(×14.47)	253.02(×9.12)	288.08(×8.01)	20.69(×111.54)	2.71(×851.49)

Note: M - the atom number; N - the grid size; T_0 - CPU computation without any acceleration algorithm; NS - neighboring search; KD - KD-tree structure; BVH - bounding volume hierarchy. The numbers in parentheses are the speedups compared to T_0.

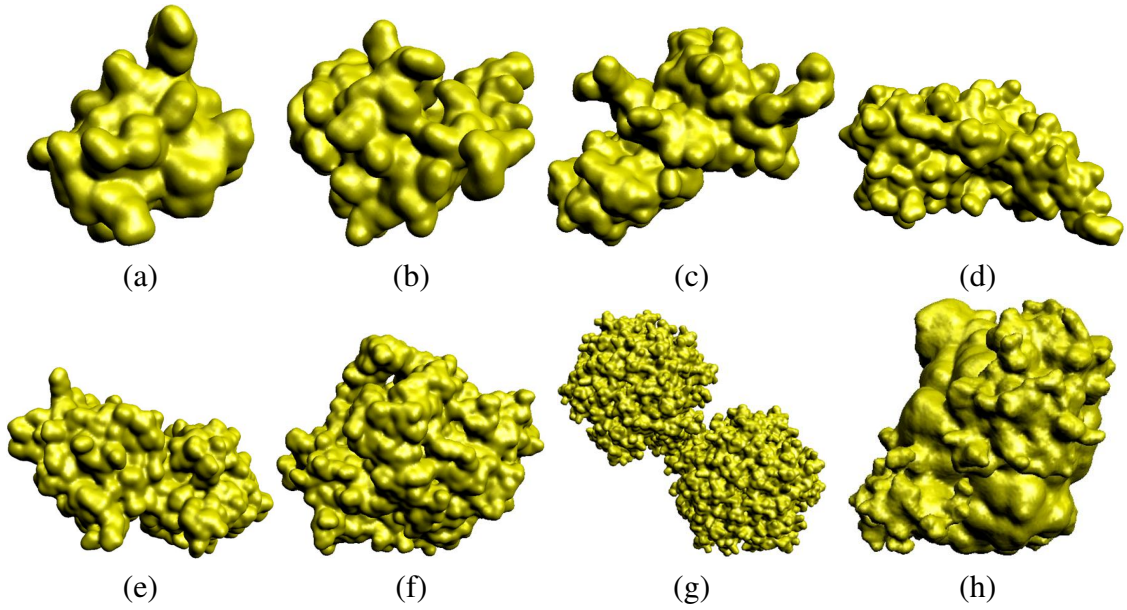


Figure 3.9: Eight proteins tested in Table 3.3. (a) 1BOR; (b) 1NEQ; (c) 1A63; (d) 1A7M; (e) 1BEB; (f) 1VNG; (g) 1GTP; and (h) 2KXH.

1GTP and 2KXH have similar atom numbers, but the structures in 2KXH are much tighter than in 1GTP, leading to many more overlaps among the bounding boxes. Therefore, the speedup of BVH for 2KXH (8.01 times) is not as significant as 1GTP (30.88 times).

$T_{CPU(NS+KD)}$ and $T_{GPU(NS+BVH)}$ shows the effect of different combinations of the techniques. As the hierarchical structure information is ignored in the KD-tree, a re-organization of the atoms is required for multi-level summation of Gaussian kernel functions. The re-organization needs a lot of memory. Therefore instead of KD-tree, the BVH is chosen for GPU-based computation. The combination of neighboring search, KD-tree structure and multi-core CPU computation results in a speedup varying from 6.01 to 111.54 times. For the combination of neighboring search, BVH and GPU computation, the speedup can range from 19.44 times for the smallest protein (1BOR) to 1,367.74 times for the largest protein (1GTP), due to the highly parallel computational capability of the GPU and the maximum effect of neighboring search.

3.3 Mesh Generation

The dual contouring method [48, 145] is applied to generate adaptive tetrahedral meshes from the Gaussian density map. The multi-core CPU and GPU-based parallel computation are employed to speed up mesh generation.

3.3.1 Dual Contouring Method

The biomolecular surface is defined as an isosurface of the Gaussian density map, from which tetrahedral meshes can be extracted using the dual contouring method [145, 147]. A strongly-balanced octree structure is built from the rectilinear grid that contains the Gaussian density, and the mesh adaptation is controlled by a feature sensitive error function [145]. To resolve topology ambiguities, we detect ambiguious cells using a trilinear function, and then split them into tetrahedral cells [146]. For each octree cell, the minimum and maximum (min-max) density values are calculated, making it easy to tell if the octree cell is inside or outside the domain to be meshed. For each leaf cell, a dual vertex is generated and the tetrahedral mesh is constructed by connecting the dual vertices with octree grids. For each octree boundary cell that intersects with the biomolecular surface, we choose the mass center as the dual vertex. The mass center is defined as the the average of all the intersection points between the biomolecular surface and the cell edges. For interior leaf cells, the dual vertex is the cell center.

Tetrahedral elements are generated around each minimal edge, which is defined as an edge of a leaf cube that do not properly contain any edge of its neighbors. The minimal edge intersecting with the biomolecular surface is called a sign change edge, and those inside the domain to be meshed are called interior edges. For each sign change edge, we first find out all its surrounding leaf cells and obtain three or four dual vertices. These dual vertices and the interior grid point of this edge form a tetrahedron or a pyramid. For each interior edge, we also obtain three or four dual vertices. Differently, these dual vertices

and two endpoints of this edge form a pyramid or a diamond. Later, the pyramids and diamonds can be splitted into tetrahedra. The ambiguous leaf cells are split into tetrahedra, and meshes are generated by analyzing the edges of these tetrahedra [146].

3.3.2 Paralleled Mesh Generation

GPU-based computation has been used in isocontouring for surface mesh generation [107]. Differently, in this chapter we aim to apply multi-core CPU and GPU-based techniques to the dual contouring method for adaptive tetrahedral mesh generation. In the twelve edges of one leaf cell, at least three of them are independent. As shown in Fig. 3.10, we divide all the edges into four groups (orange, green, blue and red), and analyze one group in each step. For example, the orange group (edge e_{01}, e_{03} and e_{04}) is analyzed for all the leaf cells. Each octree cell is distributed to a CPU or GPU thread, and the connectivity can be constructed in a parallel way. During octree subdivision, the most time-consuming step is the min-max computation, which is more expensive for the lower level octree cells with more grid points. To improve the workload balance, octree cells at the same or similar levels are distributed to the same GPU blocks.

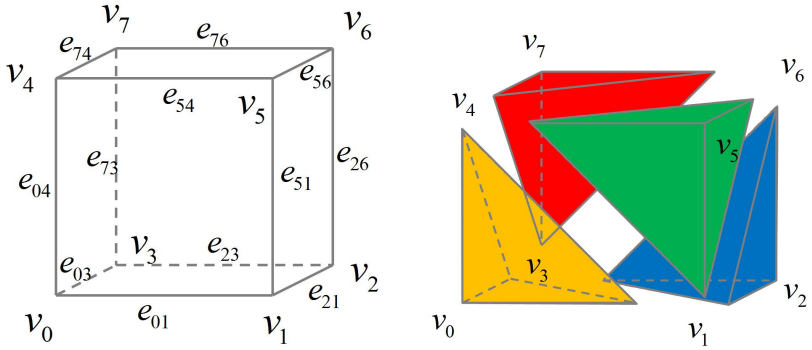


Figure 3.10: Group of the edges.

3.4 Quality Improvement

In our generated tetrahedral meshes, most elements are in good quality except some elements around the boundary. Therefore, we need to improve the mesh quality. First, let us choose a few metrics to measure mesh quality [145, 64]: the edge ratio, the Jue-Liu parameter [77], and the dihedral angle. The edge ratio is the ratio of the longest edge length over the shortest edge length in one element. The Joe-Liu parameter is defined as

$$Q = \frac{8 \cdot 3^{\frac{5}{2}} V}{\left(\sum_{j=1}^{6} e_j^2\right)^{\frac{3}{2}}} \tag{3.16}$$

where $\{e_j\}_{j=1}^{6}$ are six edge lengths, and V is the volume of a tetrahedron.

Three techniques are applied to improve the mesh quality: face swapping, edge contraction, and geometric flow [64]. Both face swapping and edge contraction are operations of topological optimization. Face swapping reconnects vertices of some elements, while edge contraction removes a few poor quality elements. Differently, geometric flow relocates vertices iteratively to improve the overall mesh quality. Generally speaking, face swapping and edge contraction only change the local topology for a few elements ($< 1\%$), and the process is very fast. Geometric flow smoothing needs to relocate vertices overally, which is the most time-consuming part for quality improvement. Using a similar data structure with [24], the parallel computation is applied to the smoothing step. Vertices are relocated and updated one by one. In the CPU-based computation, METIS [52] is employed to partition the mesh, and vertices in different parts are independent from each other. The location of the vertices on the shared boundaries are synchronized after each smoothing step.

For GPU-assisted computation, vertices are relocated after each step to avoid any conflict. The workload imbalance is the main concern in the implementation. For adaptive meshes, vertices with a large valence number may happen, which can cause serious workload imbalance during the GPU-based geometric flow smoothing. Therefore, vertices are

grouped according to their valence number, and the ones with similar valence number are distributed to the same GPU blocks.

3.5 Results and Discussion

The biomolecule simplification and Gaussian density computation methods introduced in this chapter make up an efficient process for the geometric modeling of biomolecular complexes, together with the CPU- and GPU- based parallel computation techniques. In this section, various biomolecular complexes are tested using our methods. All the results are generated from a computer with an Intel Xeon E5-1620 CPU, a Nvidia GeForce GTX680 graphic card, and 16GB of memory.

3.5.1 Biomolecule Simplification Results

Tab. 3.4 shows the results of atom simplification for proteins with various sizes. Among them, 2O53 is a protein in human brain, and one of its two identical components is emphasized. 4KYT, 4N78, 4A7F and 2W4U are involved in the heart contractile process. Components with important biological functions are chosen to be emphasized. 2KU2 is one of the largest proteins in the PDB with 1.23M atoms. One of its seven symmetric components is chosen to be emphasized. As shown in Tab. 3.4, the percentage of remaining atoms varies from 39.3% to 69.6%, and tends to decrease as the protein size increases because most atoms are buried inside the biomolecular surface. The number of remaining atoms also depends on the size of emphasized components. For example in 2O53, when the emphasized components are considered, the percentage of remaining atoms increases by 16.3%. Contrast to the other proteins, the emphasized components in 4A7F are exposed to the surface, and most of their atoms are kept during simplification. Therefore the number of remaining atoms does not change much when considering these emphasized components.

Table 3.4: Simplification results for biomolecules.

	No Simplification		Atom Simplification (no emphasized chains)			Atom Simplification (with emphasized chains)		
PDB ID	N_A	T_G	N_R	T_G	e_G^M	N_R	T_G	e_G^M
2O53	4,912	0.92	3,316 (67.5%)	0.55	0.019	4,114 (83.8%)	0.87	0.018
4KYT	7,908	1.35	5,377 (68.0%)	0.97	0.018	5,501 (69.6%)	0.99	0.018
4N78	22,843	3.26	14,436 (63.2%)	2.13	0.023	15,107 (66.1%)	2.33	0.021
4A7F	33,500	3.88	20,134 (60.1%)	2.42	0.026	20,465 (62.7%)	2.47	0.026
2W4U	0.14M	16.21	74,962 (54.4%)	10.13	0.032	78,853 (57.2%)	10.98	0.031
2KU2	1.23M	33.21	0.51M (39.3%)	15.81	0.055	0.64M (49.3%)	17.48	0.051

Note: N_A - the number of all the atoms; N_R - the number of remaining atoms; and T_G - time for Gaussian density map computation (unit: second). $\epsilon_G = 0.01$ for the atom elimination. The percentage of remaining atoms is shown in the parentheses.

3.5.2 Geometric Modeling Results

In this section, all the steps in our multi-scale modeling, including Gaussian density computation, mesh generation and quality improvement, are tested in three modes: CPU sequential, CPU 8-core parallel and GPU parallel. Among the tested biomolecular complexes (see Figs. 3.1, 3.11 and 3.12), 2W4U, the CERCA system and the thin filament group (TFG) are from the human cardiac calcium signaling system, while 2W4A, 1HTQ and 2KU2 are chosen from the Protein Data Bank (PDB). A 3-level summation of Gaussian kernel functions is applied for 2W4U, while the other models use a 2-level summation. Table 3.5 shows the modeling results.

2W4U: 2W4U is an important thin filament in myofibrils, which is also the key functional part in the muscle fibers. The contraction of muscle fibers rely heavily on the interaction between the myosin heads and the filaments. To study the calcium ion signaling process during the muscle contraction, a cuboidal outer boundary is inserted around the thin filament, and the exterior tetrahedral meshes are generated, see Fig. 3.1.

CERCA system: The CERCA system works as a pump for the calcium ions in the signaling. The system contains a segment of lipid and a protein 1SU4 intersecting with it. As shown in Fig. 3.11(a-c), a cuboidal boundary is insert around it and intersecting with the membrane. The exterior mesh is generated which is divided by the membrane into two separated parts.

Thin filament group (TFG): During the heart muscle contraction, the calcium signaling process is a combination of the interaction between the thin and thick filaments. A model containing six thin filaments and four thick filaments is built with a cuboidal outer boundary intersecting with the filaments, see Fig. 3.11(d-f). The exterior tetrahedral mesh is generated to represent the environment around the filaments. The thin filaments here contains six myosin heads, and the thick filaments are represented by cylinders. The density map for a single thin filament is generated using the multi-level summation of Gaussian

Table 3.5: Time cost for three testing modes (CPU sequential, CPU 8-core parallel and GPU parallel) and mesh statistics. (Unit: second)

Complex	2W4U			CERCA			TFG			2W4A			1HTQ			2KU2		
M	69,222			80,157			95,853			0.19M			0.98M			1.23M		
Mode	CS	C-8	G	CS	C-8	G	CS	C-8	G	CS	C-8	G	CS	C-8	G	CS	C-8	G
DM	188.6	37.8 (×5.0)	4.8 (×39.3)	165.1	38.4 (×4.3)	11.7 (×14.1)	377.5	75.6 (×5.2)	7.3 (×51.7)	687.4	143.5 (×4.8)	10.2 (×67.4)	6,494.7	1589.4 (×4.1)	88.4 (×73.5)	3,050.2	645.5 (×4.7)	63.2 (×48.3)
MG	13.2	6.6 (×2.0)	2.4 (×5.5)	12.8	6.3 (×2.0)	2.3 (×5.6)	13.1	6.7 (×2.0)	2.4 (×5.5)	12.9	6.7 (×1.9)	2.4 (×5.4)	13.1	6.7 (×2.0)	2.7 (×4.9)	12.9	6.5 (×2.0)	2.4 (×5.4)
QI	157.8	35.8 (×5.0)	6.3 (×25.2)	163.5	37.7 (×4.3)	7.2 (×22.7)	189.5	36.7 (×5.2)	5.8 (×32.7)	213.7	53.2 (×4.0)	6.5 (×32.9)	250.6	68.3 (×3.7)	12.5 (×20.0)	178.0	44.2 (×4.0)	6.7 (×26.6)
Total	359.6	80.2 (×4.5)	13.5 (×26.6)	341.4	82.4 (×4.1)	21.2 (×16.1)	580.1	119.0 (×4.9)	15.5 (×37.4)	914.0	203.4 (×4.5)	19.1 (×47.9)	6,758.4	1,664.4 (×4.1)	103.6 (×65.2)	3241.1	696.2 (×4.7)	72.3 (×44.8)
Mesh size (V#, E#)	(212,120, 1,094,424)			(190,356, 946,273)			(271,523, 1,327,336)			(276,458, 1,334,008)			(399,994, 1,989,245)			(243,463, 1,201,048)		
Edge ratio (min, max)	(1.0, 10.24)			(1.0, 10.23)			(1.0, 10.31)			(1.0, 10.98)			(1.0, 9.56)			(1.0, 10.60)		
Joe-Liu (min, max)	(0.12, 1.0)			(0.14, 1.0)			(0.18, 1.0)			(0.11, 1.0)			(0.10, 1.0)			(0.15, 1.0)		
Dihedral angle (min, max)	(15.00°, 165.25°)			(15.01°, 167.25°)			(15.03°, 164.20°)			(15.10°, 166.23°)			(15.03°, 167.46°)			(14.86°, 167.42°)		

Note: *M* - atom number; CS - CPU sequential; C-8 - 8-core CPU parallel; G - GPU parallel; DM - time for density map construction; MG - time for mesh generation; QI - time for quality improvement; (V#, E#) - (vertex number, element number).

kernel functions, it is used for the assembly of the intact density map containing six thin filaments. The assembly step costs about ten seconds, which is not count in Table 3.5.

As shown in Fig. 3.12, three large proteins are chosen from the Protein Data Bank, and tested using our modeling methods. **2W4A** is a contractile protein for insect flight muscle. **1HTQ** is a glutamine synthetase from Mycobacterium tuberculosis, the second largest protein we found in the PDB. **2KU2** is a hydrolase protein, which is also the largest protein we found in the PDB.

In Gaussian density map generation, the rectilinear grid size for the Gaussian density map is constraint to be $513 \times 513 \times 513$. For the CPU-based computation, neighboring search and the KD-tree structure are applied in both the sequential and parallel modes; while the combination of neighboring search and BVH is employed in the GPU-based mode. The speedups of the 8-core CPU computation are similar for all the complexes (4.1~5.2), while the speedups of the GPU-assisted computation vary from 14.1 to 73.5 times. The worst speedup of the GPU computation happens on the CERCA system, as shown in Fig. 3.11(a). In CERCA, the lipid (pink) contains most of the atoms without a hierarchical structure, which greatly reduces the subdivision efficiency of the BVH.

In both 8-core CPU and GPU-based computation, the time costs of mesh generation are similar for all the tested models although the mesh size differs. This is because in mesh generation, the most time-consuming step is octree subdivision. Compared to the uniform grids, the adaptive octree construction requires a much more complicated index system, which significantly limits the speedups. From Table 3.5, we can observe that the speedups are 1.9~2.0 times for the 8-core CPU computation, and 4.9~5.6 times for the GPU-based computation.

During quality improvement, for both the CPU sequential and the 8-core CPU computation, we can observe that the time cost increases as the mesh size becomes larger. However, it is more complicated for the GPU-based computation due to its high sensitivity to the vertex valence number, which also brings in serious workload imbalance on different threads.

As discussed in Section 3.4, high valence numbers are introduced by mesh adaptation. In particular, 2W4U, the CERCA system, and 1HTQ contain large volumes, and they require more adaptive meshes to reduce the mesh size. Due to this reason, a workload imbalance is introduced in their GPU computation. Therefore, for these three models the speedups are only about 20.0~25.2 times, not as significant as the others (26.6~32.9 times).

For the whole modeling process, the 8-core CPU computation introduces a similar speedup on the efficiency for all the tested proteins (4.1~4.9 times), while the speedups of the GPU-assisted computation vary from 16.1 to 65.2 times. In addition, from Table 3.5 we can observe that the obtained meshes are in good quality with the minimal dihedral angle $\geq 14.86^\circ$.

3.6 Conclusion and Future Work

In this chapter, a multi-level summation of Gaussian kernel functions is applied to generate multi-scale implicit models for the biomolecules. Structures at different levels are specified and emphasized with more details on the local surface. The computational efficiency is improved by using a combination of neighboring search, KD-tree structure and bounding volume hierarchy. The CPU and GPU-assisted parallel computation techniques are employed in all the modeling steps, including Gaussian density map construction, mesh generation and quality improvement. In our approach, large proteins can be modeled quickly with quality adaptive tetrahedral meshes as output.

In the future, it is worth to apply the multi-level summation of Gaussian kernel functions in various application problems such as the diffusion simulation and boundary element solvers of the Poisson-Boltzmann equation [34, 35, 84]. In particular, the study of large biomolecules will benefit a lot from the high efficiency introduced by our techniques. The multi-level summation can also be extended to work for a combination of biomolecular information from the Protein Data Bank and Cryo-EM scanned images.

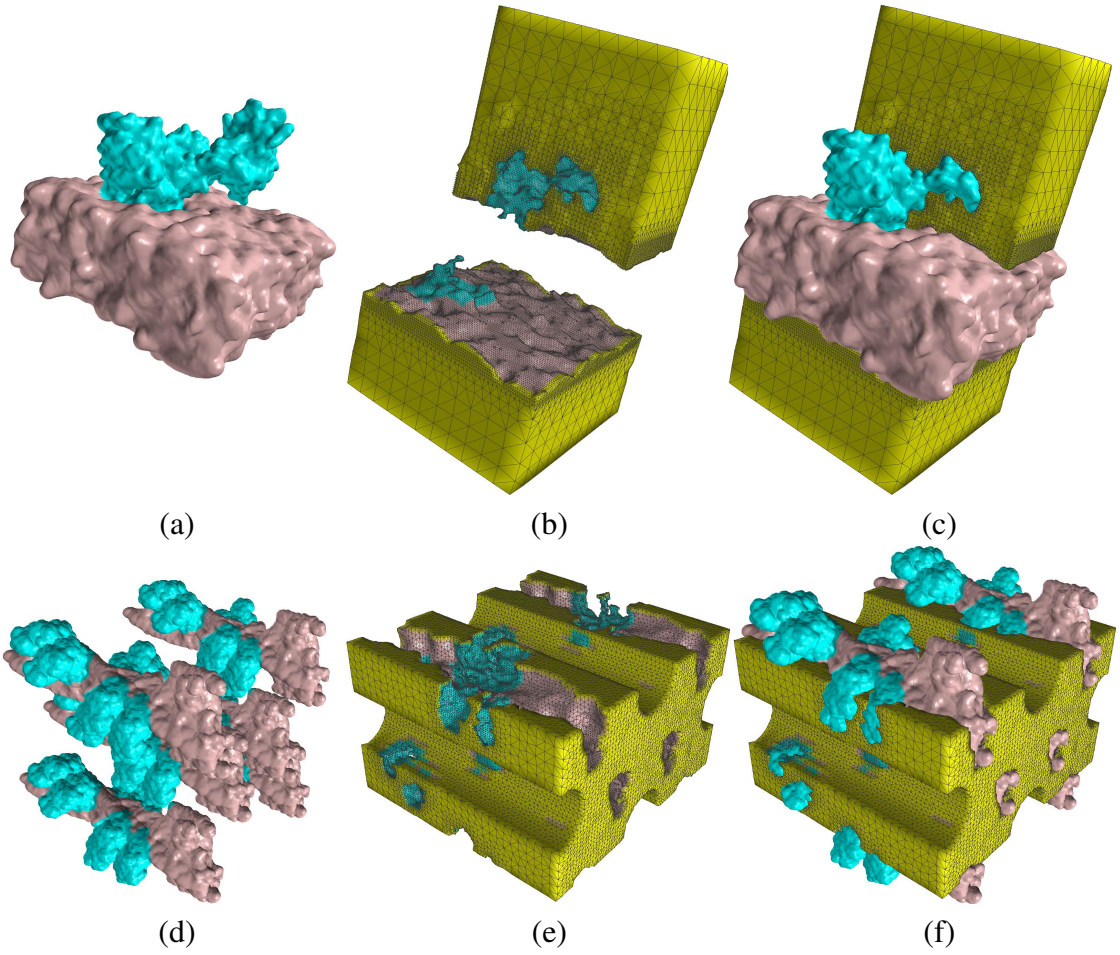


Figure 3.11: Two biomolecualr complexes in human cardiac calcium signaling system. (a-c) CERCA system with $P_R = 0.2$, $P_C = 0.8$ and $P_R = 0.2$, $P_C = 1.1$ for low and high resolution components, respectively; and (d-f) thin/thick filament group (TFG) with $P_R = 0.5$, $P_C = 0.3$ and $P_R = 0.7$, $P_C = 0.4$ for low and high resolution components, respectively. (a) and (d) show the protein structure with the blue region emphasized in high resolution; (b) and (e) show exterior tetrahedral meshes; and (c) and (f) show exterior meshes with embeded proteins.

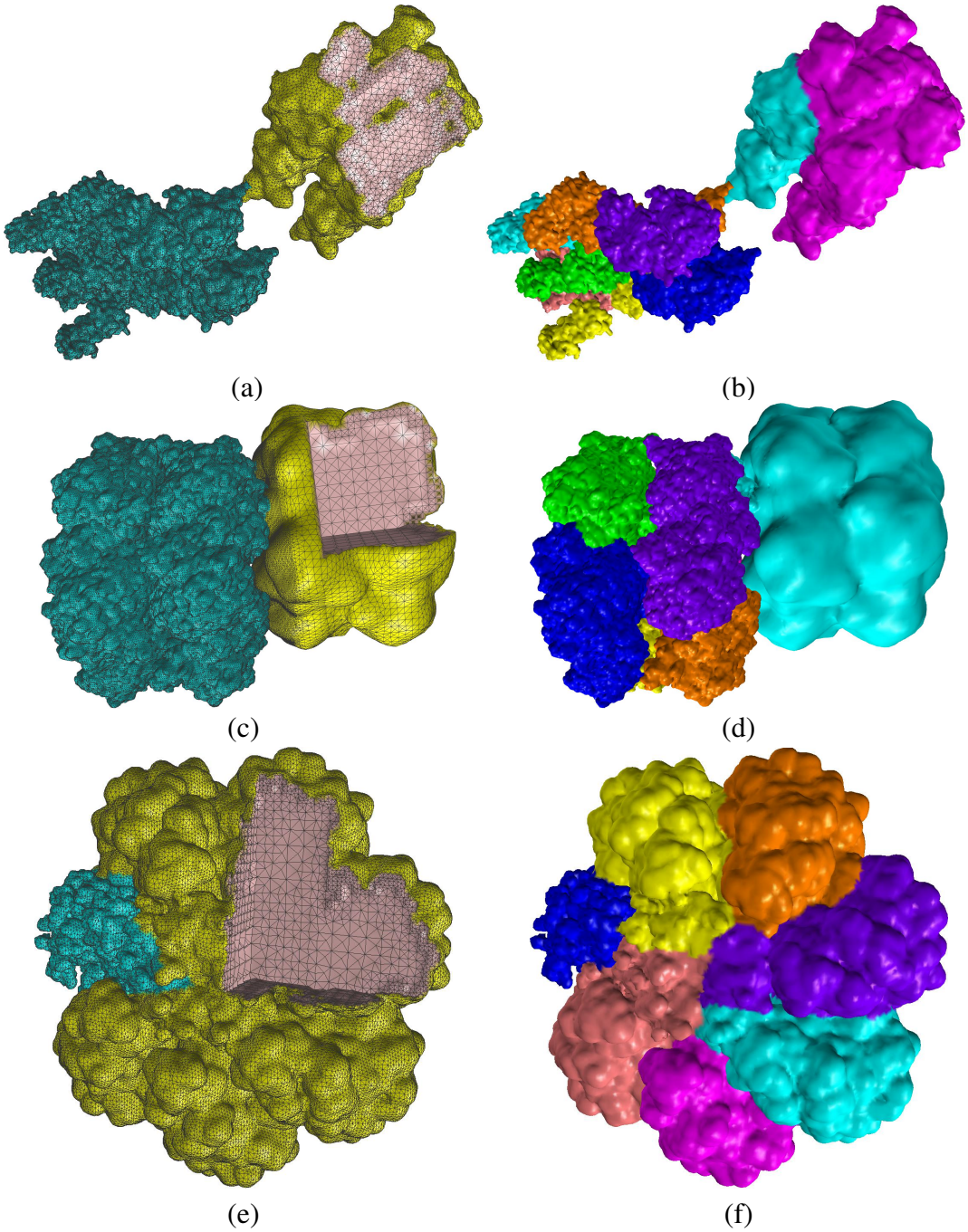


Figure 3.12: Three large proteins. (a-b) 2W4A with $P_R = 0.2$, $P_C = 0.8$ and $P_R = 0.4$, $P_C = 1.2$ for low and high resolution components, respectively; (c-d) 1HTQ with $P_R = 0.3$, $P_C = 0.8$ and $P_R = 1.0$, $P_C = 1.5$ for low and high resolution components, respectively; and (e-f) 2KU2 with $P_R = 0.5$, $P_C = 1.0$ and $P_R = 1.6$, $P_C = 2.0$ for low and high resolution components, respectively. (a), (c) and (e) show tetrahedral mesh of the proteins with the blue region emphasized in high resolution, and the pink region shows a cross section of the model; (b), (d) and (f) show the partition results in parallel computation.

Chapter 4

Structure-aligned Guidance Estimation in Surface Parameterization Using Eigenfunction-based Cross Field

Different types of meshes may be preferred for various studies of biomolecules. For given triangular biomolecular surfaces, quadrilateral meshes can be generated based on surface parameterization. In this chapter, we present a novel cross field-based parameterization method which aligns parametric lines to the main structures of objects, based on the eigenfunctions of Laplace-Beltrami operator. Compared with the existing methods based on principal curvatures, our method yields fewer singularities in the cross field. Besides, an anisotropy can be achieved in the parametric lines for a given scalar field.

4.1 Introduction

Surface parameterization is of great importance for many applications, such as quadrilateral meshing [12], texture mapping and synthesis [128, 66]. An important issue for surface parameterization is how to align parametric lines with the feature directions. Some simpli-

fication techniques [85, 95, 119] were developed to generate very coarse domain meshes with a good user control. Although feature alignment was achieved in a certain degree [85], it is difficult to control the simplification process to preserve surface features. Using the harmonic field [58, 122], features can be captured, but feature alignment is limited due to the difficulty in generating the field and placing singularities. In recent years, methods based on the cross field have been introduced [58, 13, 51, 96, 99]. Generally, the captured features in the cross field are represented by the principal curvatures, which are sensitive to the local detailed features and may fail in capturing structural features of an object at desired scales.

Eigenfunctions of the Laplace-Beltrami operator (LBO) are well-known for their property of capturing the shape behavior and structural feature of an object [65, 102, 104, 113]. Various eigenfunctions reflect structural features at different scales, which has been utilized in surface segmentation and reconstruction [102, 112]. The eigenfunctions vary along the object surface and are invariant to different poses, which makes them ideal for describing the structural feature of the object. A variety of applications have been introduced taking the advantages of eigenfunctions, such as pose-invariant Reeb graph [113], shape matching [65] or registration [86], and the Shape-DNA [104]. Another important application of eigenfunctions is surface quadrangulation or parameterizaton [150]. For example, the Morse-Smale complexes [26, 45, 76] were built by connecting the saddle and extrema of eigenfunctions, dividing the surface into several coarse quadrilateral patches. Despite of these developments, feature alignment is still a challenging problem in surface parameterization.

In this chapter, we introduce a novel method to define a guidance for cross field generation using eigenfunctions, and generate a structure-aligned parameterization for the input triangle mesh. A guidance is first constructed using the gradient of multiple eigenfunctions of the LBO to capture the structural feature at various scales. Then a smooth cross field is

built following the guidance, based on which a surface parameterization is computed. The main contributions of our work include:

1. A novel structure-aligned approach is developed for surface parameterization using eigenfunctions, which is insensitive to local detailed surface features;

2. Multiscale structural features are captured using the gradient of multiple eigenfunctions as a guidance for cross field generation; and

3. A new algorithm is introduced to enable anisotropy in the parameterization by adapting the cross field to non-uniform parametric line spacings.

The remainder of this chapter is organized as follows. Sec. 4.2 describes eigenfunctions. Sec. 4.3 explains how to define the guidance for cross field construction using the gradient of multiple eigenfunctions. Sec. 4.4 discusses cross field construction and surface parameterization. Sec. 4.5 shows some results. Finally, Sec. 4.6 draws conclusions and points out future work.

4.2 Eigenfunctions

Given a G^2 smooth surface S, the eigenproblem is to find the eigenvalues λ and their corresponding eigenfunctions f defined on it, such that

$$-\Delta_S f = \lambda f, \tag{4.1}$$

where Δ_S is the LBO defined on surface S. Since $-\Delta_S$ is a symmetric and nonnegative operator, the eigenvalues of $-\Delta_S$ are real and nonnegative. Eigenfunctions of the LBO provide a set of convenient basis to describe the shape behavior or structural feature of an object. Let M be a triangulation of surface S, $\{\mathbf{x}\}_{i=1}^{n}$ be the vertex set of M. A given

discretization scheme [29, 100, 137] for the LBO can be represented as

$$\Delta_S f(\mathbf{x}_i) \approx \sum_{j \in N(i)} w_{ij} f(\mathbf{x}_j), \quad w_{ij} \in \mathbf{R}, \tag{4.2}$$

where $N(i)$ contains the 1-ring neighborhood of $\mathbf{x}_i$, and w_{ij} are the weights defined in different discretizations of the LBO. The eigenproblem becomes

$$-\sum_{j \in N(i)} w_{ij} f(\mathbf{x}_j) = \lambda f(\mathbf{x}_i) \tag{4.3}$$

or in matrix form,

$$-WF = \lambda F, \tag{4.4}$$

where $F = [f(\mathbf{x_1}), ..., f(\mathbf{x_n})]^T$ and W is the coefficient matrix defined by w_{ij}. Eq. 4.4 yields *n modes*, each corresponds to an eigenvalue and eigenfunction. Let λ_k and F_k ($k = 0, 1, ..., n-1$) be the k^{th} eigenvalue and the corresponding eigenfunction, we have

$$0 = \lambda_0 \leq \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{n-1}. \tag{4.5}$$

$\lambda_0 = 0$ represents a rigid-body mode, its eigenfunction F_0 is a constant-scalar field. All the eigenfunctions used in this chapter are normalized such that $\sum_{i=1}^{n} [f(\mathbf{x}_i)]^2 = 1$.

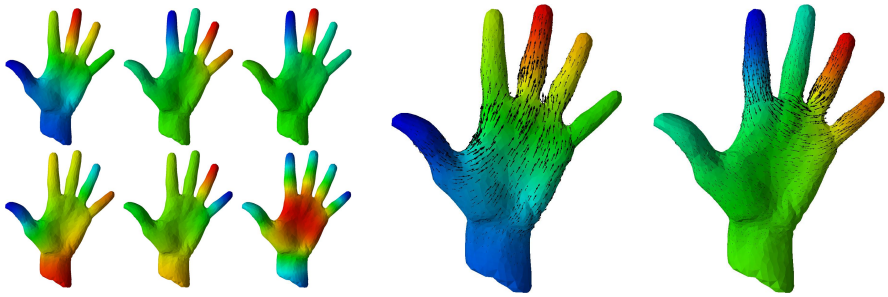
Various methods have been developed for the discretization of LBO. In this chapter, we use the cotangent scheme [65, 25, 67, 88, 97]. This discretization provides a symmetric matrix, which makes all the resulting eigenvalues and eigenfunctions real. However, the cotangent scheme was proved to be not convergent for irregular nodes, and it can not deal with non-uniform meshes well [137]. There are some research conducted on convergent discretization of the LBO. For example, a k-nearest neighbor of a vertex was considered using a truncated heat kernel [9]. In [137], the Laplace matrix was constructed based on a quadratic fitting and its convergence rate was proved to be linear [138]. This discretization

provides a non-symmetric matrix, resulting in complex eigenvalues and eigenfunctions. In this chapter we use the cotangent scheme LBO to obtain real eigenfunctions.

4.3 Guidance Estimation Using Eigenfunctions

Different eigenfunctions reflect surface features at different scales [102]. Compared with the high-mode eigenfunctions used in [26, 45, 76], the low-mode eigenfunctions are less sensitive to the detailed surface features and capture the major structure of the object. In this chapter, we will use multiple low-mode eigenfunctions to design a direction guidance and then build a cross field, from which we can obtain a structure-aligned surface parameterization.

The gradient of the eigenfunctions can be used to represent structural features. For example in the Hand model in Fig. 4.1, the gradient of the first and second eigenfunctions (black arrows) reflects the slim cylindrical structure of the fingers. However, a single eigenfunction may only reflect features in certain regions well. For example in (b), the gradient of Mode 1 eigenfunction follows the middle finger and the thumb very well, but not the little finger because the gradient magnitude is very small on it. Similarly in (c), the gradient of Mode 2 eigenfunction follows the index, third and little fingers well but not the thumb. From Fig. 4.1, we can observe that each eigenfunction plays a dominant role in certain regions, where the gradient of this eigenfunction reflects the structural features at a certain scale. We call such region a *feature region* of that eigenfunction. By combining the gradient in the feature regions from multiple eigenfunctions, we can build a structure-aligned guidance for the cross field construction. For example, we can define the middle finger and the thumb as the feature region of Mode 1 and the index, third and little finger as the feature region of Mode 2. Then the slim cylindrical structure of all five fingers can be captured using these two modes.



(a) (b) Mode 1 (c) Mode 2

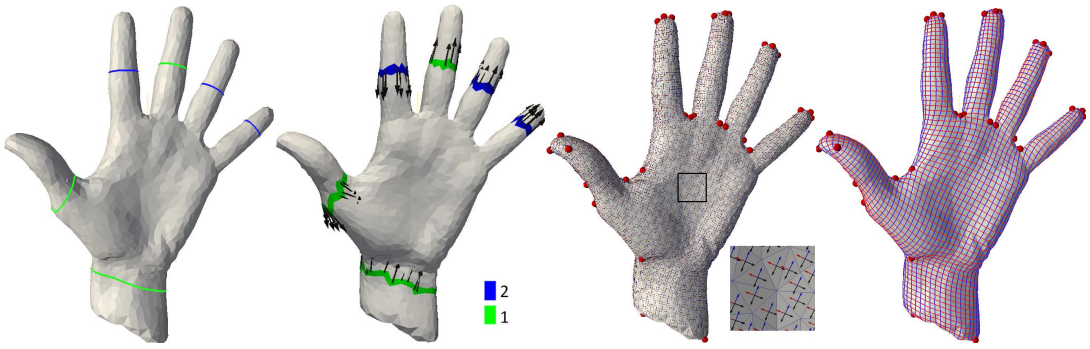
Figure 4.1: Eigenfunctions of the Hand model. (a) The first six eigenfunctions; and (b, c) the gradient (black arrows) distribution for the first and second eigenfunctions of the Hand model. The length of the arrows represents the gradient magnitude.

Then, the next problem is how to represent the feature region for each eigenfunction. Here, we design two different ways to represent the feature regions: isocontours and characteristic values.

4.3.1 Isocontours

Isocontours of eigenfunction f can locate important structural features, and they are often used for shape identification and segmentation [65, 102]. For each eigenfunction, we first generate several representative isocontours, and then form the *feature bands* using the intersecting triangles to represent the feature region of that mode. For example in Fig. 4.2(a), the isocontours from Mode 1 (green curves) and Mode 2 (blue curves) are generated. The green and blue bands in Fig. 4.2(b) represent the feature bands of the two eigenfunctions. The guidance for the cross field is defined as the gradient directions on triangles in the feature band, which we call the *guidance directions*. These triangles are called the *guidance triangles*. The guidance direction in each feature band follows the gradient of the eigenfunctions (black arrows).

To combine different modes using isocontours, we need to provide isovalues. One common choice is zero-isovalue because the corresponding isocontour represents the static region of a standing wave on the surface [65]. The gradient at the zero-isocontour of some eigenfunctions may have a very small magnitude, and thus a large numerical error may



(a) (b) (c) (d)

Figure 4.2: Process to generate a surface parameterizations using isocontours. (a) Isocontours from Modes 1 (green) and 2 (blue); (b) the corresponding feature bands (green and blue) with the defined guidance directions (black arrows); (c) the built smooth cross field, in which the singularities are marked in red nodes; and (d) the resulting parametric lines.

happen in the gradient calculation. The region with the extreme value is not a good choice either due to the same reason. For example in Fig. 4.3, the gradient magnitude is very small in Region A (the black zero-isocontour) and Region B (the red area with the maximum eigenfunction value), and the gradient directions vary intensively due to the large numerical error. The gradient usually follows the structure well for an isocontour with an isovalue away from the extreme and zero eigenfunction values, such as the orange area (Region C) surrounding the white curve. To choose proper isovalues automatically, we define

$$\alpha_l = f_{min} + r(f_{max} - f_{min}) \tag{4.6}$$

and

$$\alpha_u = f_{max} - r(f_{max} - f_{min}), \tag{4.7}$$

where $r \in (0, 0.5)$ is the parameter adjusting the location of the isocontours to define a good direction guidance, and f_{min} and f_{max} are the minimal and maximal eigenfunction values. Here we choose $r = 0.125$.

Generally if the guidance directions are very different in the adjacent triangles, the smoothness of the cross field will be affected. This may happen when the adjacent guidance triangles are from feature bands of different eigenfunctions. For example in Fig. 4.4, the

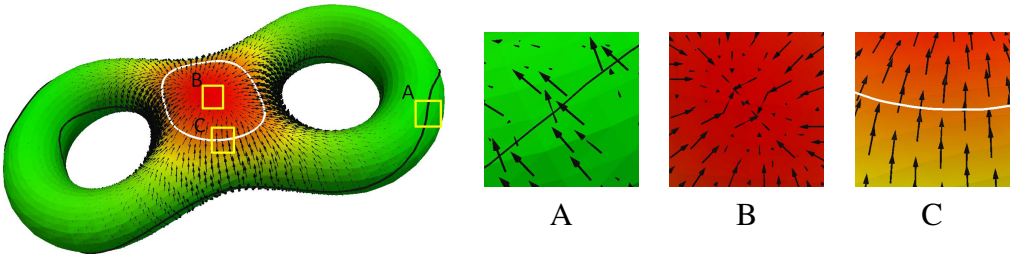


Figure 4.3: The gradient for the 7^{th} eigenfunction of the Eight model. The black and white curves represent the isocontours with the isovalue of zero and $f_{max} - 0.125(f_{max} - f_{min})$, respectively. The normalized gradient for Regions A, B and C is shown in the zoom-in pictures.

blue and red isocontours on the wrist intersect with each other, defining very different guidance directions for the triangles around the intersection. In this chapter, we define the guidance directions in adjacent triangles as non-consistent if these triangles are from feature bands of different eigenfunctions. Such a non-consistent situation can be avoided by adjusting the r value in Eqs. 4.6-4.7 to separate isocontours from different modes.

Given certain modes, although the isocontours can be adjusted easily by modifying the isovalues, it is difficult to avoid non-consistent situations automatically. To resolve this problem, we need to leave enough space between guidance triangles in different feature bands. In the following, we will talk about another method to represent the feature region, which can automatically separate guidance triangles in different feature bands.

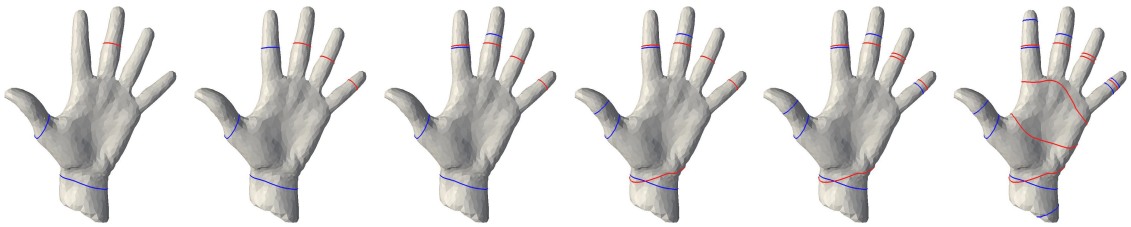


Figure 4.4: The isocontours collected from various modes. The blue and red lines correspond to α_l and α_u, respectively.

4.3.2 Characteristic Value

Instead of using isocontours, another way to represent the feature region is to divide the surface into patches, and each patch represents a feature region. Here, we define a positive characteristic value $C_{i,k}$ for each triangle T_i of the k^{th} mode. For a set of mode indices $\mathbf{K}$, triangle T_i is assigned to the k^{th} mode if

$$k = \arg\max_{j \in \mathbf{K}} C_{i,j}. \tag{4.8}$$

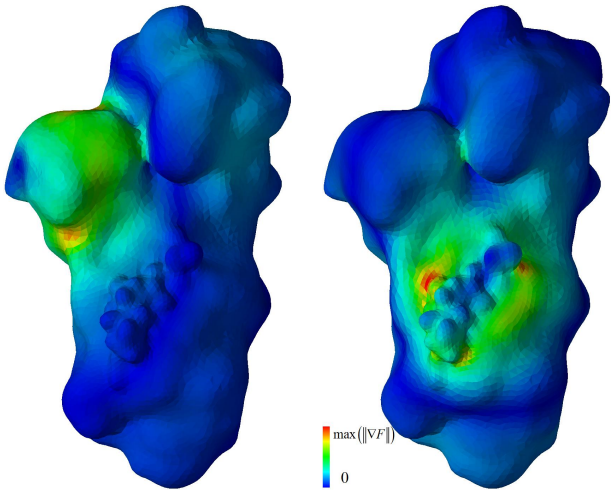
A *feature patch* is formed by the triangles assigned to the same mode.

In this chapter, we use the gradient magnitude of the eigenfunctions as the characteristic value. An eigenfunction usually has a large gradient magnitude at certain regions with special structural features. For example in Fig. 4.5, Modes 2 and 3 of the 4KYT model have a large gradient magnitude around two chains, while almost zero gradient everywhere else. Therefore, we define the characteristic value as $C_{i,j} = \| \nabla F_{i,j} \|$, where

$$\begin{aligned}\nabla F_{i,j} = \frac{1}{4A_i^2}\{&F_{i,j}(\mathbf{x}_a)\left[\gamma(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c) + \gamma(\mathbf{x}_a, \mathbf{x}_c, \mathbf{x}_b)\right] \\ &+ F_{i,j}(\mathbf{x}_b)\left[\gamma(\mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_a) + \gamma(\mathbf{x}_b, \mathbf{x}_a, \mathbf{x}_c)\right] \\ &+ F_{i,j}(\mathbf{x}_c)\left[\gamma(\mathbf{x}_c, \mathbf{x}_a, \mathbf{x}_b) + \gamma(\mathbf{x}_c, \mathbf{x}_b, \mathbf{x}_a)\right]\},\end{aligned}$$

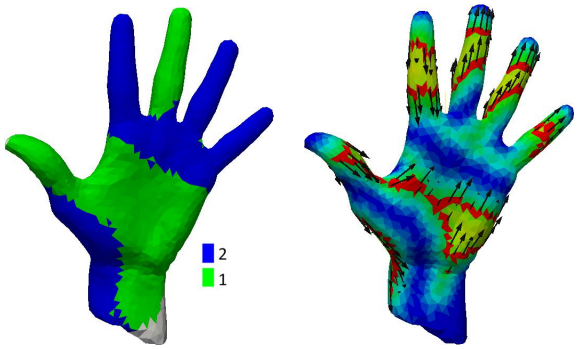
A_i denotes the area of T_i and $\gamma(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c) = \langle \mathbf{x}_a - \mathbf{x}_b, \mathbf{x}_b - \mathbf{x}_c \rangle (\mathbf{x}_c - \mathbf{x}_a) \in \mathbf{R}^3$.

Fig. 4.6(a) shows the obtained feature patches from Modes 1-2 using the characteristic value. In each feature patch, the Dijkstra distance d from the center of each triangle to the patch boundary is computed, which is shown in Fig. 4.6(b). Suppose the maximum distance in Patch k is D_k, we choose triangles with distance $0.45D_k < d < 0.55D_k$ as the guidance triangles (red triangles inside each feature patch, they form a feature band). The guidance directions are the gradient of the k^{th} eigenfunctions (black arrows). In this method, triangles



Mode 2 Mode 3

Figure 4.5: The gradient magnitude of Modes 2 and 3 eigenfunctions for the 4KYT model.

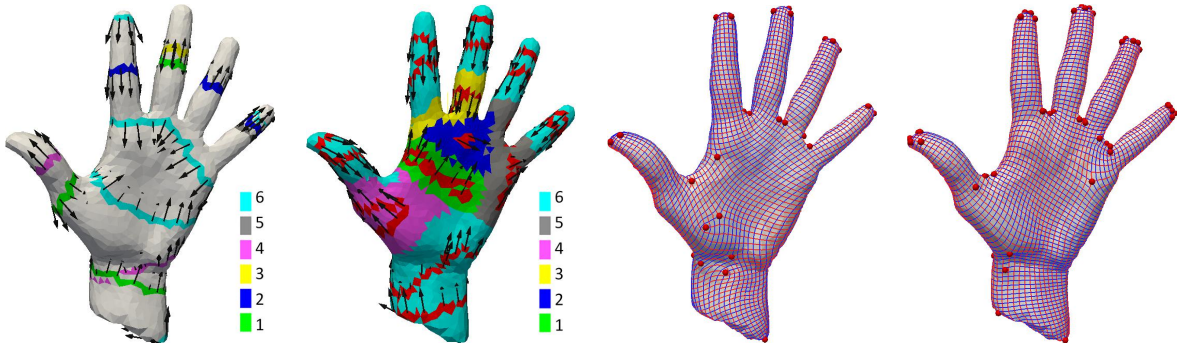


(a) (b)

Figure 4.6: Process of defining the guidance directions for the Hand model using the characteristic value. (a) The feature patches for Modes 1 (green) and 2 (blue); and (b) the Dijkstra distance distribution on the surface with guidance triangles (red triangles) and guidance directions (black arrows).

with a small gradient magnitude ($\leq 10^{-6}$) are ignored to avoid large numerical error in the gradient directions, see the grey regions in Fig. 4.6(a).

Discussion 3.1. Compared with the feature bands defined by the isocontours, feature bands caused by different modes are separated using the characteristic value and thus non-consistent guidance directions are avoided automatically. For example in Fig. 4.7(a), when the first six modes are used, non-consistent directions happen on the wrist due to the intersecting isocontours and feature bands (green and magenta bands) from Modes 1 and 4.



(a) (b) (c) (d)

Figure 4.7: Hand model. (a, b) The generated guidance directions using the first six modes; and (c, d) the resulting surface parameterization. Two ways are used to represent feature regions: (a, c) the isocontours; and (b, d) the characteristic values.

However in Fig. 4.7(b), there is no intersecting feature band because each feature band lies inside a feature patch and is also away from the patch boundary.

4.4 Cross Field Generation and Surface Parameterization

Using the guidance created from eigenfunctions, in this section we build a cross field via a smoothing process, and generate a surface parameterization following the mixed integer method [13]. Similar to [51, 93], we also enable a tradeoff control between the guidance alignment and the field smoothness. Differently, we further introduce a new algorithm to adapt the cross field to non-uniform parametric line spacings, enabling an anisotropy in the parameterization.

4.4.1 Overview of Cross Field-based Parameterization

Let $\mathbf{T}^g$ be the set of guidance triangles, and $\mathbf{T}^f$ be the set of free triangles on the surface. A smooth cross field is constructed based on the guidance directions. The guidance direction and the cross field in triangle T_i can be represented as $(\theta_i, \mathbf{e}_i)$ and $(\gamma_i, \mathbf{e}_i)$, where θ_i and γ_i are defined as a module of $\pi/2$, and $\mathbf{e}_i$ is the reference edge in triangle T_i. Similar to [51], our smoothness energy of the cross field on M is defined as

$$\Gamma^s = \sum_{e_{ij} \in E} (\theta_i + \kappa_{ij} + \frac{\pi}{2} p_{ij} - \theta_j)^2 + \bar{\lambda} \sum_{T_i \in \mathbf{T}^g} (\theta_i - \gamma_i)^2, \tag{4.9}$$

where e_{ij} is the edge shared by triangle T_i and T_j, E is the set of all the edges in the mesh, κ_{ij} is the angle between the reference edges of triangle T_i and T_j, and p_{ij} is the integer valued period jump of the cross field across e_{ij}. A parameter $\bar{\lambda}$ is used to control the tradeoff between the alignment to the guidance directions and the field smoothness. The smooth cross field is obtained by minimizing the energy function. We solve the minimization problem using the mixed-integer solver introduced in [14].

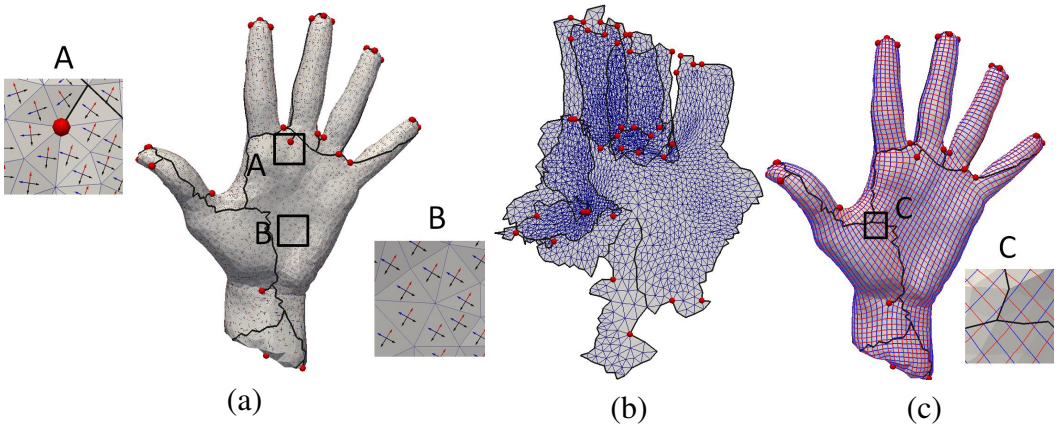


Figure 4.8: Process of surface parameterization for Fig. 4.6(b). (a) The smooth cross field (four arrows in each triangle) and singularities (red nodes); (b) disk-like planar region from the original surface; and (c) parametric lines.

Fig. 4.8(a) shows the built cross field using the guidance defined in Fig. 4.6(b). Red dots are the singularities of the field. The surface is cut into a disk-like planar region and all the singularities should be on the boundary of the planar region [13], see Fig. 4.8(b). In the disk-like planar region, the parametric coordinates (u, v) can be taken as two scalar fields, and two directions $\mathbf{u}_i$ and $\mathbf{v}_i$ in each triangle T_i are defined as the gradient of these two scalar fields, see blue and red arrows in Region A (singular) and Region B (regular) in Fig. 4.8(a). $\mathbf{u}_i$ is chosen as one of the four directions of the cross field in triangle T_i, which is forced to be consistent with the neighboring triangles. $\mathbf{v}_i$ is defined by rotating $\mathbf{u}_i$ by $\pi/2$ following the clockwise direction.

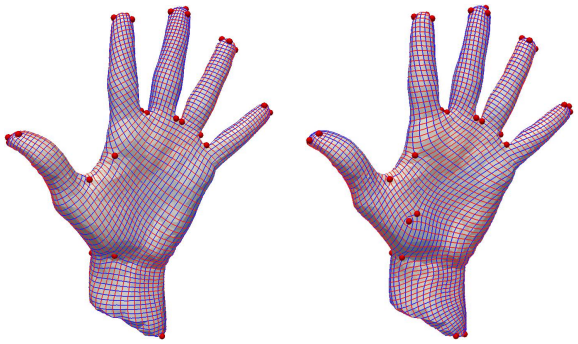
The parametric coordinates (u_i, v_i) of Vertex $\mathbf{x}_i$ are computed by minimizing an orientation energy [13]

$$\Gamma^o = \sum_{T_i \in M} A_i \cdot \left(\|h\nabla_T u_i - \mathbf{u}_i\|^2 + \|h\nabla_T v_i - \mathbf{v}_i\|^2 \right), \tag{4.10}$$

where h is a parameter controlling the spacing of the parametric coordinates, A_i is the area of triangle T_i, and $\nabla_T u_i$ and $\nabla_T v_i$ are two gradients. Integer constraints are set on the parametric coordinates for the other vertices on the planar region boundary to ensure that the integer-value parametric lines meet at the boundary. In this way, a quadrangulation of the surface can be generated directly from the parametric lines. For example in Fig. 4.8(c), the blue and red lines represent the integer-value parametric lines of u and v, respectively.

Discussion 4.1. As the cross field follows the guidance directions defined by eigenfunctions, the resulting parametric lines align with structural features automatically. As shown in Figs. 4.2(b), 4.6(b) and 4.7(a, b), the guidance directions are defined using the isocontours and the characteristic values. They all yield parametric lines aligned with the five fingers, see Figs. 4.2(d), 4.7(c, d) and 4.8(c). Compared with Fig. 4.7(b), Fig. 4.7(a) contains non-consistent guidance directions generated using the isocontours. Generally, consistent guidance directions yield good feature-aligned parameterization, see Fig. 4.7(d), while non-consistent guidance may introduce distortions. On the wrist of the hand in Fig. 4.7(c), there is a foldover in the parametric lines, where the resulting $\nabla_T u_i$ and $\nabla_T v_i$ are very different with $\mathbf{u}_i$ and $\mathbf{v}_i$. This can be improved by assigning larger weights for the triangles in this region when minimizing the orientation energy [13].

Discussion 4.2. Reducing the distortion is an important issue for surface parameterization. In previous literature [90, 91], the distortion was reduced by evolving a pre-defined metric on the surface. Differently, we reduce the distortion mainly by using the characteristic value to avoid non-consistent guidance directions and also modifying $\bar{\lambda}$ in Eq. 5.1 to balance the tradeoff between guidance alignment and field smoothness. A small $\bar{\lambda}$ value can



(a) (b)

Figure 4.9: A tradeoff control for the parameterization using the guidance directions in Fig. 4.7(c). (a) $\bar{\lambda} = 0.05$; and (b) $\bar{\lambda} = 5.0$.

relax the constraint and thus the distortion is reduced, although the feature alignment may be sacrificed at certain extent. Fig. 4.9 shows the parameterization results corresponding to different $\bar{\lambda}$ values. When a small $\bar{\lambda}$ is chosen, the parametric lines fail in aligning with the index finger, but the distortions in the wrist region are avoided (Fig. 4.9(a)). As $\bar{\lambda}$ becomes larger, the parametric lines align with the index finger better but distortions happen in the wrist region as shown in Fig. 4.9(b) and Fig. 4.7(c) ($\bar{\lambda} \to \infty$).

4.4.2 Anisotropic Surface Parameterization

While the alignment of the parametric lines are determined by the cross field, the spacing of them depends on the parameter h in Eq. 5.2. A smaller h value results in denser parametric lines which correspond to the integer value of the parametric coordinates u and v. Using a constant h, we obtain uniform parametric lines. While using a non-uniform h from a sizing field [11], the local spacing of the parametric lines can be controlled to achieve adaptive and anisotropic parameterization at certain extent. But the parametric lines with different spacings may not be compatible with each other. In this chapter we introduce a new method to modify the original cross field and make it adapt to the variation of the parametric line spacings.

First, we define a region that must follow the original direction as the *invariant region*, and regions between invariant regions as the *transition region*. In each iteration, the cross

field in the transition regions is modified according to the difference between the resulting $\nabla_T u_i$ and $\mathbf{u}_i$, or $\nabla_T v_i$ and $\mathbf{v}_i$. Given that the four directions of the cross field are perpendicular or parallel with each other, the modification of the cross field can be represented by the modification in one of its directions. If $\frac{|\nabla_T u_i \cdot \mathbf{u}_i|}{\|\nabla_T u_i\|} < \frac{|\nabla_T v_i \cdot \mathbf{v}_i|}{\|\nabla_T v_i\|}$,

$$\mathbf{u}_i = \mathbf{u}_i + r_c \left(\frac{\nabla_T u_i}{\| \nabla_T u_i \|} - \mathbf{u}_i \right), \tag{4.11}$$

else

$$\mathbf{v}_i = \mathbf{v}_i + r_c \left(\frac{\nabla_T v_i}{\| \nabla_T v_i \|} - \mathbf{v}_i \right), \tag{4.12}$$

where $r_c \in (0, 1)$ is a pre-defined parameter and here we choose $r_c = 0.2$.

Generally, a large gradient magnitude indicates an intensive varying of the field, so the resulting mesh should be dense along that gradient direction. Therefore, we set the anisotropy where $\| \nabla_T F_i^a \| > 0.5 * max(\| \nabla_T F^a \|)$ (the region pointed by the red arrow in Fig. 4.10(a)). Here, $\nabla_T F_i^a$ is the gradient of F^a in triangle T_i. The h values in u, v directions are defined as

$$\begin{aligned} h_{u,i} &= \left[\frac{|\mathbf{u}_i \cdot \nabla_T F_i^a|}{\| \nabla_T F_i^a \|} r_a + \left(1 - \frac{|\mathbf{u}_i \cdot \nabla_T F_i^a|}{\| \nabla_T F_i^a \|} \right) \right] h_0, \\ h_{v,i} &= \left[\frac{|\mathbf{v}_i \cdot \nabla_T F_i^a|}{\| \nabla_T F_i^a \|} r_a + \left(1 - \frac{|\mathbf{v}_i \cdot \nabla_T F_i^a|}{\| \nabla_T F_i^a \|} \right) \right] h_0, \end{aligned} \tag{4.13}$$

where r_a is a parameter controlling the ratio of line spacing in u, v directions, and h_0 defines the base size. We choose $r_a = 0.2$ in this chapter. Fig. 4.10 shows an example of anisotropic parameterization. (a) shows the input field F^a (Mode 1 eigenfunction). Using the original cross field in (c), the parameterization fails because the line spacing varies intensively, see (e). To adapt the cross field, a transition region is defined where $0.45 * max(\| \nabla_T F^a \|) < \| \nabla_T F_i^a \| < 0.5 * max(\| \nabla_T F^a \|)$, as shown in (b). After adapting the cross field, a valid parameterization is obtained with $r_c = 0.1$ and 40 iterations, as shown in (d) and (f). In the invariant region, the parametric lines keep aligned to the input field, while in the transition

region the alignment is sacrificed and more singularities are introduced. As a result in the region pointed by the red arrow in (a), the red parametric lines (perpendicular with the gradient direction) are much denser than the blue ones.

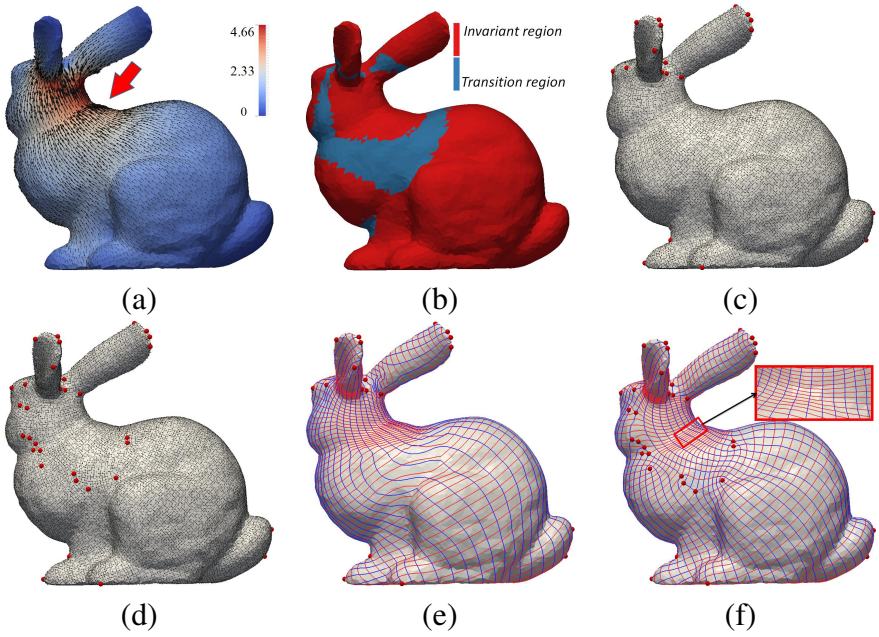


Figure 4.10: Anisotropic parameterization of the Bunny model using Mode 1 eigenfunction. (a) The gradient of the input field; (b) the invariant and transition regions; (c, e) the original cross field and the corresponding parameterization result; and (d, f) the adapted cross field and the corresponding parameterization result.

Discussion 4.3. In our anisotropic parameterization scheme, we define a novel transition region on the surface and adapt the cross field in this region to connect parametric lines with different line spacings smoothly. Although extra singularities may be yielded in the transition region, the orthogonality of the cross field is preserved during the entire adaptation process and the u, v parametric lines are automatically aligned to two perpendicular directions, resulting in quadrilateral elements with good quality. While the previous anisotropic parameterization methods [58, 39] seldom consider the quality of the output quadrilateral elements. For example in [58], the orthogonality was sacrificed to achieve the anisotropy, resulting in extremely small angles in the mesh.

4.5 Results and Discussion

Our surface parameterization algorithm has been applied to various models based on the guidance of eigenfunctions. The resulting parametric lines differ when different eigenfunctions are used. The results shown in this section were obtained using a computer with an Intel Xeo E5-1620 CPU and 16GB of memory.

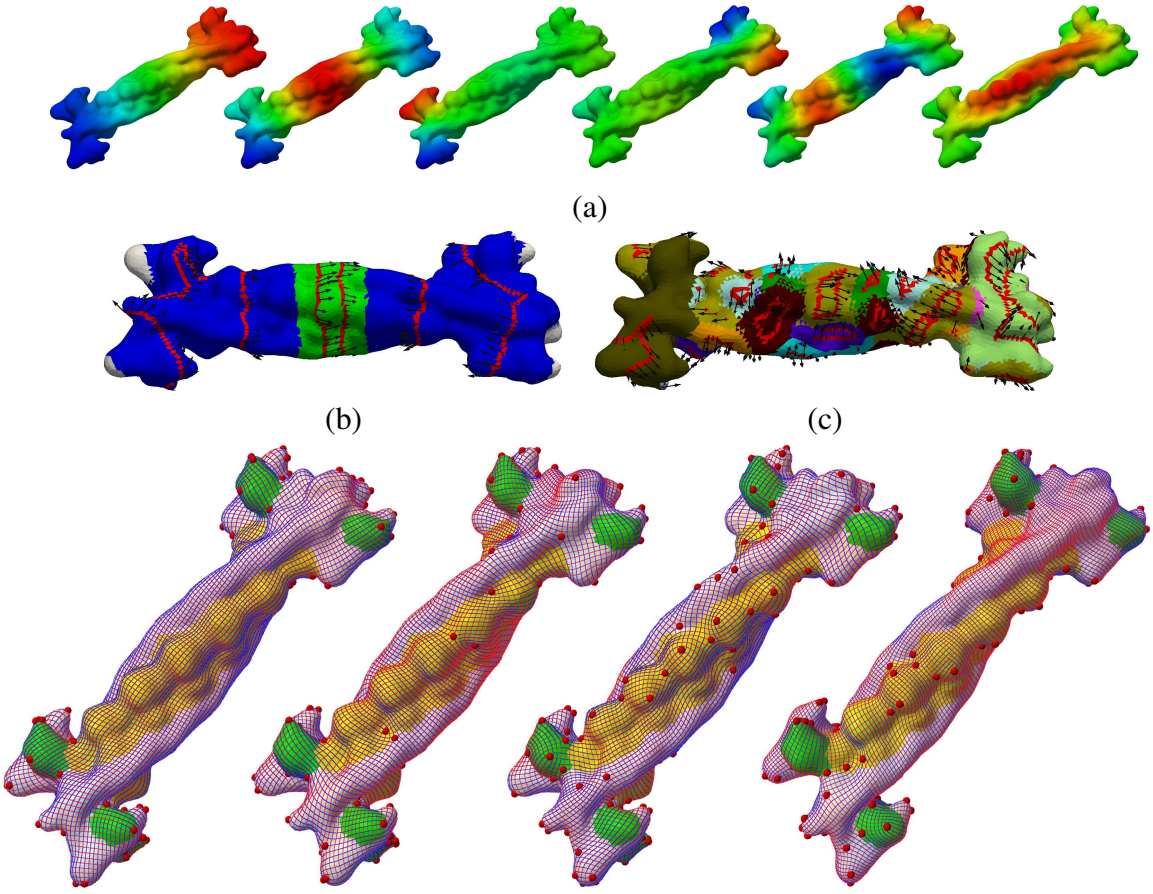
Using eigenfunctions to design guidance directions for cross field, multiscale structural features can be captured. Generally speaking, major structural features can be captured with a few low modes. For example, using only the first two modes, all the five fingers of the Hand model are captured in Fig. 4.8(c). Fig. 4.11 shows the parameterization results for the Thin Filament protein in the human heart muscle. The overall structure of the protein can be easily captured using the first two modes as shown in (d). When more modes are included in (e-f), more detailed features are captured on the actin (yellow region), which is an essential component for the cardiac contractile mechanism.

The principal curvatures have been applied in various applications to capture surface features at different scales [13, 51]. Here, we also compare our scheme with the principal curvature guided methods. The curvatures are estimated using a common technique introduced in [13, 51, 20]. Similarly, we measure the relative anisotropy of the principal curvatures using

$$\tau = \frac{||\kappa_{max}| - |\kappa_{min}||}{|\kappa_{max}| + |\kappa_{min}|},$$

and the principal directions in the region with $\tau > 0.8$ are defined as the guidance for the parameterization. Generally, both principal curvatures and eigenfunctions can be used to capture the structural feature of an object. Compared to principal curvatures, eigenfunctions especially low-mode ones are less sensitive to the local detailed features on the surface. For the Thin filament model in Fig. 4.11(d vs g), the trunk of the structure is full of sags and crests. Compared with the results from principal curvatures, the parametric lines from the eigenfunctions (Modes 1-2) are aligned to the axial direction of the object very

well, ignoring the detailed sags and crests. Therefore, our eigenfunction-based method could be a better choice when the users intend to exclude the influence of local detailed features and small changes on the surface.



(a)

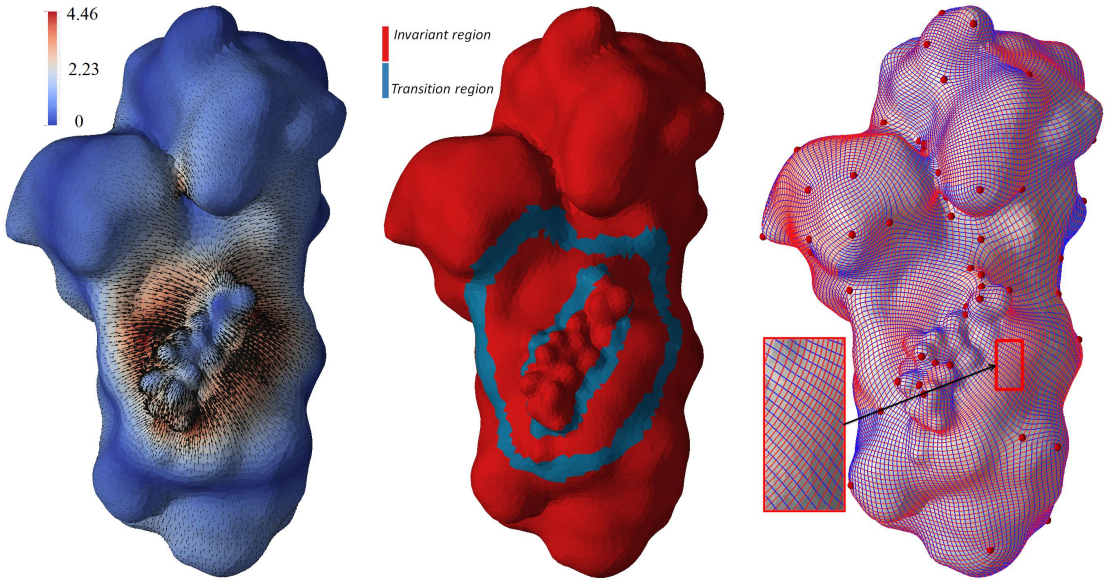
(b) (c)

(d) Modes 1-2 (e) Modes 1-14 (f) Modes 1-39 (g) Curvature

Figure 4.11: Surface parameterization results for the Thin Filament model guided by different numbers of eigenfunctions. (a) The first six modes; (b, c) guidance directions using Modes 1-2 and 1-14, respectively; (d-f) surface parameterization using Modes 1-2 ($N_S = 144$), 1-14 and 1-39, respectively; and (g) parameterization using the principal curvature directions ($N_S = 178$).

Besides combining different modes to capture the overall structure of the object, we can also construct anisotropic parameterization using the algorithm introduced in Sec. 4.4.2. For an eigenfunction, denser parametric lines along one certain direction can be generated in the region with large gradient magnitudes, achieving the anisotropy defined in Eq. 4.13. Fig. 4.12 shows an example of anisotropic parameterization for the Mode 3 eigenfunc-

tion of the 4KYT model. In the red region, the red parametric lines (perpendicular to the gradient direction) are much denser than the blue lines.



(a) (b) (c)

Figure 4.12: Anisotropic parameterization for 4YKT (d) The gradient magnitude and directions of Mode 3 eigenfunction; (e) invariant and transition regions; and (f) the resulting parametric lines from the adapted cross field.

Our algorithm also has some limitations. Generally, the parametric lines align well with the structural features using the guidance directions from eigenfunctions. However, there are two factors that may affect the alignment results. Firstly, as discussed in [11] the cross field generated by minimizing the smoothness energy (Eq. 5.1) may be affected by the input mesh quality, and the directions of the cross field may even fail to follow the guidance. Secondly, the parametric coordinates u and v are computed by minimizing the orientation energy (Eq. 5.2), and the resulting parametric lines are not guaranteed to align with the cross field due to the integer constraints. The anisotropic parameterization is also limited by the input mesh. As shown in Fig. 4.10(f), the anisotropy is achieved with more singularities introduced in the transition region. Generally, stronger anisotropy requires more singularities. As the singularities locate on the mesh vertices, the possible anisotropy is limited by the distribution of vertices in the input mesh. The number of singularities can be reduced if T-junctions are allowed in parameterization. Finally, as all the other

eigenfunction-based algorithms, we need users to provide which modes to be used, which is not fully-automatic. Although eigenfunctions of the LBO operator have a clear multi-scale behavior, they do not provide an immediate access to what a human would call different structural scales on an object, so it is still difficult to decide which combination of modes is reasonable.

4.6 Conclusion and Future Work

In this chapter, we have introduced a novel approach to define guidance directions for cross field-based surface parameterization. Two methods are designed to combine the gradient information from different eigenfunctions. Based on the guidance from multiple eigenfunctions, a cross field is built with a tradeoff between the guidance alignment and the field smoothness. As a result, the parametric lines are aligned with the structural features at multiple scales, also enabling an anisotropy by adapting the cross field to non-uniform parametric line spacings. In the future, we intend to continue working on anisotropic parameterization and also extend our structure-aligned surface parameterization to T-spline surface modeling.

Chapter 5

Adaptive and Anisotropic Quality T-mesh Generation for Multi-resolution Biomolecular Surfaces

Surface parameterization can also be used to generate the control meshes of T-spline, namely T-meshes, for the input biomolecular surfaces. In this chapter, a novel surface parameterization method is introduced, which adapts the parametric line spacing to multiple resolutions on biomolecular surfaces. Similar with the previous chapter, an anisotropy can also be achieved in the parametric lines. Based on the parameterization results, T-meshes are generated, yielding T-spline representations for biomolecular surfaces.

5.1 Introduction

In the context of isogeometric analysis [46, 8, 125, 49, 120], the T-spline surface provides a powerful basis for computation in different applications, which can also greatly benefit the analysis of biomolecules. Due to the high efficiency of the T-mesh, the multi-resolution feature on the biomolecular surface can be represented efficiently. Various

methods have been developed for T-mesh generation [130, 131, 89, 40, 69]. In recent years, the cross field-based global parameterization methods were introduced in surface quadrangulation [12], which capture surface features based on the principal curvature directions [58, 13, 51, 96, 99, 73] or eigenfunctions of the Laplace-Beltrami operator [74]. These techniques provide a nice basis for us to build T-meshes for biomolecules with multi-resolution features preserved efficiently.

In this chapter, an atom simplification method is developed to eliminate these atoms and improve the computational efficiency. Based on the simplified structure, a multi-resolution biomolecular surface can be built for quality T-mesh generation. The main contributions in this chapter include:

1. An extended cross field-based method is developed for adaptive and anisotropic parameterization, which adapts the parametric line spacings to multi-resolution surface features; and

2. A new gradient flow-based method is introduced for T-mesh quality improvement, preserving the anisotropy in quadrilateral elements robustly.

The remainder of this chapter is organized as follows. Section 5.2 explains the extended parameterization algorithm together with the new quality improvement method. Section 7.5 shows the results. Finally, Section 7.6 draws conclusions and points out the future work.

5.2 Multi-resolution T-mesh Construction

5.2.1 Multi-resolution Biomolecular Surface

For the analysis of large biomolecular complexes, usually only a specific component is essential for accuracy. A multi-resolution surface can be used to represent the biomolecular surface, maintaining high resolution details on the surface for the emphasized components while providing low resolution for the rest. In this chapter, the emphasized components are

defined as specific chains in the biomolecular complexes based on their biological functions. For example in Fig. 5.1(a), the chains belonging to actin (blue and orange) in the human Thin Filament protein should be emphasized because they are receptors for some inhibitors. For the emphasized chains, we keep all the atoms and also use larger coefficients (P_R, P_C) in Eq. (3.4) to obtain higher resolution; while for the rest of the biomolecule, we simplify atoms and choose smaller (P_R, P_C). Sometimes, sharp noises may happen in the Gaussian density map around the connection region of the lower and higher resolution surface, which can be removed by applying a low-pass filter based on the fast Fourier transform [31]. Then, the multi-resolution surface is extracted from the constructed Gaussian density map using the dual contouring method [147, 145], and adaptive triangular meshes are obtained. In the following, we will talk about how to construct surface parameterization and quality T-meshes using these triangular meshes.

5.2.2 Surface Parameterization and T-mesh Construction

In this chapter, we extend the cross field-based parameterization method [58, 13, 51, 96, 99, 74] to T-mesh generation for biomolecular surfaces. Firstly, a cross field is built on the triangle mesh guided by the principal curvature directions. Then an adaptive parameterization is computed based on the cross field. Anisotropy can also be achieved during surface parameterization. Finally, T-meshes are constructed by connecting the nodes with integer parametric coordinates.

Review of Cross Field-based Parameterization. A cross field is defined in each triangle T_i with four perpendicular vectors, which can be represented as an angle θ_i referring to an edge $\mathbf{e}_i$ of the triangle, namely the reference edge. These vectors can be initialized using different inputs, such as the principal curvature directions. Then the cross field is

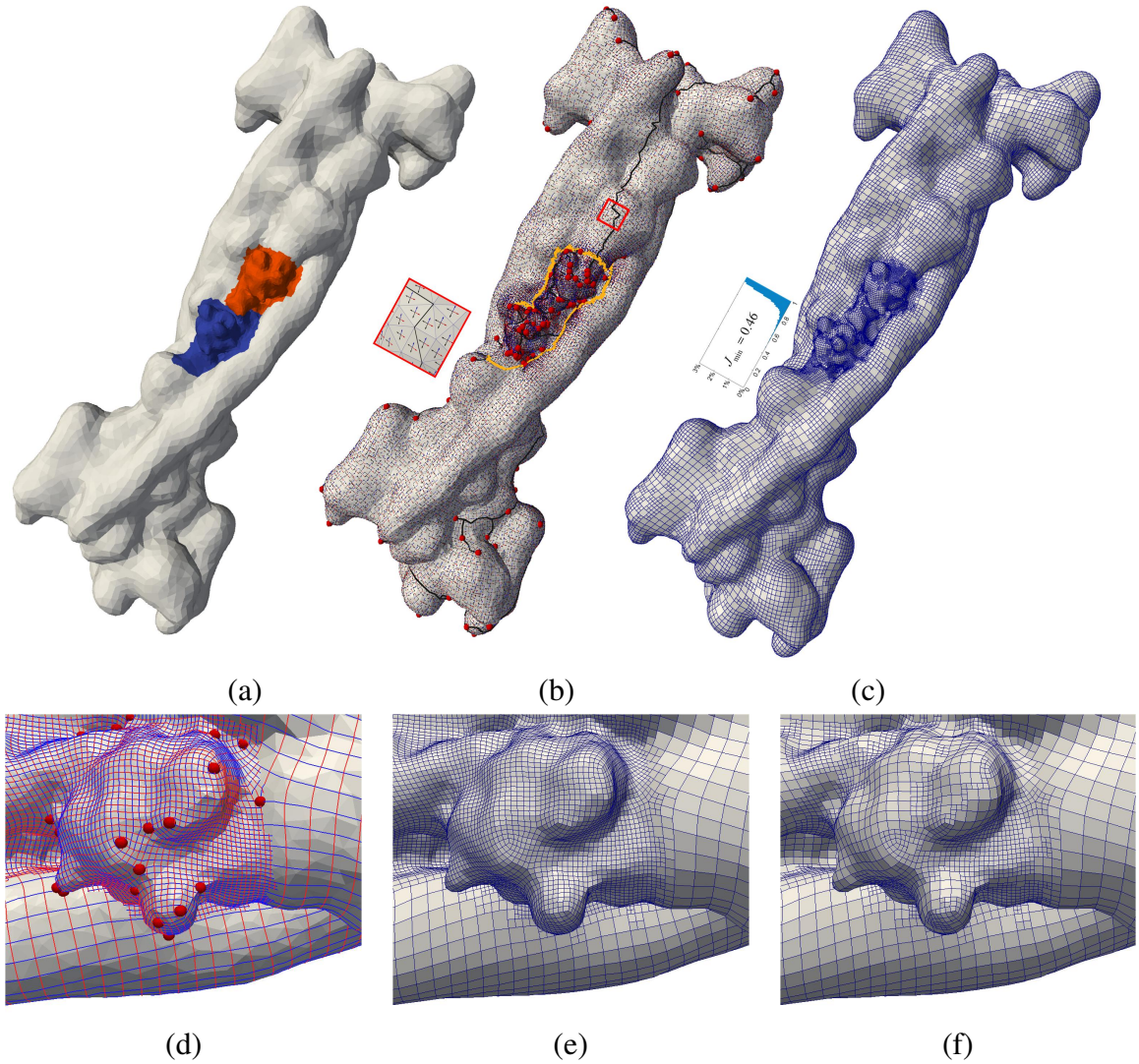


Figure 5.1: Adaptive parameterization for the multi-resolution surface of 2W4U. (a) The multi-resolution triangle surface; (b) the cross field; (c) the adaptive T-mesh; (d) the parametric lines; and (e, f) the T-meshes corresponding to (d) before and after the removal of redundant vertices.

smoothed by minimizing the smoothness energy [13]

$$\Gamma^S = \sum_{e_{ij} \in E} (\theta_i + \varphi_{ij} + \frac{\pi}{2} p_{ij} - \theta_j)^2, \tag{5.1}$$

where e_{ij} is the edge shared by triangles T_i and T_j, E_T is the set of edges in the mesh, φ_{ij} is the angle between the reference edges of triangle T_i and T_j, and p_{ij} is the integer valued period jump of the cross field across e_{ij}. By minimizing the smoothness energy, θ_i is updated

in each triangle T_i and a smooth cross field is obtained, see Fig. 5.1(b). Then, the surface is cut into a disk-like planar region with all the singularities (red dots) on its boundary (black lines). The parametric coordinates (u, v) of each vertex behave as two piece-wise linear scalar fields, which can be obtained by minimizing an orientation energy [13, 96, 74],

$$\Gamma^O = \sum_{i=1}^{N_T} A_i \left(\left\| h_i^u \nabla_{T_i} u - \mathbf{u}_i \right\|^2 + \left\| h_i^v \nabla_{T_i} v - \mathbf{v}_i \right\|^2 \right), \tag{5.2}$$

where N_T is the number of triangles, A_i is the area of triangle T_i, $\left(h_i^u, h_i^v\right)$ are parameters controlling the spacings of the parametric lines, $(\nabla_{T_i} u, \nabla_{T_i} v)$ are gradients of (u, v) in T_i, and $(\mathbf{u}_i, \mathbf{v}_i)$ are two perpendicular vectors chosen from the cross field (red and blue arrows in Fig. 5.1(b)). The parametric line spacings in u and v directions equal to $1/h_i^u$ and $1/h_i^v$, respectively. Integer constraints are set on the planar region boundary to ensure consistent parametric lines, enabling a valid quadrangulation of the surface.

Adaptive T-mesh Generation. For the multi-resolution surface in Fig. 5.1(a), we need to generate denser elements in the high resolution regions (blue and orange) to capture the detailed features. This can be achieved by adapting the line spacings in the orientation energy in Eq. (5.2): smaller (h_i^u, h_i^v) values are set for the higher resolution regions, while larger values are set for others. As shown in Fig. 5.2 when a higher resolution patch i (blue) is connected with a lower resolution patch j (yellow), we restrict $h_j = 2^m h_i$, where m is a positive integer. In addition, the parametric coordinates at the patch boundaries should satisfy

$$\begin{cases} (u_{A,i}, v_{A,i})^T = 2^m \mathbf{R}_{AB} \left(u_{A,j}, v_{A,j}\right)^T + 2^m (I_u, I_v)^T \\ (u_{B,i}, v_{B,i})^T = 2^m \mathbf{R}_{AB} \left(u_{B,j}, v_{B,j}\right)^T + 2^m (I_u, I_v)^T \end{cases}, \tag{5.3}$$

where $\mathbf{R}_{AB}$ is the rotation matrix across edge AB, (I_u, I_v) are integer-valued shift in the parametric coordinates, and m controls the difference of the parametric line spacing across the patch boundary. The parametric coordinates are computed by minimizing the orientation energy Γ^O with the constraints in Eq. (5.3). By connecting vertices with integer-valued

parametric coordinates, a quadrilateral mesh can be built with T-junctions on the edges. To ensure the strongly-balanced structure in the T-mesh ($m \leq 1$ for any two neighboring elements), some quadrilateral elements in the transition region need to be subdivided.

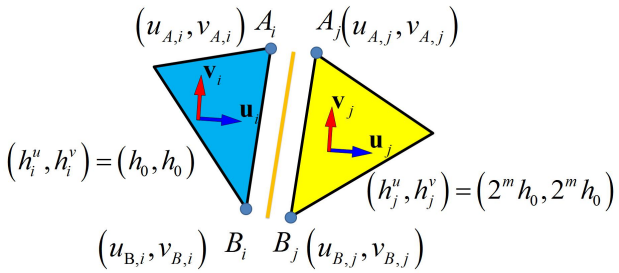


Figure 5.2: Two triangles across the patch boundary (orange).

Fig. 5.1(d-e) shows the adaptive parameterization result for the multi-resolution surface (a local region) of Thin filament (2W4U). High resolution is set for two emphasized chains (Chains 18 and 20) with denser elements. To have a smooth transition from a higher resolution to a lower one, some T-mesh elements need to be modified. Fig. 5.3 shows four different connections across the patch boundary (orange curve). Here $m = 2$, so some quadrilateral elements are subdivided to ensure a strongly-balanced structure. The obtained T-meshes may be too fine in some regions, therefore we identify redundant vertices based on their surface error and remove them from the T-meshes. Fig. 5.1(e-f) shows a comparison between the T-meshes before and after removing redundant vertices.

Anisotropic T-mesh Generation. Similar with [74], an anisotropy can be defined from an input scalar field f. For the parametric lines following the gradient direction ∇f, h_i is set to be an uniform value h^0. For the perpendicular direction to ∇f, the line spacing $h_i^{\perp}$ is determined by the gradient magnitude,

$$\frac{1}{h_i^{\perp}} = \alpha \cdot \frac{\max(\|\nabla f\|)}{h^0 \|\nabla f_i\|}, \tag{5.4}$$

where α controls the minimum line spacing. If $\|\nabla f_i\| < \alpha \cdot \max(\|\nabla f\|)$, the line spacing is set to be $h_i^{\perp} = h^0$. As such, the quadrilateral elements are stretched along the direction

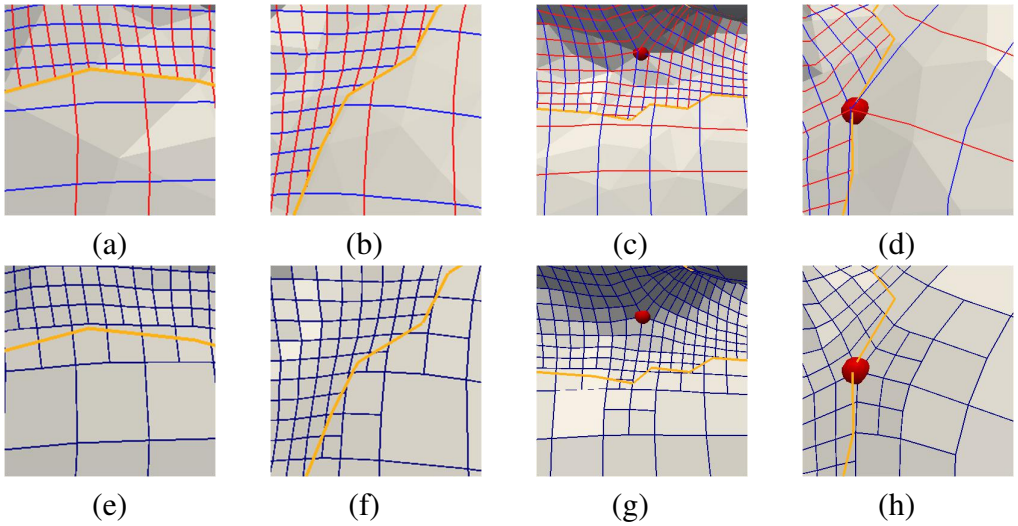


Figure 5.3: Various parametric lines across the patch boundary and their resulting T-meshes. (a-d) The parametric lines; and (e-h) the corresponding T-meshes.

perpendicular to the gradient, and dense elements are generated in the gradient direction. In Fig. 5.4, Mode 2 eigenfunction of the Laplace-Beltrami operator is used as the input scalar field. Similar with the isotropic parameterization in Fig. 5.1(d), we adapt the parametric line spacings to the surface resolution by quadrupling the line spacings across the patch boundary.

5.3 T-mesh Quality Improvement

For a Vertex $\mathbf{x}_i$ in Element j, the other vertices in the element can be represented as $\mathbf{x}^j_{i+1}$, $\mathbf{x}^j_{i+2}$ and $\mathbf{x}^j_{i+3}$ in the counter-clockwise order, see Fig. 5.5(a). The *scaled Jacobian* [143, 33] at $\mathbf{x}_i$ equals to $\det([\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3])$, where

$$\mathbf{v}_1 = \frac{\mathbf{x}^j_{i+1} - \mathbf{x}_i}{\left\|\mathbf{x}^j_{i+1} - \mathbf{x}_i\right\|}, \ \mathbf{v}_2 = \frac{\mathbf{x}^j_{i+3} - \mathbf{x}_i}{\left\|\mathbf{x}^j_{i+3} - \mathbf{x}_i\right\|}, \text{ and } \mathbf{v}_3 = \frac{\mathbf{v}_1 \times \mathbf{v}_2}{\|\mathbf{v}_1 \times \mathbf{v}_2\|}.$$

The quality of Element j can be measured by the minimum Jacobian at its four vertices. Generally, the overall quality of the T-mesh from surface parameterizaion is good in term of Jacobian, except a few elements. To improve the Jacobian of the T-mesh elements,

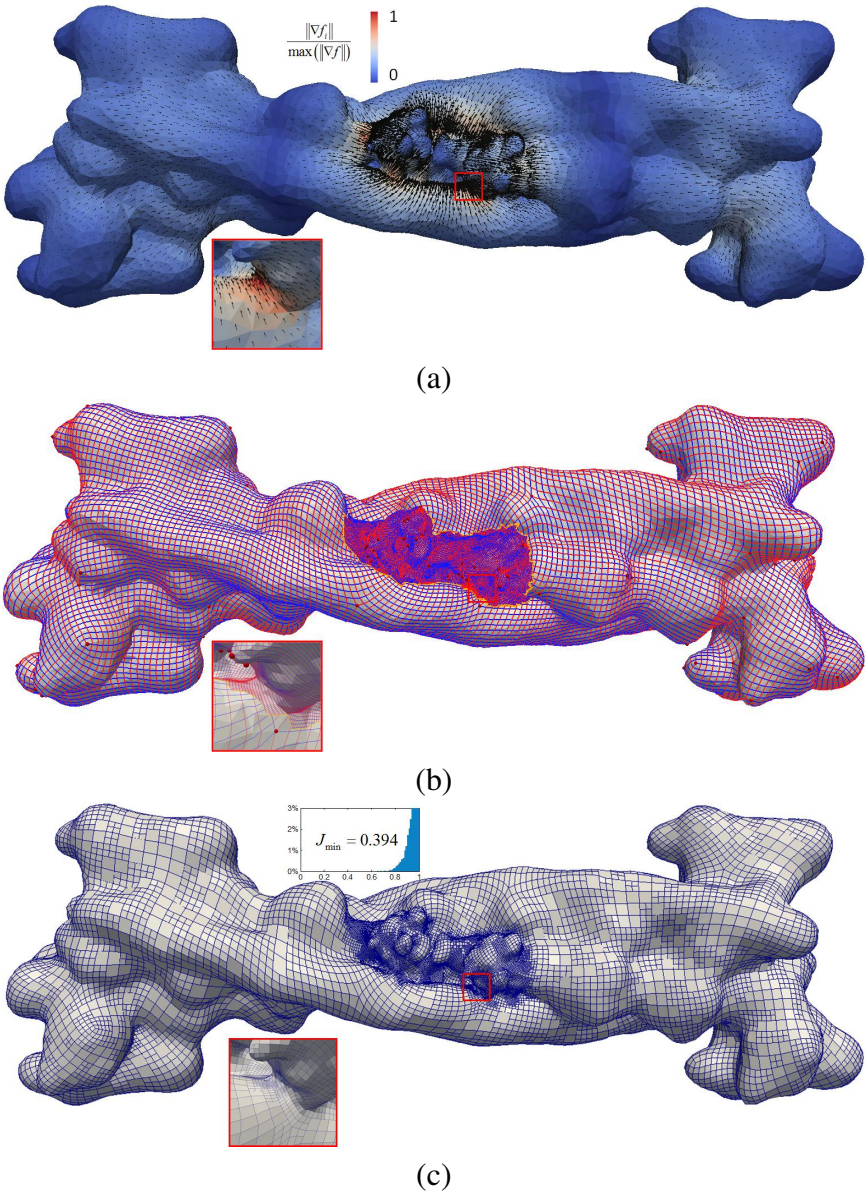


Figure 5.4: Anisotropic T-mesh generation from the Mode 2 eigenfunction of LBO ($\alpha = 0.1$). (a) The gradient and gradient magnitude of the input field; (b) the adaptive and anisotropic parameterization; and (c) the corresponding T-mesh.

we apply a new gradient flow-based quality improvement method, which considers both the original shape of elements and the orthogonality of edges. Contrast to other existing approaches [135, 15], our method can preserve the anisotropy of the T-mesh.

As shown in Fig. 5.5(b), for each T-junction we extend it to make the surrounding elements form a local unstructured quadrilateral mesh. Suppose Element j in Fig. 5.5(a) is

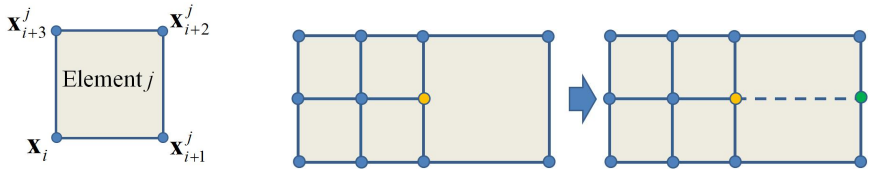


Figure 5.5: (a) A neighboring element j surrounding Vertex $\mathbf{x}_i$; and (b) extending the T-junction (orange) to form a local unstructured mesh.

a neighboring element surrounding Vertex $\mathbf{x}_i$, the *cross distortion* at $\mathbf{x}_i$ is defined as

$$\Gamma^C(\mathbf{x}_i) = \sum_{j \in Q_i} \frac{\frac{1}{a_j^2}\left\|\mathbf{x}^j_{i+1} - \mathbf{x}_i\right\|^2 + \frac{1}{b_j^2}\left\|\mathbf{x}^j_{i+3} - \mathbf{x}_i\right\|^2}{\frac{2}{a_j b_j}\left\|\left(\mathbf{x}^j_{i+1} - \mathbf{x}_i\right) \times \left(\mathbf{x}^j_{i+3} - \mathbf{x}_i\right)\right\|}, \tag{5.5}$$

where Q_i is the set of neighboring elements of Vertex $\mathbf{x}_i$, and

$$\left(a_j, b_j\right) = \left(\frac{\left\|\mathbf{x}^j_{i+1} - \mathbf{x}_i\right\| + \left\|\mathbf{x}^j_{i+2} - \mathbf{x}^j_{i+3}\right\|}{2}, \frac{\left\|\mathbf{x}^j_{i+3} - \mathbf{x}_i\right\| + \left\|\mathbf{x}^j_{i+2} - \mathbf{x}^j_{i+1}\right\|}{2} \right)$$

are the *feature lengths* of Element j. The total cross distortion energy of the T-mesh is defined as

$$\Gamma^C_M = \sum_{i=1}^{N} \Gamma^C(\mathbf{x}_i), \tag{5.6}$$

where N is the number of vertices. The T-mesh quality can be improved by minimizing Γ^C_M via a gradient flow method. Let $\mathbf{g} = (x_1, y_1, z_1, ..., x_N, y_N, z_N)^T$ be the vector containing the coordinates of all the vertices in the T-mesh. For the k^{th} step of the gradient flow, we have

$$\mathbf{g}^{k+1} = \mathbf{g}^k - \tau \nabla \Gamma^C_M, \tag{5.7}$$

where

$$\nabla \Gamma^C_M = \left(\frac{\partial \Gamma^C_M}{\partial x_1}, \frac{\partial \Gamma^C_M}{\partial y_1}, \frac{\partial \Gamma^C_M}{\partial z_1}, ..., \frac{\partial \Gamma^C_M}{\partial x_N}, \frac{\partial \Gamma^C_M}{\partial y_N}, \frac{\partial \Gamma^C_M}{\partial z_N} \right)^T.$$

Note that in each step, vertices can only move on the tangent plane. Therefore, we update $\mathbf{x}_i^k$ with $\mathbf{x}_i^{k+1} - \left(\mathbf{n}_i \cdot \mathbf{p}_i^k\right)\mathbf{n}_i$, where $\mathbf{n}_i$ is the surface normal and $\mathbf{p}_i^k = \mathbf{x}_i^{k+1} - \mathbf{x}_i^k$.

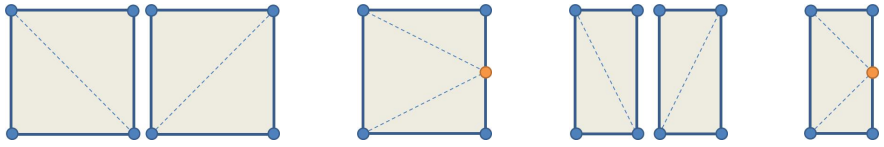
We also implemented an isotropic T-mesh quality improvement method based on the triangle optimization [135, 15] and compared our method with it. For an isotropic T-mesh, the ideal shapes of the elements are squares or rectangles, as shown in Fig. 5.6, which can be decomposed into several *ideal triangles*. For a general quadrilateral element, we decompose it into several *real triangles* following the same splitting format. An affine mapping between a real triangle j and its corresponding ideal triangle j_I can be defined as

$$\begin{aligned} \mathbf{f}_\mathbf{K} : j_I &\rightarrow j \\ \tilde{\mathbf{x}} &\mapsto \mathbf{x} = \mathbf{K}_j\tilde{\mathbf{x}} + \mathbf{v}, \end{aligned} \tag{5.8}$$

where $\mathbf{x}$ is a vertex in the real triangle, $\tilde{\mathbf{x}}$ is the corresponding vertex in the ideal triangle, and $\mathbf{v}$ is a constant term. The distortion of the T-mesh is defined as

$$\Gamma_M^T = \sum_{i=1}^{N_Q} \sum_{j \in R_i} \frac{\left\|K_j\right\|_F^2}{\det\left(\mathbf{K}_j\right) + \sqrt{\left(\det\left(\mathbf{K}_j\right)\right)^2 + 4\delta^2}}, \tag{5.9}$$

where N_Q is the number of quadrilateral elements, R_i is the set of real triangles generated from Element i, $\left\|K_j\right\|_F = \sqrt{\mathrm{tr}\left({\mathbf{K}_j}^T\mathbf{K}_j\right)}$ is the Frobenius norm of $\mathbf{K}_j$, and δ is an arbitrary small value (e.g. $\delta = 0.1$). Similar with Eq. (5.7), a gradient flow method can be applied to minimize Γ_M^T.



(a) (b) (c) (d)

Figure 5.6: Ideal triangles from a square (a-b) or rectangle (c-d). (a, c) No T-junction; and (b, d) with a T-junction. The orange dots are T-junctions.

Fig. 5.7 shows a comparison between our method and the triangle optimization method. We can observe that both methods can improve the mesh quality. As shown in (b-c), for the

isotropic elements in Region B, both methods yield similar results. But for the anisotropic elements in Region A, only our method preserves the anisotropic rectangle shape of the elements.

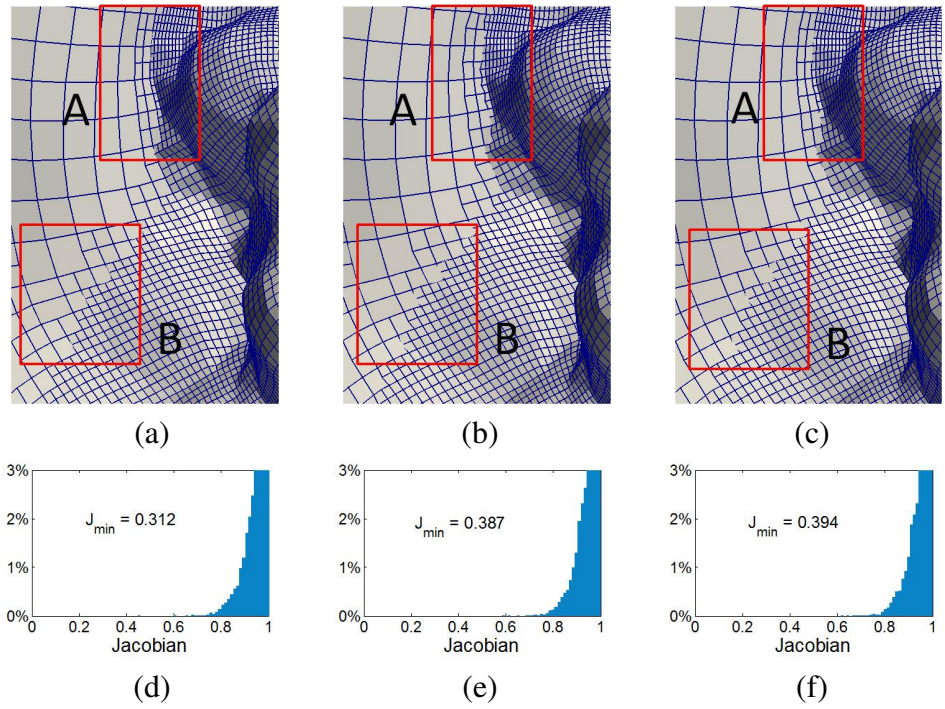


Figure 5.7: Quality improvement results for the T-mesh in Fig. 5.4(c) using two different methods. (a) The original T-mesh; (b-c) the improved T-meshes using the triangle optimization method and our method, respectively; and (d-f) the Jacobian distribution corresponding to (a-c).

5.4 Results and Discussion

In this section, the biomolecule simplification and multi-resolution T-mesh generation algorithms are applied to various biomolecular complexes. All the results are generated from a computer with an Intel Xeon E5-1620 CPU, a Nvidia GeForce GTX680 graphic card, and 16GB of memory.

Figs. 5.1, 5.8 and 5.9 show adaptive T-mesh generation results for the five proteins in Tab. 3.4, which can be used directly as the control mesh to build rational T-spline surfaces [131]. The principal curvatures are used to guide the parametric line directions, there-

fore the obtained T-meshes follow the local surface features. On the contrary in Fig. 5.10, eigenfunctions are used to define the input vector field. Therefore, the obtained anisotropic T-mesh follows the gradient direction of those eigenmodes. The variation of different modes is reflected by both the element orientations and the stretched shapes. We can also set various resolutions for different emphasized components. In Fig. 5.11, the parametric line spacings are $1 : 2 : 4$ for Chain 20 (orange), Chain 18 (blue) and the rest of the surface. Tab. 5.1 shows statistics of all the T-mesh generation results. We can observe that the number of T-junctions and the number of singularities vary for different proteins due to their complex surface features. In addition, the generated T-meshes are in good quality with $J_{\min} \geq 0.37$.

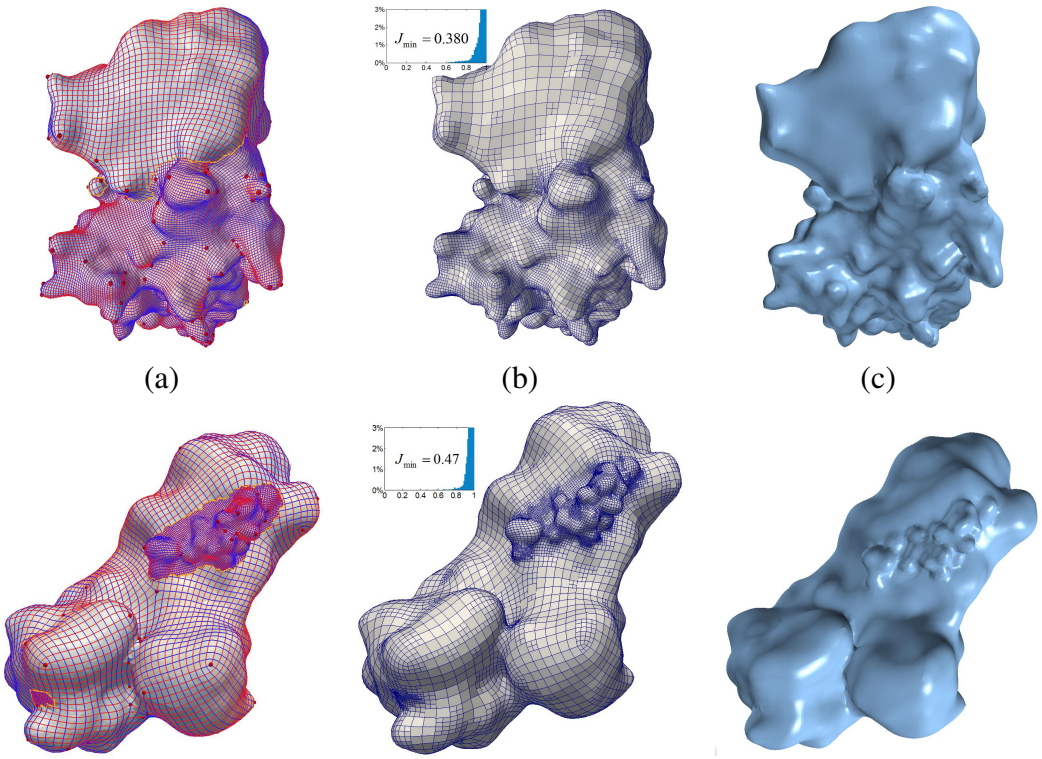


Figure 5.8: Multi-resolution surfaces for 2O53 and 4KYT. (a-c) 2O53; and (d-f) 4KYT. Left column: adaptive parameterization; Middle column: T-mesh; Right column: T-spline surface.

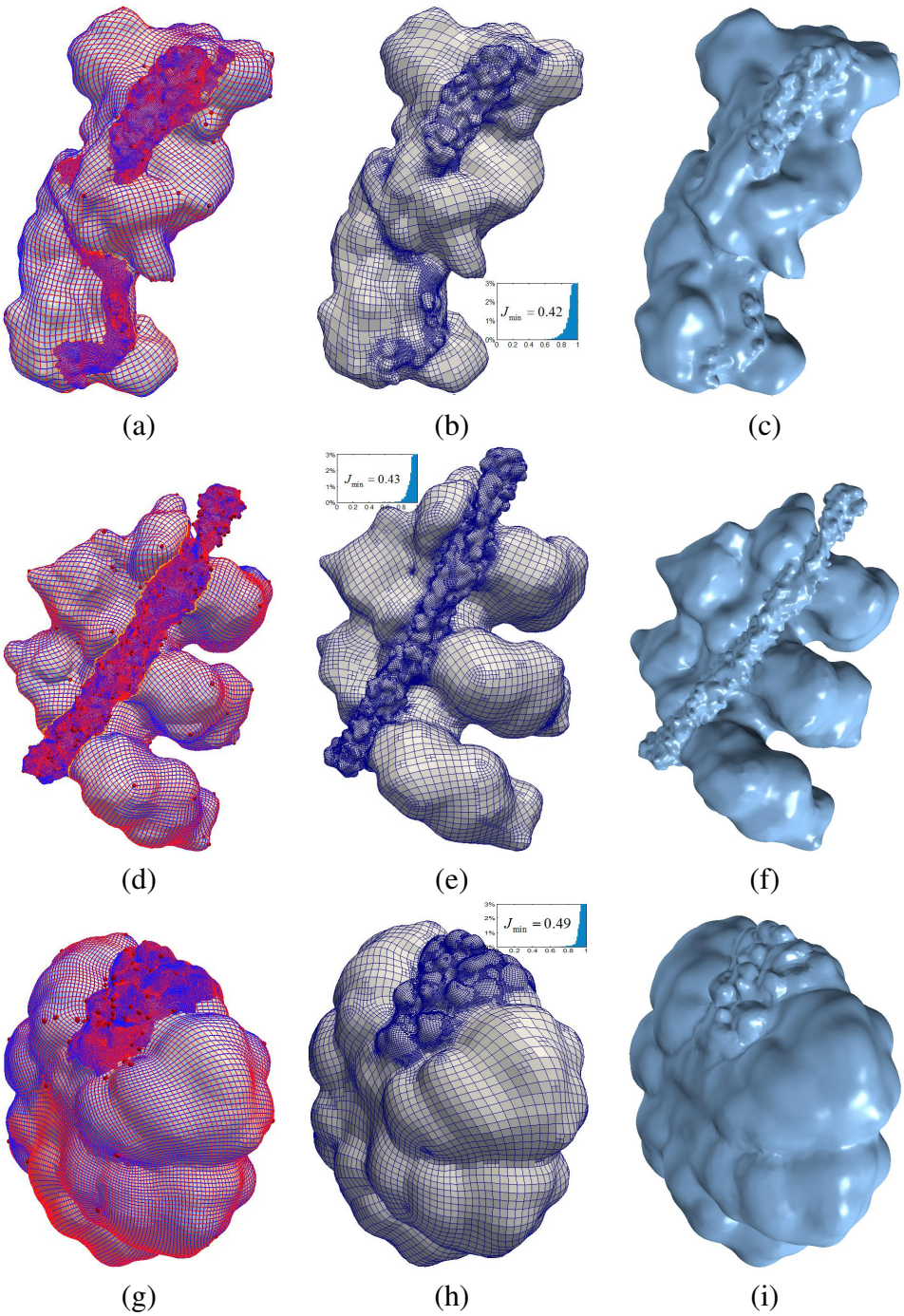


Figure 5.9: Multi-resolution surfaces for 4N78, 4A7F and 2KU2. (a-c) 4KYT; (d-f) 4N78; and (g-i) 2KU2. Left column: adaptive parameterization; Middle column: T-mesh; Right column: T-spline surface.

Table 5.1: T-mesh generation results for biomolecules.

PDB ID		(Vertex No., Element No.)	Number of T-junctions	Number of Singularities	Number of Levels	min_Jacobian $J_{\min}$	Time (s)
2O53	Fig. 5.8(b)	(17,235 15,494)	3,766	156	4	0.47	0.38
4KYT	Fig. 5.8(e)	(11,815, 10,576)	3,272	136	4	0.47	13.3
	Fig. 5.10(g)	(13,557, 12,251)	2,770	56	4	0.59	13.6
	Fig. 5.10(h)	(16,432, 15,045)	2,270	62	4	0.37	13.9
	Fig. 5.10(i)	(13,212, 11,942)	2,582	56	4	0.37	13.5
4N78	Fig. 5.9(b)	(14,484, 12,843)	4,782	122	4	0.42	14.2
4A7F	Fig. 5.9(e)	(42,560 38,423)	3,272	172	4	0.43	15.6
2W4U	Fig. 5.1(c)	(30,142, 26,782)	5,214	240	4	0.46	16.3
	Fig. 5.4(c)	(28,993, 26,227)	2,956	226	4	0.39	16.6
	Fig. 5.11(b)	(21,908 19,298)	5,210	168	4	0.56	15.2
2KU2	Fig. 5.9(h)	(25,759, 23,774)	5,433	120	4	0.49	18.3

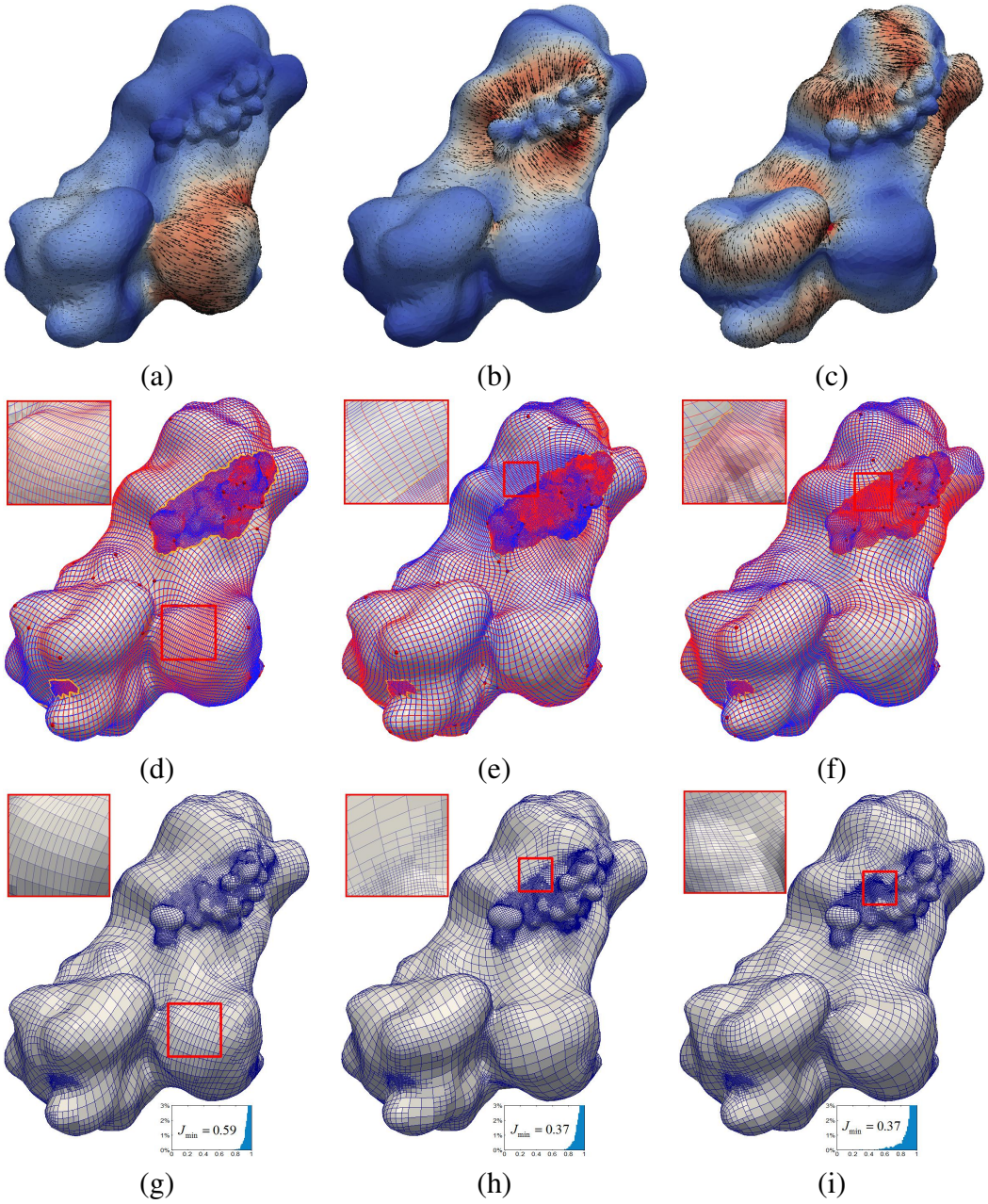
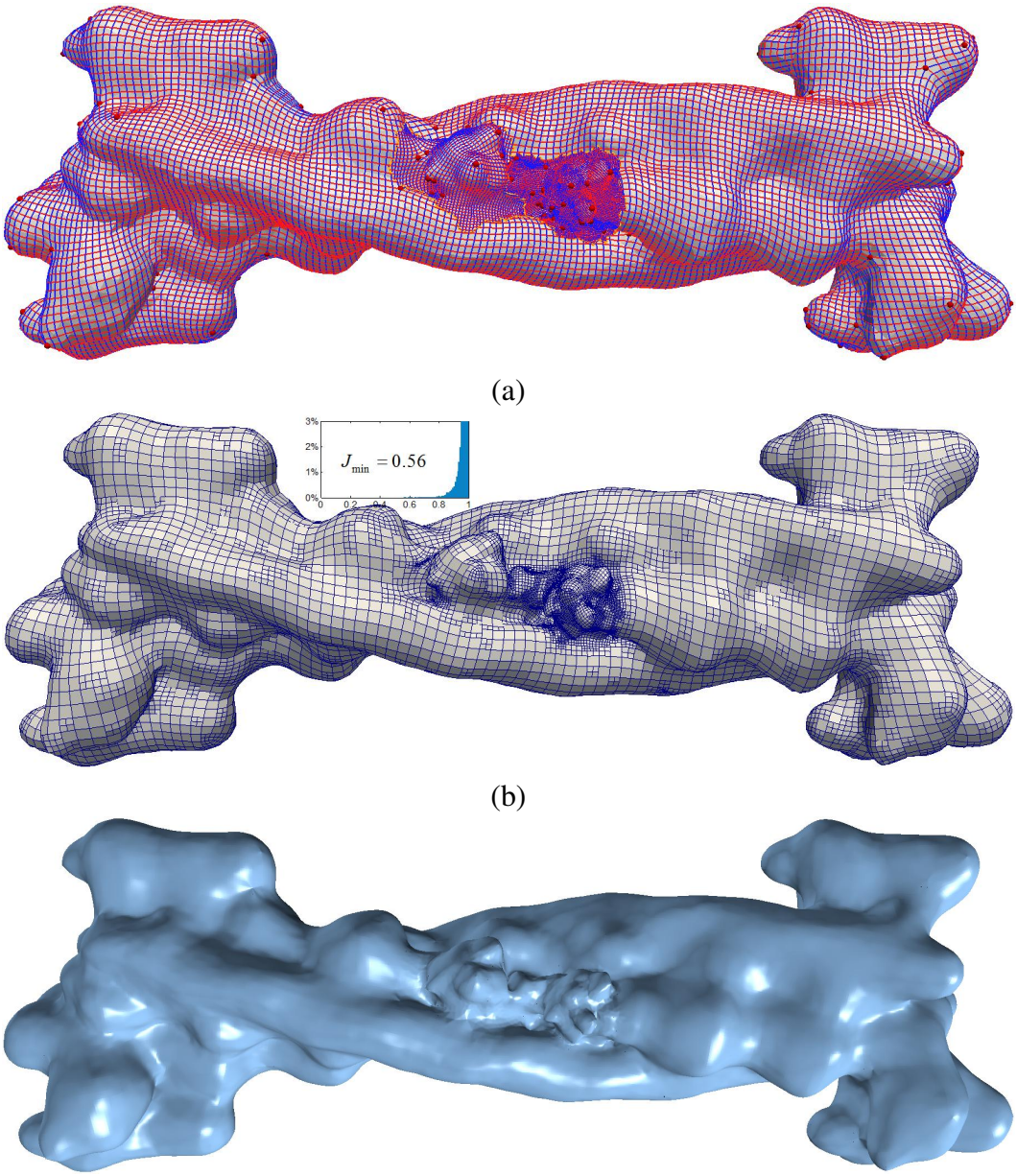


Figure 5.10: Adaptive and anisotropic T-mesh construction of 4KYT from three different eigenmodes. (a-c) The gradient direction and magnitude of the eigenfunctions; (d-f) surface parameterization; and (g-i) adaptive and anisotropic T-meshes. Left column: results from Mode 2; Middle column: results from Mode 3; and Right column: results from Mode 6.



(a)

(b)

(c)

Figure 5.11: Biomolecular surfaces for 2W4U emphasizing Chains 18 and 20 with different resolutions. (a) Adaptive parameterization; (b-c) the corresponding T-mesh and T-spline surface.

5.5 Conclusion and Future Work

In this chapter, we have introduced a new approach to simplify low-contributing atoms and generate quality T-meshes for multi-resolution biomolecular surfaces. An error-bounded atom elimination algorithm is designed to reduce the atom number and preserve multi-resolution surface feature at the same time. An extended cross field-based parameterization is introduced to generate adaptive and anisotropic T-meshes, which can be used further for T-spline surface construction. In addition, a new gradient flow-based method is introduced for T-mesh quality improvement, preserving the anisotropy in the input T-mesh.

Isogeometric analysis has been applied in a lot of engineering fields, it also has a great potential for applications in computational biology to study biomolecular complexes or proteins. In the future we intend to explore along this direction.

Chapter 6

Correspondence Analysis for Biomolecules Based on Volumetric Eigenfunctions

With the geometric models, various studies for shape analysis can be performed for biomolecules. In this chapter, we introduce a novel shape correspondence analysis method based on volumetric eigenfunctions, which enables applications such as deformation tracking and detailed shape comparison for biomolecular shapes.

6.1 Introduction

Biomolecules such as proteins are the basic functional units of biological processes. Since the biomolecular interactions depend heavily on their surface shapes, shape analysis such as deformation tracking and shape comparison become essentially important for the study of biomolecules. In previous studies, various methods have been developed to measure the shape difference [71]. The shape of an object can be characterized by the statistical information of the surface features. In [41], the outlines of different biomolecular shapes were

described and compared based on the multi-resolution Reeb graph (MRG). A robust skeleton extraction method was developed for shape analysis of arbitrary-genus surfaces [70]. Spectral analysis compares shapes based on the spectrum of the Lapace-Beltrami operator (LBO) [103, 104, 32]. Mean shift clustering and dimension reduction are coupled together to accelerate the computation and preserve geometrical structure for shape classification [61]. These methods focus on scoring the similarities between shapes for retrieval or classification, and they usually cannot detect detailed shape differences.

Different from shape retrieval, shape correspondence analysis explicitly provides details of shape similarity or difference [50]. Since eigenfunctions of the graph Laplacian or the LBO are invariant to deformations, they can be used to find correspondence between two flexible objects [123, 19]. Usually, each point in one shape is associated to another using the Gaussian proximity matrix in a supervised [108, 111, 16] or unsupervised [86] way. However, the invariance of eigenfunctions may be affected by large deformations, leading to inaccurate correspondence results [101, 92]. Rigid and nonrigid transformations were adopted in [86, 79, 80, 47] to reduce the influence of these perturbations. In [81], a joint graph was defined based on an initial correspondence between similar features on the cerebral cortex surfaces, providing a guidance for the eigenfunction computation. In [79, 80], a fast computation method was introduced for correspondence analysis of large models. Instead of only considering several landmarks [50], shape analysis of biomolecules prefers to a dense point correspondence.

In this chapter, we develop a novel spectral shape correspondence analysis method based on the joint graph, matching each point in the volumetric shape of biomolecules. The main contributions of this chapter include:

1. A new method is introduced for the initial correspondence construction, which matches boundary nodes with a similar shape diameter distribution and guides the volumetric eigenfunction computation;

2. A two-step scheme is developed to correct node association violating the main structures, and preserve local neighborhoods of boundary nodes by minimizing the distortion from deforming one shape to another; and

3. Based on the correspondence results between biomolecular shapes, a shape approximation and prediction method is introduced, together with a quantitative comparison of different shapes.

The remainder of this chapter is organized as follows. Section 6.2 briefly reviews spectral shape correspondence analysis. Section 6.3 describes the eigenfunction computation from the joint graph of different shapes, and the two-step scheme for shape correspondence analysis. Section 6.4 presents two applications of our shape correspondence results. Finally, Section 7.6 draws conclusions and points out future directions.

6.2 Review of Spectral Point Correspondence Computation

Biomolecular shapes can be obtained from the electron microscope data, or they can be computed according to the atomic resolution structures. In this chapter for a given biomolecular structure from the Protein Data Bank (PDB), we compute its Gaussian density map in a rectilinear grid based on the multi-level summation of Gaussian kernel functions [145, 75]. The shapes are represented by interior voxels where the density is greater than 1.0. For each shape, a graph can be built by connecting the voxel centers with their neighbors. In this chapter, we denote the center of the i^{th} voxel as Node i. Eigenfunctions are computed based on the graph Laplacian, and the corresponding nodes in different shapes can be matched using the distribution of eigenfunctions.

For Graph $\mathcal{G}_A$ built from Shape A, the adjacency matrix W_A is constructed to describe the connections between nodes. Entries of W_A can be computed as

$$w_{ij} = \begin{cases} \exp\left(-d_{ij}^2\right) & if\ j \in N(i), \\ 0 & if\ j \notin N(i), \\ 0 & if\ j = i, \end{cases} \tag{6.1}$$

where d_{ij} denotes the distance between Nodes i and j, and $N(i)$ is the set of 1-ring neighbours of Node i. The normalized graph Laplacian matrix can be written as $L_A = D_A^{-1/2} W_A D_A^{-1/2}$, where $D_A = diag\left(\sum_j w_{1j}, \cdots, \sum_j w_{Mj}\right)$ and M is the node number. Solving the eigenproblem of L_A, we have

$$L_A V = -\lambda_A V, \tag{6.2}$$

where V is the eigenvector and λ_A is the eigenvalue. For the $M \times M$ matrix L_A, M eigenvalues can be obtained with their corresponding eigenvectors, which are ordered according to the ascending order of eigenvalues. Let φ_k^i be the i^{th} element of the k^{th} eigenvector, we have the eigenfunction

$$\varphi_k(\mathbf{x}_i) = \varphi_k^i, \tag{6.3}$$

where $\mathbf{x}_i$ is the position vector of Node i. Each eigenfunction is normalized to $[-1, 1]$.

The correspondence between two different Shapes A and B can be obtained by associating a node in Shape A to the corresponding node in Shape B. In this chapter, Shapes A and B are called the *observed shape* and the *targeted shape*, respectively. Based on the first K eigenfunctions, the nodes in both shapes can be mapped onto a K-dimensional space. Suppose η_k^j is the k^{th} eigenfunction value at Node j in Shape B. For Node i in Shape A and Node j in Shape B, we have

$$\mathbf{x}_i = (x_i, y_i, z_i)^T \ \mapsto\ \tilde{\mathbf{x}}_i = \left(\varphi_1^i, \varphi_2^i, ..., \varphi_K^i\right)^T,$$

and

$$\mathbf{y}_j = \left(x_j, y_j, z_j\right)^T \;\mapsto\; \tilde{\mathbf{y}}_j = \left(\eta_1^j, \eta_2^j, ..., \eta_K^j\right)^T.$$

Since these eigenfunctions are invariant to the deformations, the corresponding nodes should have similar coordinates in the K-dimensional space. The corresponding node $\beta(i)$ of Node i can be found as the one with the shortest distance in the K-dimensional space [81],

$$\beta(i) = \arg\min_j \left\|\tilde{\mathbf{y}}_j - \tilde{\mathbf{x}}_i\right\|. \tag{6.4}$$

In practice, manual work is usually required to order different modes before mapping the nodes. This is because modes with similar patterns may switch the order for different shapes [87].

Discussion 2.1. The LBO eigenfunctions on a triangle surface can be obtained with a cotangent scheme discretization [74]. Fig. 6.1 shows the comparison between surface eigenfunctions from the LBO and volumetric eigenfunctions of 2BPF and 2BPG, two proteins with similar chemical structure but different poses with topological change. Compared to the surface eigenfunctions from the LBO in (a-b), volumetric eigenfunctions in (c-d) tend to be less sensitive to the topological change of the model, which is very common for many biomolecules. The red and blue dots in the first mode represent the maximum and minimum values. Note that these dots may switch the colors due to the sign flipping of eigenfunctions [86]. For a slim stick-like structure, these dots basically reflect the head and tail locations. In volumetric eigenfunctions, we can observe that their locations are barely affected by the bending deformation, while in surface eigenfunctions they move due to the topological change of the surface. Therefore in this chapter, we adopt volumetric eigenfunctions in our shape correspondence analysis.

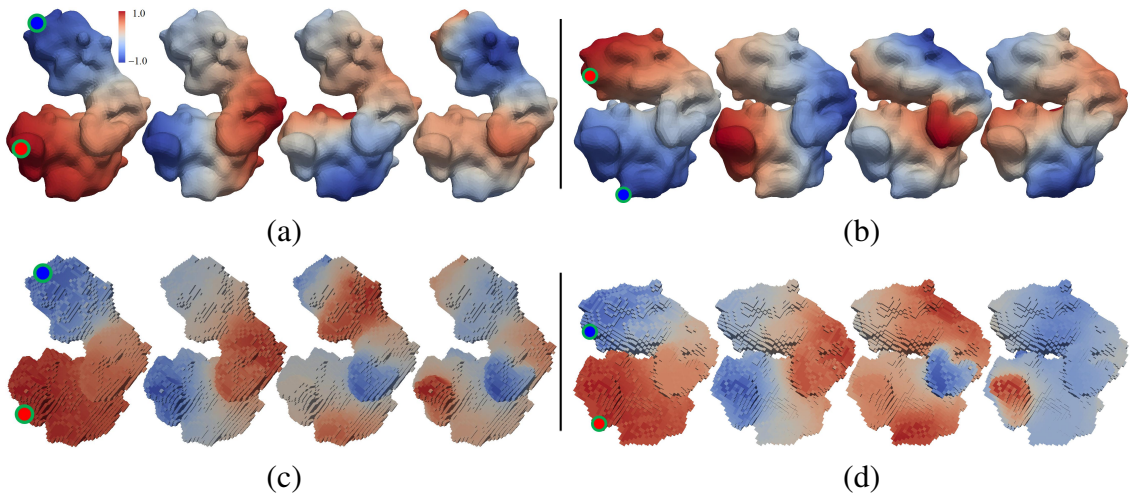


Figure 6.1: The first four modes of 2BPF (a, c) and 2BPG (b, d). (a-b) Surface eigenfunctions from the LBO; and (c-d) volumetric eigenfunctions from the graph Laplacian. The red and blue dots represent the maximal and minimal eigenfunction values, respectively.

6.3 Shape Correspondence Analysis Based on Joint Graph

The invariance of eigenfunctions to different poses may be perturbed by large deformations. As shown in Fig. 6.1(c-d), patterns of the first two modes are very similar with each other, but the third and fourth modes are obviously different. These different modes may result in incorrect node association, failing to reflect the shape similarity. In this chapter, we introduce a new spectral shape correspondence analysis approach based on the joint graph of two shapes. Surface features are used to guide the volumetric eigenfunction computation and improve the shape correspondence.

6.3.1 Eigenfunctions of Joint Graphs

Instead of solving eigenproblems of graphs $\mathcal{G}_A$ and $\mathcal{G}_B$ independently, we can compute the eigenfunctions using the joint graph of these two shapes [81]. The $(M + N) \times (M + N)$

adjacency matrix of the joint graph can be represented as

$$W_C = \begin{pmatrix} W_A & W_{AB} \\ W_{AB}^T & W_B \end{pmatrix}, \tag{6.5}$$

where M and N are the node number of Shapes A and B, W_A and W_B are the adjacency matrices of graph $\mathcal{G}_A$ and $\mathcal{G}_B$ respectively, and the $M \times N$ matrix W_{AB} describes the *initial correspondence* between these two shapes. Additional information can be included in W_{AB} to guide the eigenfunction computation. Let C be the set of nodes in the observed shape (Shape A) where the initial correspondence is defined, the entries in W_{AB} are initialized as

$$w_{ij} = \begin{cases} 0 & if\ i \notin C \\ 0 & if\ i \in C\ \&\ \beta(i) \neq j \\ c & if\ i \in C\ \&\ \beta(i) = j \end{cases} \tag{6.6}$$

where c is a constant. The eigenvectors from matrix W_C in Eq. (6.5) are $(N + M) \times 1$ vectors. The first N elements of the k^{th} eigenvector are embedded in Shape A and the following M elements are embedded in Shape B. The shape correspondence can be obtained based on the shortest distance similar to Eq. (6.4). The initial correspondence builds a set of virtual connections between different shapes, guiding the distribution of the resulting eigenfunctions.

6.3.2 Initial Correspondence

Eigenfunctions from the joint graphs depend on the definition of initial correspondences. In previous literature, the initial correspondences are usually built according to known landmarks such as specific neocortices for the brain surfaces [81], which are hard to specify for an arbitrary pair of biomolecules. Instead, here we detect the regions with similar surface features in different biomolecular shapes and use them to define the initial correspondence.

These regions imply similar components in the chemical structures, leading to the best correspondence results. In this chapter, we characterize the local surface feature around each boundary node using the local shape diameter distribution. By connecting nodes with similar surface features, the initial correspondence can be built.

After that, a distance field from the boundary nodes is constructed throughout the volumetric model. A skeleton is constructed with all the local maxima of the distance field. The shape diameter d_i of each boundary node i is the shortest distance from it to the skeleton. Let $d_{\min}$ and $d_{\max}$ be the minimal and maximal shape diameters for all the boundary nodes respectively, and the diameter is normalized as $d_i = 2\,(d_i - d_{\min}) \,/\, (d_{\max} - d_{\min}) - 1$. The neighborhood $\tilde{N}(i)$ around Node i consists of boundary nodes lying within a certain distance d_o, such that $\tilde{N}\,(i) = \left\{ j | \left\| \mathbf{x}_i - \mathbf{x}_j \right\| < d_o \right\}$. In this chapter we choose $d_o = 18\,\mathring{\mathrm{A}}$. The shape diameter distribution is summarized as a histogram with L bars, see Fig. 6.2. Then each boundary node i can be mapped onto an L-dimensional space with coordinate $\mathbf{u}_i = \left[\mu_1^i, \mu_2^i, ..., \mu_L^i\right]^T$. Nodes with similar surface features should be close to each other in the L-dimensional space.

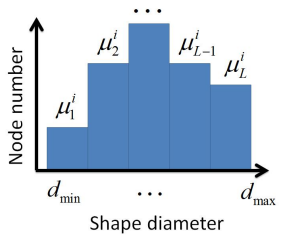


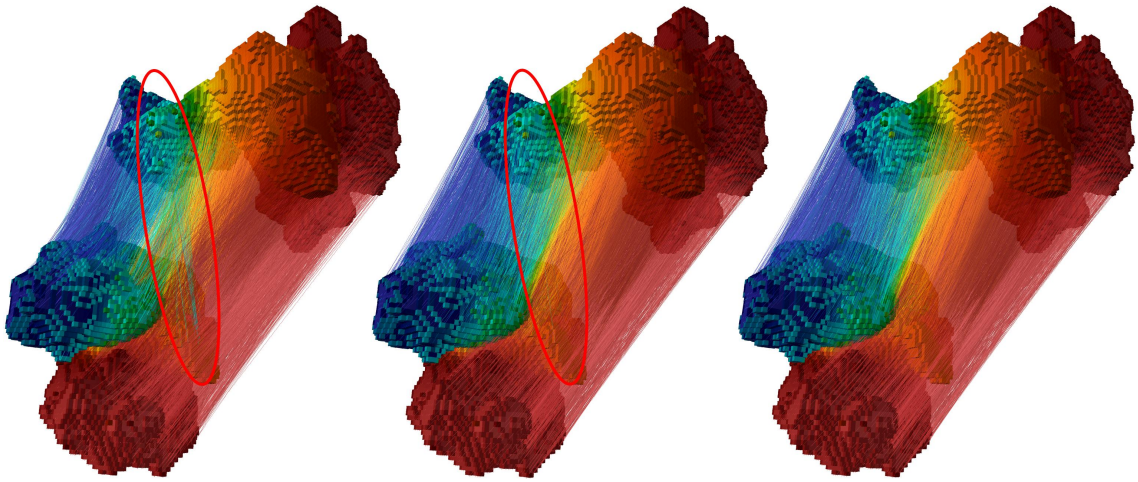
Figure 6.2: The histogram summarizing the shape diameter distribution around Node i.

Nodes at different locations of the shape (e.g. head and tail) may happen to have similar surface features. Therefore, an additional dimension is considered in the initial correspondence definition based on the first eigenfunctions shown in Fig. 6.1(c-d). Then Node i in Shape A is mapped onto an $(L+1)$-dimensional space with coordinate $\mathbf{u}_i = \left[\mu_1^i, \mu_2^i, ..., \mu_L^i, \varphi_1^i\right]^T$, and Node j in Shape B is located at $\mathbf{v}_j = \left[\nu_1^j, \nu_2^j, ..., \nu_L^j, \eta_1^j\right]^T$. If sign flipping happens between φ_1 and η_1, we need to correct it beforehand. Suppose set C contains all the boundary nodes in Shape A. Each boundary node i is connected to the closest

node in Shape B, then we have

$$\beta(i) = \arg\min_j \left\| \mathbf{u}_i - \mathbf{v}_j \right\|. \tag{6.7}$$

Fig. 6.3(a) shows the initial correspondences between 2BPF and 2BPG, which are obtained directly from Eq. (6.7). Each pair of corresponding nodes are connected by a straight line. Fig. 6.4(a-b) show the eigenfunctions resulted from Fig. 6.3(a), with similar patterns on both shapes. However, some nodes are still incorrectly matched, which disobey the main structure of two shapes, see the red circle region in Fig. 6.3(a). This is mainly caused by the limited resolution of the voxel-based models. Therefore, we introduce a two-step scheme to improve the initial correspondence.



(a) (b) (c)

Figure 6.3: The correspondence between shapes of 2BPF (top) and 2BPG (bottom). (a) The initial correspondnence; (b) the correspondence after the first step of improvement; and (c) the final shape correspondence. Each voxel is colored with the first eigenfunction from independent graphs of 2BPF and 2BPG for visualization.

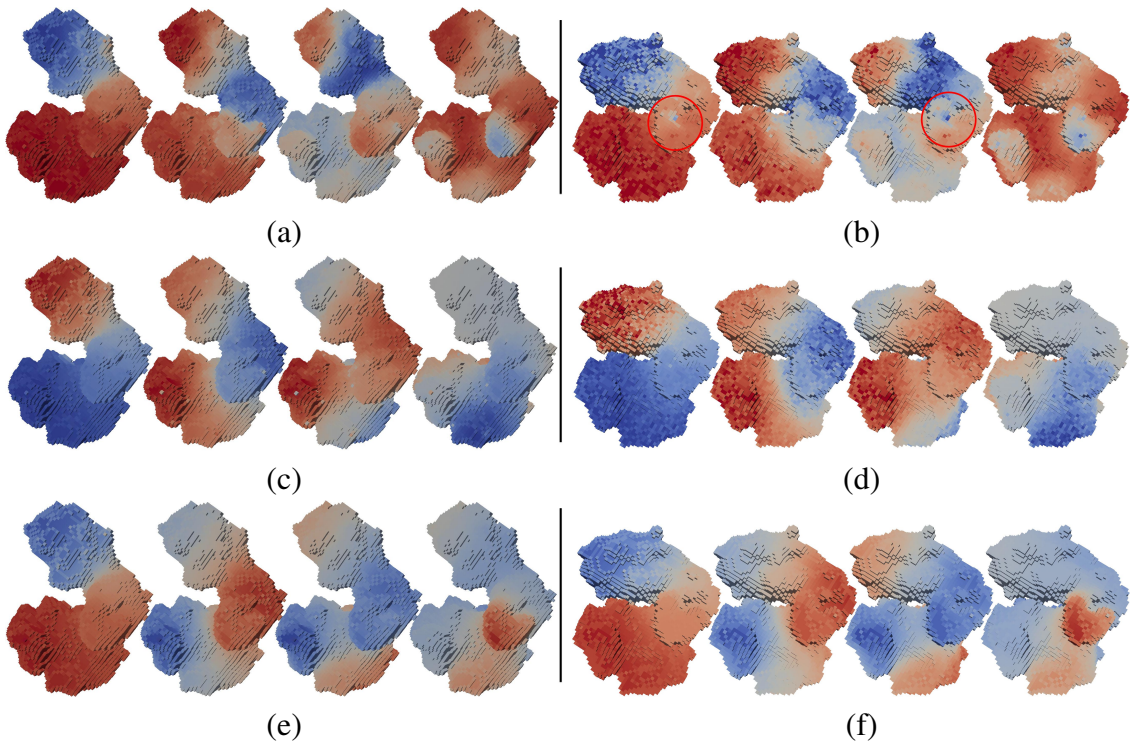


Figure 6.4: First four modes of 2BPF (left) and 2BPG (right) from the joint graph. (a-b) Eigenfunctions resulted from the initial correspondence in Fig. 6.3(a); (c-d) eigenfunctions resulted from the correspondence in Fig. 6.3(b); and (e-f) eigenfunctions resulted from the final correspondence in Fig. 6.3(c).

6.3.3 Initial Correspondence Improvement

Although surface features have been included in the shape diameter-based initial correspondence, it is still very rough in matching two shapes. In the following, the initial correspondence is improved in two steps.

Step 1 (Updating the initial correspondence with eigenfunctions). In the initial correspondence, the connections disobeying the main structures can be taken as high-frequency noises in the joint graph, see the red circle regions in Fig. 6.3(a) and Fig. 6.4(b). The influence of these noises can be reduced by the low mode eigenfunctions, which filter out high-frequency features. With these eigenfunctions, we can update the initial correspondence. As shown in Fig. 6.3(b), most incorrect connections are eliminated and the joint graph is updated, leading to much smoother eigenfunctions, see Fig. 6.4(c-d).

When we deform the observed shape to match with the targeted shape by relocating each node to its corresponding one, the deformation of the local neighborhood around each node should be restricted. As shown in Fig. 6.5 when the shape of 2BPF in (a) is deformed to match 2BPG according to the initial correspondence, the corresponding nodes of the same neighborhood may be faraway to each other, see the red circle region in (b). This is mainly because there are some ambiguities when matching nodes based on the shape diameter. As shown in (c), the local neighborhoods are still not preserved well after the first step improvements. This problem can be resolved using an iterative distortion minimization (Step 2).

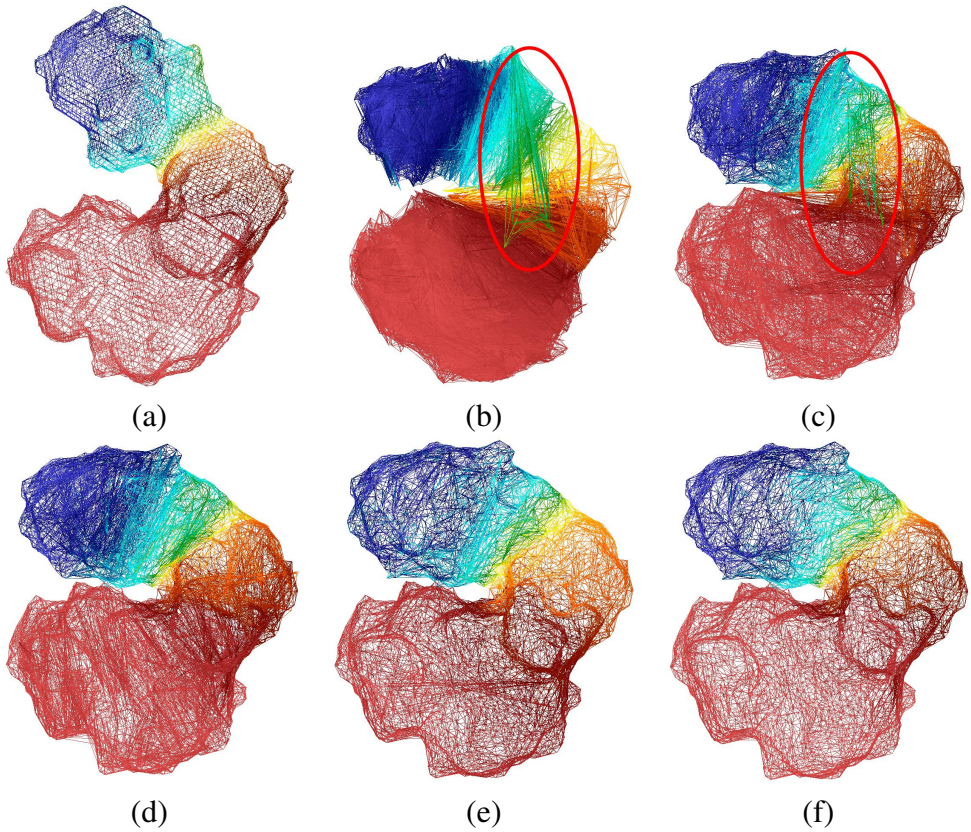


Figure 6.5: Deformed 2BPF shape based on shape correspondence. (a) The original 2BPF shape; (b) the deformed shape based on the initial correspondence; (c) the deformation after the first step improvement; and (d-f) the deformed shapes in the 10^{th}, 75^{th} and 150^{th} iterations in the second step improvement.

Step 2 (Preserving local neighborhoods by minimizing distortions). The distortion due to the correspondence-based deformation can be defined as

$$E = \sum_{i \in C} \sum_{j \in \tilde{N}(i)} \left\| \left(\mathbf{x'}_i - \mathbf{x'}_j \right) - R_i \left(\mathbf{x}_i - \mathbf{x}_j \right) \right\|, \tag{6.8}$$

where $\mathbf{x}_i$ and $\mathbf{x'}_i$ are the locations of Node i and its corresponding node respectively, and R_i is the rotation matrix of the neighborhood around Node i. R_i can be approximated using the as-rigid-as-possible deformation [117]. The covariance matrix S_i for Node i can be computed as

$$S_i = \sum_{j \in \tilde{N}(i)} \mathbf{e}_{ij} \mathbf{e'}^T_{ij}, \tag{6.9}$$

where $\mathbf{e}_{ij}$ and $\mathbf{e'}_{ij}$ represent edges connecting Nodes i and j, and their corresponding nodes, respectively. Applying the singular value decomposition, we have

$$S_i = U_i \Sigma V_i^T. \tag{6.10}$$

Then the rotation matrix can be approximated as

$$R_i = V_i U_i^T. \tag{6.11}$$

In this chapter, a gradient flow method is applied for minimizing the distortion E. Let $\mathbf{F}^n = [x'_1, y'_1, z'_1, ..., x'_{N_C}, y'_{N_C}, z'_{N_C}]^T$ be the position vector of the corresponding nodes in the n^{th} iteration, we update it in the $(n+1)^{th}$ iteration as

$$\mathbf{F}^{n+1} = \mathbf{F}^n - r \nabla \mathbf{F}, \tag{6.12}$$

where N_C is the node number in set C, r is the step length, and the gradient $\nabla\mathbf{F}$ is defined as

$$\nabla\mathbf{F} = \left[\frac{\partial E}{\partial x'_1}, \frac{\partial E}{\partial y'_1}, \frac{\partial E}{\partial z'_1}, ..., \frac{\partial E}{\partial x'_{N_C}}, \frac{\partial E}{\partial y'_{N_C}}, \frac{\partial E}{\partial z'_{N_C}}\right]^T.$$

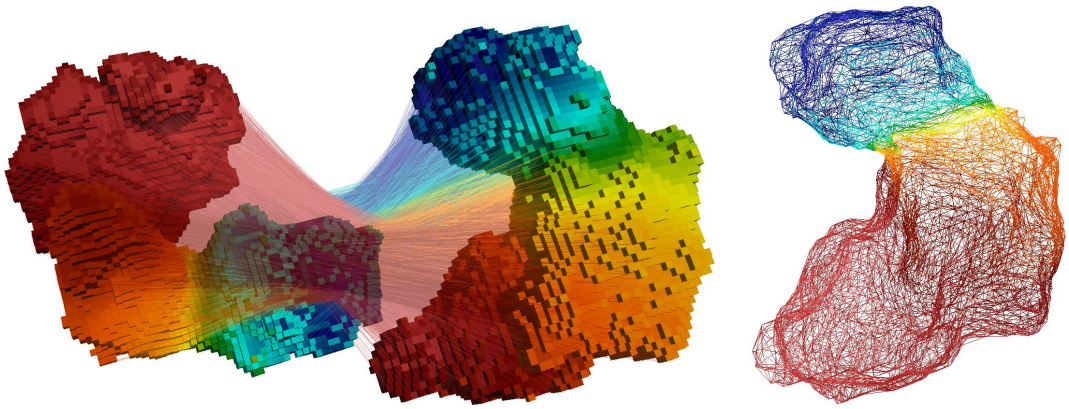
In each iteration, the corresponding nodes are updated with the closest boundary nodes in the targeted shape.

Fig. 6.5(d-f) shows the evolution of 2BPF using our two-step scheme, and it is obvious that the local neighborhood is preserved iteratively. The minimization process terminates when $\frac{|E^n - E^{n+1}|}{E^n} < \epsilon_E$, where E^n and E^{n+1} are the distortion energy in the n^{th} and $(n+1)^{th}$ iterations, and ϵ_E is a predefined threshold (here we choose $\epsilon_E = 0.01$). After the distortion minimization, the joint graphs are updated for eigenfunction and shape correspondence computation. Fig. 6.4(e-f) are the resulting eigenfunctions with consistent patterns for each pair of corresponding eigenfunctions, and Fig. 6.3(c) shows the final shape correspondence between 2BPF and 2BPG, where all the connections follow the main structures.

Discussion 3.1. Unlike traditional spectral correspondence methods, our algorithm is robust of sign flipping and order confusion due to the usage of volumetric eigenfunctions from the joint graph. In addition, our algorithm is quite automatic. Throughout the correspondence analysis, the only required manual work is to correct the eigenfunctions from independent graphs when sign flipping happens during the initial correspondence construction. By minimizing pre-defined distortions, our algorithm preserves the local neighborhood of each boundary node. Therefore, we can track the deformation of local areas and study the behavior of some active sites (e.g. surface pockets).

Discussion 3.2. Due to the invariance of low mode eigenfunctions to the shape deformation, our shape correspondence method is very suitable for matching biomolecules with different poses such as 2BPF and 2BPG. Together with the shape diameter distribution, the resulting correspondence also reflects the association of their mutual chemical components. Note that for biomolecules with different chemical structures such as 2BPF and 1BPB in

Fig. 6.6, although the shape correspondence results follow their similar poses, it does not indicate any association between the chemical structures.



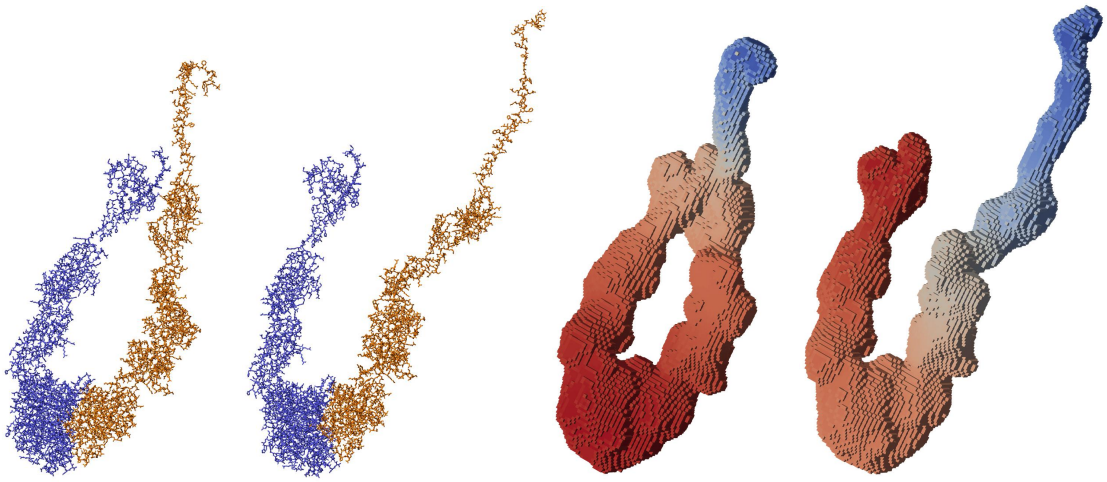
(a) (b)

Figure 6.6: The shape correspondence between 2BPF and 1BPB. (a) The point correspondence between 2BPF (left) and 1BPB (right); and (b) the deformed shape of 2BPF according to (a).

Limitation. Our algorithm also has limitations since it assumes the first eigenfunctions of two similar shapes are invariant to the deformation and also reflect the main structure. For two biomolecules with very different poses and significant topology change as shown in Fig. 6.7, their first eigenfunctions may be very different from each other, and we cannot define any proper initial correspondence. In such situation, more biochemical information can be included to help establish a proper initial correspondence.

6.4 Applications and Results

With the shape correspondence results, various operations can be performed for the shape analysis of biomolecules. In this section, we present two applications using the biomolecular complexes from the Macromolecular Movement Database (http://www.molmovdb.org/), including shape approximation and prediction, as well as shape comparison.



(a) (b) (c) (d)

Figure 6.7: Chemical structure and eigenfunctions for the Integrin. (a-b) The deformation of chemical structure; and (c-d) the first eigenfunctions for the shapes from (a) and (b), respectively.

6.4.1 Shape Approximation and Prediction

In drug design and disease mechanism analysis, it is important to track the deformation of biomolecules over time. Let S_O be the observed shape and S_T be the targeted shape of a biomolecule at time t_O and t_T, the shape of the biomolecule at an arbitrary time t can be approximated by deforming S_O. In this section, the observation time ranges are normalized to $[0, 1]$, we have $t_O = 0$ and $t_T = 1$.

In this chapter, we approximate the intermediate shapes using as-rigid-as-possible deformation [117], which moves a set of landmarks and relocates other nodes by minimizing the distortion. The landmarks are chosen as the local minima and maxima of a harmonic field $h(\mathbf{x})$. Similar to [27], the harmonic field is obtained by solving a Poisson equation

$$\bar{\Delta} h(\mathbf{x}) = \left\| \bar{\Delta} \mathbf{x} \right\|, \tag{6.13}$$

where $\bar{\Delta}$ is the graph Laplacian operator. Let $H = [h_1, h_2, \cdots, h_M]^T$ be a set of discretized function value at each node, we have $\bar{\Delta} H = W_P H$ and $\bar{\Delta}\mathbf{x} = \left[\bar{\Delta}x, \bar{\Delta}y, \bar{\Delta}z\right]^T$. The matrix W_P

is similar but different from the adjacency matrix in Sec. 6.2, and its entries are defined as

$$w_{ij}^{P} = \begin{cases} 0 & if\ j \notin N(i), \\ \exp\left(-d_{ij}^{2}\right) & if\ j \in N(i)\ \&\ i \neq j, \\ -\sum\limits_{k \neq i} w_{ik}^{P} & if\ i = j. \end{cases} \tag{6.14}$$

Fig. 6.8 shows the harmonic field in 2BPF, in which the blue and red dots represent the local minima and maxima, respectively. The location of a landmark at time t is approximated as

$$\mathbf{x}(t) = \mathbf{x}_O + \frac{t - t_O}{t_T - t_O}\left(\mathbf{x}_T - \mathbf{x}_O\right), \tag{6.15}$$

where $\mathbf{x}_O$ is the original location in Shape S_O, and $\mathbf{x}_T$ is the corresponding node in Shape S_T. Other nodes can be obtained by minimizing a distortion energy

$$D(\mathbf{F}) = \sum_{i=1}^{M} \sum_{j \in N(i)} \left\| \left(\mathbf{x'}_i - \mathbf{x'}_j\right) - R_i\left(\mathbf{x}_i - \mathbf{x}_j\right) \right\|, \tag{6.16}$$

where $\mathbf{F} = [x'_1, y'_1, z'_1, ..., x'_M, y'_M, z'_M]^T$ is the position vector of all the nodes at time t. R_i is the rotation matrix for Node i, which can be approximated based on the SVD of the local covariance matrix described in Sec. 6.3.3.

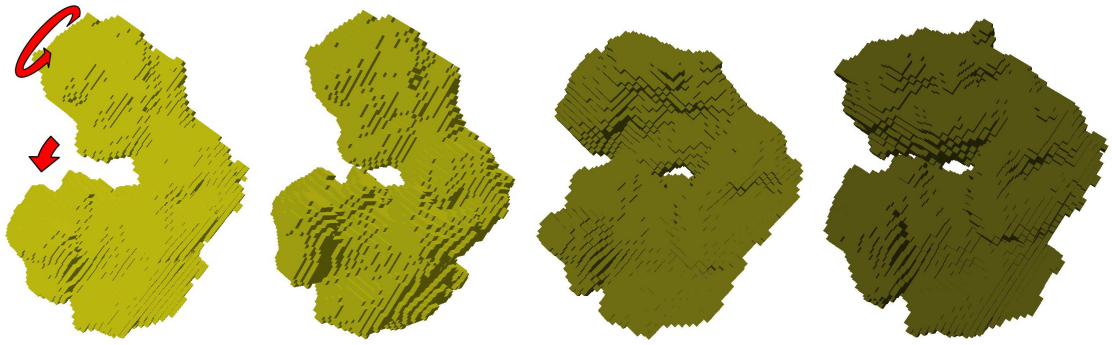
The approximated shape is represented by voxels in a rectilinear grid. First, a blank grid is initialized with the same resolution with the observed shapes. For the deformed shape S_O, each node is connected with its 1-ring neighbors by several edges, and all the voxels intersecting with these edges are marked as voxels in the approximated shape. The resulting binary data can be smoothed by filtering out the high-frequency components.

Fig. 6.9 shows the evolving process from 2BPF to 2BPG. By bending and twisting the shape of 2BPF, we can obtain the shape of 2BPG, see the red arrows in (a). As shown in (b, c), it is easy to handle the topological change of the surface using voxels. While Fig. 6.9 is a conjectural process, Fig. 6.10(a-d) shows a simulated deformation process of

an enzyme complex of yeast (YE). (e-h) are the real biomolecular shapes built from (a-d). We suppose the shapes in (e) and (h) are observed at $t = 0$ and $t = 1$, respectively. As shown in (i), the shape correspondence is computed between these two shapes, from which we can approximate the biomolecular shapes at $t = 0.33$ and $t = 0.67$, see (m) and (n). (j-k) show the original neighborhoods and the deformed ones according to (i). To measure the difference between the real shape and the approximated one, a volume error ϵ_V is defined as

$$\epsilon_V = \frac{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\sum_{k=1}^{N_z}\left|\rho_{i,j,k} - \rho'_{i,j,k}\right|}{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\sum_{k=1}^{N_z}\rho_{i,j,k}}, \tag{6.17}$$

where N_x, N_y and N_z are dimensions of the rectilinear grid in x, y and z directions, respectively. If a voxel with index (i, j, k) belongs to the real shape or the approximated shape, we set $\rho_{i,j,k} = 1$ or $\rho'_{i,j,k} = 1$, otherwise $\rho_{i,j,k}$ and $\rho'_{i,j,k}$ are set to be zero. As shown in Fig. 6.10(m-n), the approximated shapes are very similar to the real ones in (f-g). Besides approximating the shape during the observed time, we can also predict the shapes out of the time range. Fig. 6.10(l) and (o) are predictions of the shape before and after the observed time range at $t = -0.33$ and $t = 1.33$, respectively.



(a) $t = 0$ (b) $t = 0.33$ (c) $t = 0.67$ (d) $t = 1$

Figure 6.9: The deforming process from 2BPF to 2BPG. (a, d) The known shapes of 2BPF and 2BPG, respectively; and (b-c) the approximated shapes at different time.

In this chapter, landmarks are detected using the harmonic field, which only depend on the geometrical structure of biomolecules. Our method is flexible and extensible to

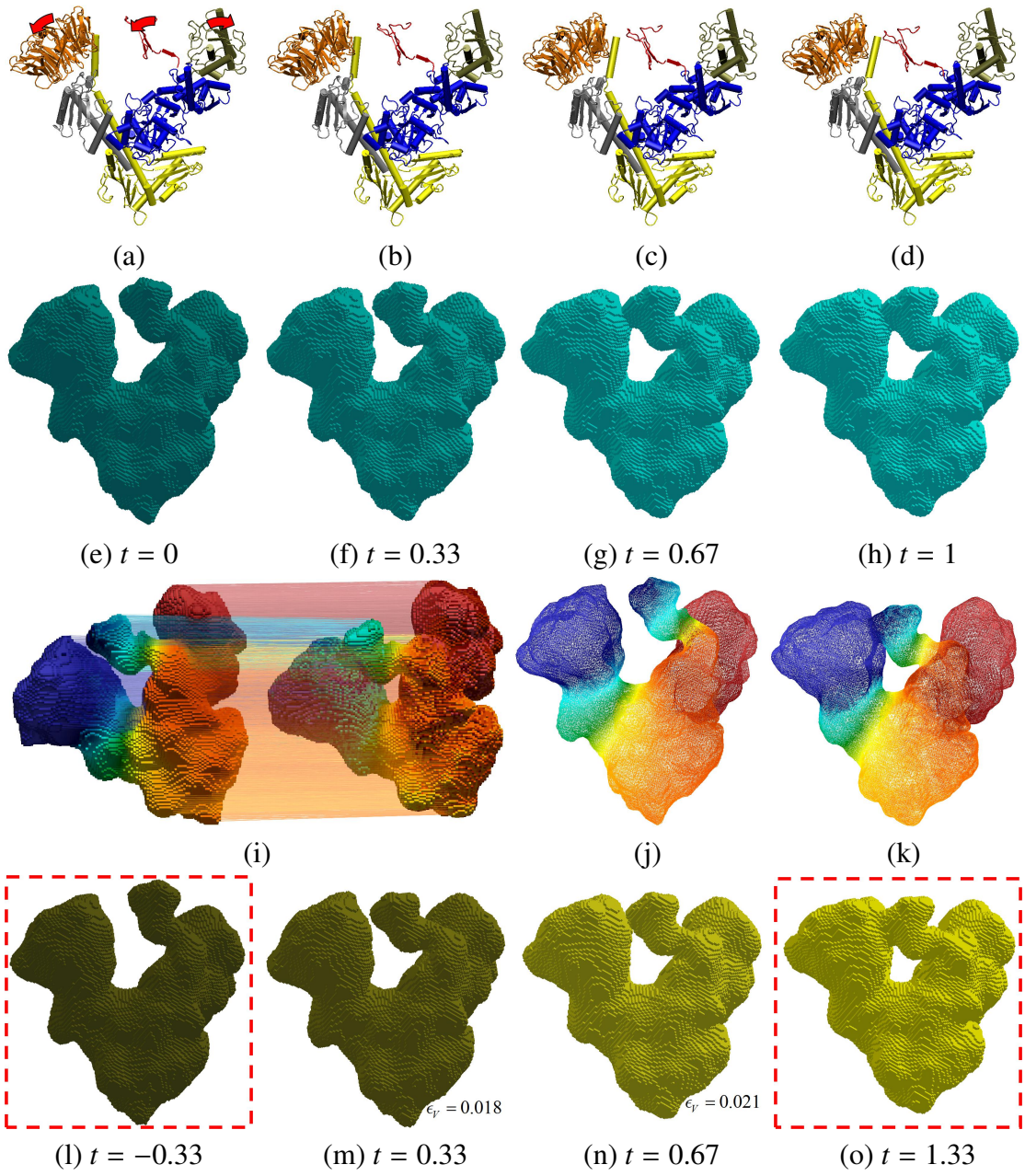


Figure 6.10: The deformation process of YE. (a-d) Deformation process of the chemical structure of YE; (e-h) volumetric shapes built from (a-d); (i) shape correspondence between (e) and (h); (j-k) the original and deformed shapes of (e) according to the shape correspondence; and (l-o) the approximated and predicted deformed shapes at different time. (i-k) are colored with the first eigenfunction from independent graphs for visualization.

incorporate additional biochemical information by specifying landmarks with important chains or residues, leading to more realistic matching results.

6.4.2 Shape Comparison

Previous shape retrieving algorithms usually only score the similarities or dissimilarities between two shapes [70, 50], without identifying detailed differences. On the contrary, a quantitative measurement of the shape difference can be performed to show the detailed differences based on our correspondence results. Two variables can be used to measure the shape difference at each node: the shape diameter difference and the local distortion induced by correspondence-based deformation. Let d_i and d_i' be the shape diameter of Node i and its corresponding node in the other shape, the shape diameter difference is simply defined as $\delta_i = |d_i - d_i'|$. We denote the average and maximal shape diameter difference and distortion as $(\bar{e}_S, e_S^{\max})$ and $(\bar{e}_D, e_D^{\max})$ respectively, which can also be used for shape retrieval and classification.

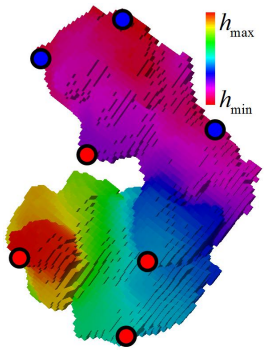
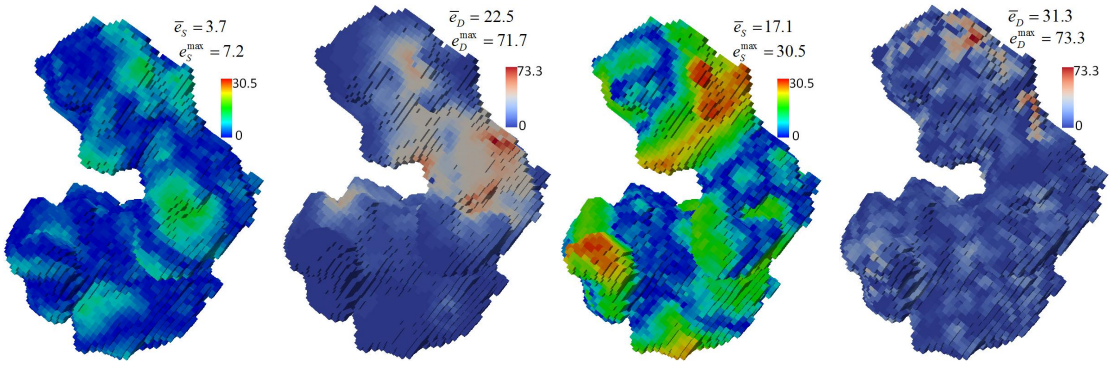


Figure 6.8: The harmonic field and local minima (blue) and maxima (red) in 2BPF.

Based on the shape correspondence results shown in Fig. 6.3(c) and Fig. 6.6(a), we measure the differences between two pairs of protein shapes as shown in Fig. 6.11. The first pair (2BPF and 2BPG) has similar chemical structure but different poses, while the second pair (2BPF and 1BPB) has similar pose but different chemical structures. For convenience of observation, the distribution of shape diameter difference and distortion are both shown over the 2BPF shape. As shown in (a-b), the similar chemical structures result in small shape diameter difference in (a), but a large concentration of distortion happens in (b), which reflects the bending induced by deforming 2BPF to 2BPG. For 2BPF and 1BPB in (c-d), both the shape diameter difference and distortion reflect their difference.

The behavior of a biomolecule during biochemical process can be characterized by combining the information from both the shape diameter and distortion. The KEAP1/CUL3/RBX1 complex (KCR) is an ubiquitin ligase in human. As shown in



(a) (b) (c) (d)

Figure 6.11: Measurement of shape difference. (a-b) Comparison between 2BPF and 2BPG; (c-d) comparison between 2BPF and 1BPB; (a, c) the shape diameter difference; and (b, d) the distortion.

Fig. 6.12(a-d), the KCR structure deforms due to the movement of main components (see the red arrows), which is observed at different time. Besides, we assume the complex loses Chain F (purple) during the deformation. The behavior of this biomolecule can be described by comparing the biomolecular shapes at different time (f-h) (the targeted shapes) with the original one in (e) (the observed shape) based on the shape correspondence results in (i-k). As shown in (l-n), the sudden change of both metrics in Region A implies the change of chemical structure, that is, the loss of Chain F at $t = 0.67$. In Region B, the shape diameter difference is very small while distortion arises over time, which indicates the bending and torsion in this region as well as the relative movement of different components.

6.5 Conclusion and Future Directions

In this chapter, a new shape correspondence analysis method has been developed for biomolecules based on volumetric eigenfunctions from the joint graph of two given shapes. The shape diameter distribution is used to build the initial correspondence, which guides eigenfunction computation with similar surface features preserved. A two-step scheme was developed to analyze the shape correspondence, which eliminates node asso-

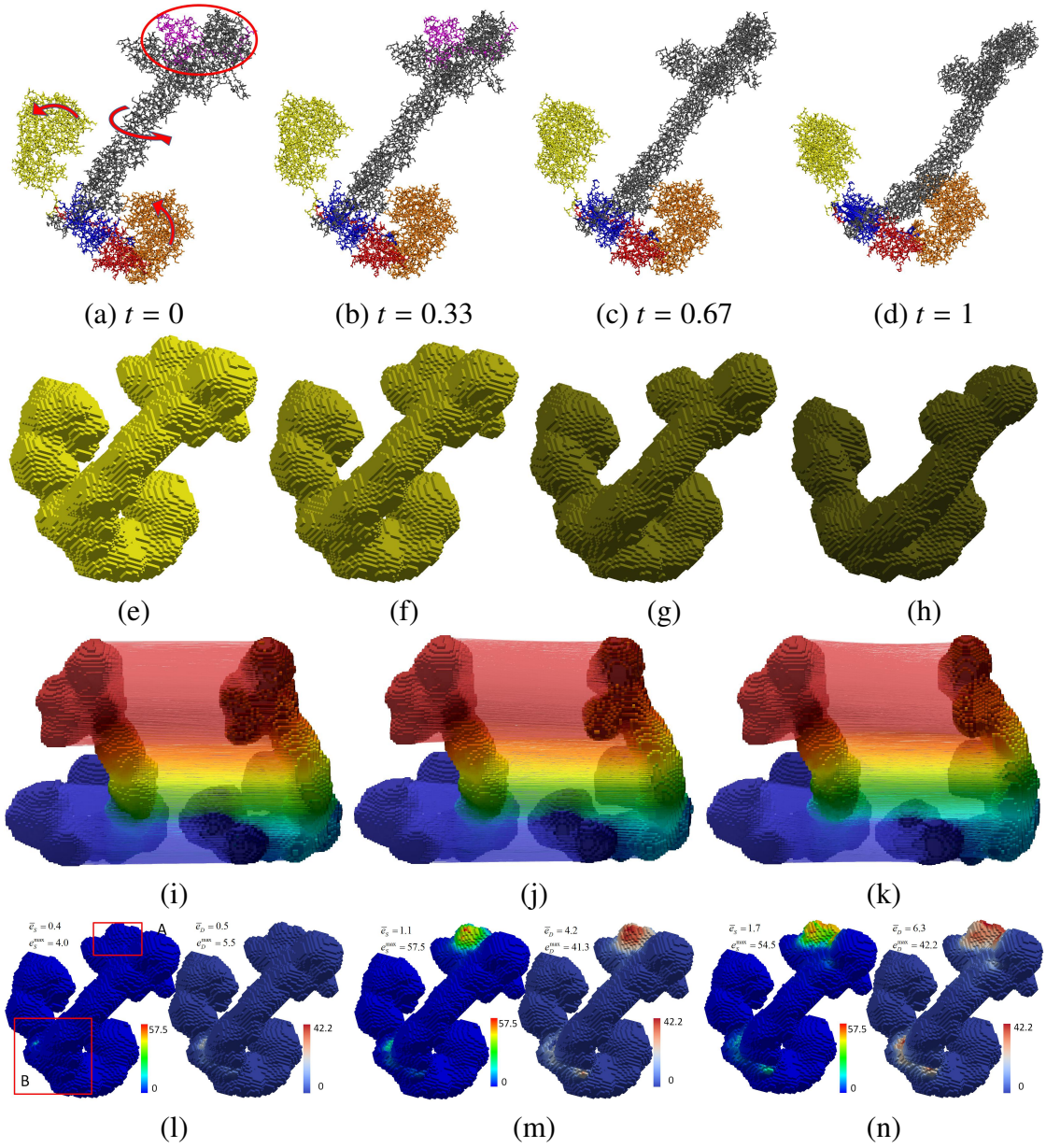


Figure 6.12: Shape deformation of KCR. (a-d) Deformation of the chemical structure; (e-h) volumetric shapes computed from (a-d); (i-k) are shape correspondence between (e) and (f), (e) and (g), and (e) and (h), respectively; (l-n) measurement of the shape differences based on the correspondence in (i-k) with shape diameter difference (left) and distortion (right). (i-k) are colored with the first eigenfunction from independent graphs for visualization.

ciation against the main structure and preserves local neighborhoods of each node. Based on the shape correspondence analysis results, we perform shape approximation, prediction and comparison, which can be used to study the behavior of biomolecules.

Our shape correspondence analysis method is designed based on the geometrical information only. We can include additional biochemical information to improve the matching accuracy. For specific proteins, special chains or ions can be used to help define the initial correspondence, and the resulting shape correspondence will better reflect the deformation of chemical structures. In the future, we plan to improve our method along this direction, and also explore other shape analysis applications for biomolecules.

Chapter 7

Secondary Laplace Operator and Generalized Giaquinta-Hildebrandt Operator with Applications on Surface Segmentation and Smoothing

Some behaviors of biomolecules are determined by special surface features such as pockets. Powerful mathematical tools are desired to detect and segment these features. In this chapter, we introduce two new geometric operators based on the second fundamental form of the surface, and develop various applications for them, including segmenting protein pockets.

7.1 Introduction

Geometric operators are the basis for many algorithms in surface processing. Based on the first fundamental form of the surface, the Laplace-Beltrami operator (LBO) is defined and its eigenfunctions are well-known for their property of capturing shape behav-

ior and structural feature of an object [65, 102, 104, 113]. They vary along the object surface and are invariant to different poses, which makes them ideal for applications in pose-invariant Reeb graph construction [113], shape matching [65] or registration [86], the Shape-DNA [104], and surface quadrangulation or parameterizaton [150, 74, 26, 45, 76]. Another important application of the LBO eigenfunctions is surface segmentation. The distribution of eigenfunctions is used together with isocontours [102, 127, 78] and point clustering [98, 142, 112] to segment surface into several components. In practice, different methods were employed to improve the performance of LBO eigenfunctions [42, 132]. To detect concavities, eigenfunctions from a concavity-aware Laplacian [127] were used to generate a single segmentation field.

Besides computing eigenfunctions, geometric operators are also used in various geometric flows for surface fairing and smoothing [23, 63, 64, 144, 139, 148]. The mean curvature and Gaussian curvatures are usually used to design geometric operators [139, 57]. Generally, geometric flows smooth surfaces and remove noise by moving nodes in the normal direction, while the tangential movement regularizes the elements. The tangential movement can also help strengthen surface features, which was seldom considered in the existing methods.

In this chapter, we introduce two new operators, namely the secondary Laplace operator (SLO) and generalized Giaquinta-Hildebrandt operator (GGHO), based on the second fundamental form of the surface. Different from the LBO eigenfunctions, the SLO eigenfunctions can capture concave and convex surface features which can be used to segment concave and convex regions. The main contribution of this chapter includes

1. Two new geometric operators (SLO and GGHO) are introduced based on the second fundamental form of the surface;

2. Surface segmentation methods are developed based on the SLO eigenfunctions with concave creases/regions and convex ridges detected; and

3. A new geometric flow method is developed based on the GGHO for surface smoothing, which preserves and strengthens sharp features on the surface.

The remainder of this chapter is organized as follows. Section 7.2 reviews the definition of several existing geometric operators and eigenfunction computation. Section 7.3 introduces the SLO and its application on surface segmentation. Section 7.4 talks about the GGHO and a new geometric flow method. Section 7.5 shows the results. Finally, Section 7.6 draws conclusions and points out the future work.

7.2 Review of Existing Geometric Operators

Before introducing our new operators, we first briefly review definitions of several existing geometric operators. More detailed descriptions can be found in [138]. Eigenfunctions of different operators can be computed using isogeometric analysis (IGA) based on Catmull-Clark basis functions [134].

7.2.1 Definitions of Existing Geometric Operators

Let $S = \{\mathbf{x}(u, v), (u, v) \in \Omega \subset \mathbb{R}^2\}$ be a smooth parametric surface in $\mathbb{R}^3$. (u, v) can also be written as (u^1, u^2) for convenience. The coefficients of the first fundamental form of S are defined as $g_{\alpha\beta} = \langle \mathbf{x}_{u^\alpha}, \mathbf{x}_{u^\beta} \rangle$ $(\alpha, \beta = 1, 2)$, where $\mathbf{x}_{u^\alpha} = \frac{\partial \mathbf{x}}{\partial u^\alpha}$ and $\mathbf{x}_{u^\beta} = \frac{\partial \mathbf{x}}{\partial u^\beta}$. The coefficients of the second fundamental form of S are defined as $b_{\alpha\beta} = \left\langle \mathbf{n}, \mathbf{x}_{\alpha\beta} \right\rangle$, where $\mathbf{x}_{u^\alpha u^\beta} = \frac{\partial^2 \mathbf{x}}{\partial u^\alpha \partial u^\beta}$ and $\mathbf{n} = (\mathbf{x}_u \times \mathbf{x}_v)/\|\mathbf{x}_u \times \mathbf{x}_v\|$. Let $g = \det[g_{\alpha\beta}]$, $[g^{\alpha\beta}] = [g_{\alpha\beta}]^{-1}$, and $[b^{\alpha\beta}] = [b_{\alpha\beta}]^{-1}$. We have the following geometric operators.

Curvatures. The mean curvature H and the Gaussian curvature K are defined as

$$H = \frac{b_{11}g_{22} - 2b_{12}g_{12} + b_{22}g_{11}}{2g} \quad \text{and} \quad K = \frac{b_{11}g_{22} - b_{12}^2}{g}. \tag{7.1}$$

Tangential gradient operator. Let $f \in C^1(S)$, the tangential operator ∇ acting on f is given by

$$\begin{aligned} \nabla f &= [\mathbf{x}_u, \mathbf{x}_v] \left[g^{\alpha\beta} \right] [f_u, f_v]^T \\ &= g_u^{\nabla} f_u + g_v^{\nabla} f_v, \end{aligned} \tag{7.2}$$

where $\nabla f \in \mathbb{R}^3$, $g_u^{\nabla} = \frac{1}{g}\left(g_{22}\mathbf{x}_u - g_{12}\mathbf{x}_v\right)$ and $g_v^{\nabla} = \frac{1}{g}\left(g_{11}\mathbf{x}_v - g_{12}\mathbf{x}_u\right)$.

Second tangential operator (STO). Let $f \in C^1(S)$, the STO $\diamond$ acting on f is defined as

$$\begin{aligned} \diamond f &= [\mathbf{x}_u, \mathbf{x}_v]\, K \left[b^{\alpha\beta} \right] [f_u, f_v]^T \\ &= g_u^{\diamond} f_u + g_v^{\diamond} f_v, \end{aligned} \tag{7.3}$$

where $\diamond f \in \mathbb{R}^3$, $g_u^{\diamond} = \frac{1}{g}\left(b_{22}\mathbf{x}_u - b_{12}\mathbf{x}_v\right)$ and $g_v^{\diamond} = \frac{1}{g}\left(b_{22}\mathbf{x}_v - b_{12}\mathbf{x}_u\right)$.

Tangential divergence operator. Let $\mathbf{v}$ be a C^1 smooth vector field on a surface $\mathbf{S}$, the tangential divergence operator div_S acting on $\mathbf{v}$ is defined by

$$div_S(\mathbf{v}) = \frac{1}{\sqrt{g}} \left[\frac{\partial}{\partial u}, \frac{\partial}{\partial v} \right] \left[\sqrt{g} \left[g^{\alpha\beta} \right] [\mathbf{x}_u, \mathbf{x}_u]^T \mathbf{v} \right]. \tag{7.4}$$

Laplace-Beltrami operator (LBO). Let $f \in C^2(S)$, the LBO acting on f is given by

$$\begin{aligned} \triangle f &= div_S(\nabla f) \\ &= \tfrac{1}{\sqrt{g}} \left[\tfrac{\partial}{\partial u}, \tfrac{\partial}{\partial v} \right] \left[\sqrt{g} \left[g^{\alpha\beta} \right] [f_u, f_v]^T \right]. \end{aligned} \tag{7.5}$$

Giaquinta-Hildebrandt operator (GHO). Let $f \in C^2(S)$, the GHO acting on f is given by

$$\begin{aligned} \square f &= div_S(\diamond f) \\ &= \tfrac{1}{\sqrt{g}} \left[\tfrac{\partial}{\partial u}, \tfrac{\partial}{\partial v} \right] \left[\sqrt{g} K \left[b^{\alpha\beta} \right] [f_u, f_v]^T \right]. \end{aligned} \tag{7.6}$$

7.2.2 Eigenfunction Computation

Let $f \in C^2(S)$, the eigenfunctions of LBO and GHO should satisfy

$$\triangle f = -\lambda_l f \;\text{ and }\; \square f = -\lambda_g f \tag{7.7}$$

respectively, where λ_l and λ_g are the eigenvalues. According to the Green formula, for $\forall f \in C^2(S)$ and $\forall h \in C^1(S)$ we have

$$\int_S (h \triangle f + \langle \nabla f, \nabla h \rangle) \, \mathrm{dA} = 0 \tag{7.8}$$

and

$$\int_S (h \square f + \langle \nabla f, \diamond h \rangle) \, \mathrm{dA} = 0. \tag{7.9}$$

Therefore, we plug Eq. (7.7) into Eqs. (7.8-7.9) and obtain

$$\int_S \langle \nabla f, \nabla h \rangle \, \mathrm{dA} = \lambda_l \int_S hf \mathrm{dA} \tag{7.10}$$

and

$$\int_S \langle \nabla f, \diamond h \rangle \, \mathrm{dA} = \lambda_g \int_S hf \mathrm{dA}. \tag{7.11}$$

Let $\{\varphi_i\}_{i=1}^N$ be a set of basis functions defined on the surface and $\varphi_i \in C^2(S)$, f can be approximately represented as $f = \sum_{i=1}^{N} w_i \varphi_i$. Let $h = \varphi_j$ ($j = 1, 2, \cdots, N$), then we have

$$\sum_{i=1}^{N} w_i \int_S \left\langle \nabla \varphi_i, \nabla \varphi_j \right\rangle \mathrm{dA} = \lambda_l \sum_{i=1}^{N} w_i \int_S \varphi_i \cdot \varphi_j \mathrm{dA}$$

and

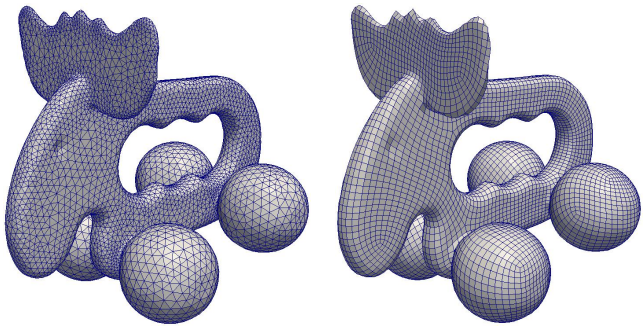
$$\sum_{i=1}^{N} w_i \int_S \left\langle \nabla \varphi_i, \diamond \varphi_j \right\rangle \mathrm{dA} = \lambda_g \sum_{i=1}^{N} w_i \int_S \varphi_i \cdot \varphi_j \mathrm{dA}$$

for the LBO and GHO, respectively. Let $m_{ij}^L = \int_S \left\langle \nabla \varphi_i, \nabla \varphi_j \right\rangle \mathrm{dA}$, $m_{ij}^G = \int_S \left\langle \nabla \varphi_i, \diamond \varphi_j \right\rangle \mathrm{dA}$, and $c_{ij} = \int_S \varphi_i \cdot \varphi_j \mathrm{dA}$. The eigenfunctions of LBO and GHO can be obtained by solving the eigenproblems

$$M^L W = \lambda_l C W \text{ and } M^G W = \lambda_g C W, \tag{7.12}$$

where $M^L = \left[m_{ij}^L\right]$, $M^G = \left[m_{ij}^G\right]$, $C = \left[c_{ij}\right]$, and $W = [w_1, w_2, ..., w_N]^T$. Embedding the eigenvector W on the surface, we obtain an eigenfunction of the specific geometric operator.

In this chapter, we focus on the two new operators based on the second fundamental form of the surface, and a cubic or higher order surface representation is required. While many other methods can also be used, we choose the cubic Catmull-Clark subdivision surface due to its wide adoption [134]. As shown in Fig. 7.1, for the input triangular surfaces we use cross field-based surface parameterization [13, 74] to generate semi-structured quadrilateral meshes with only a few extraordinary nodes. Triangle subdivision schemes such as the Loop subdivision can also be used to study SLO and GGHO, but attentions should be paid to the number and placement of extraordinary nodes. Fig. 7.2(a-b) shows the first four eigenfunctions of the LBO and GHO for the Elk model, respectively. The LBO eigenfunctions vary smoothly on the surface following the main structure of the model, but it is hard to detect curvature-related features (e.g., the concave creases connecting the four wheels, the horn and the main Elk body) because the LBO is defined based on the first fundamental form of the surface, see Fig. 7.2(c) for the segmentation result using the LBO eigenfunctions. On the contrary, the GHO eigenfunctions hardly reveal any geometric feature of the object. This is because $\diamond f$ is almost zero in the planar regions, which makes the matrix M^G close to singular. To better detect curvature-related features, we define two new geometric operators (SLO and GGHO) in the following two sections.



(a) (b)

Figure 7.1: Elk model. (a) The input triangle mesh; and (b) the quadrilateral mesh.

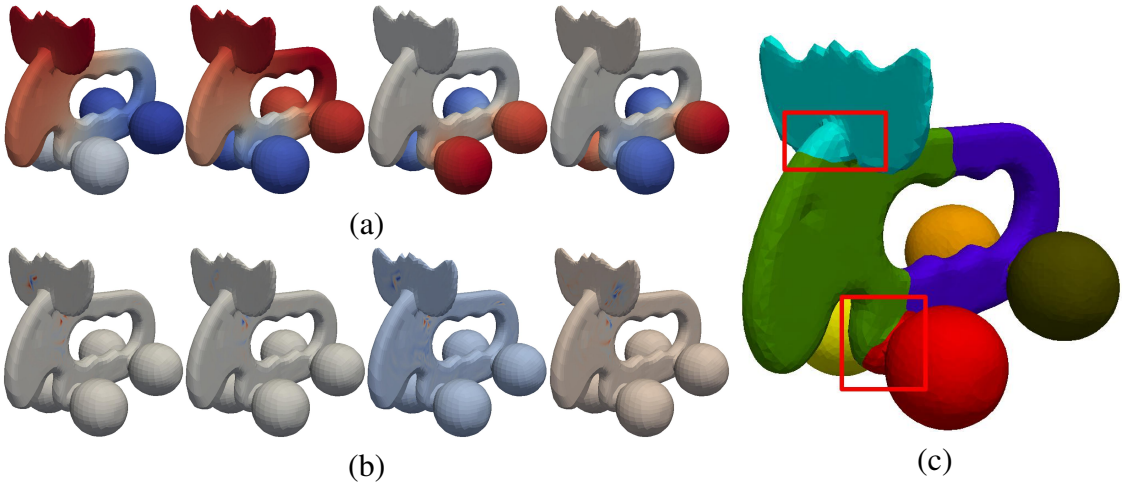


Figure 7.2: Elk model. (a, b) The first four non-constant eigenfunctions of the LBO and GHO for the Elk model, respectively; and (c) the segmentation result from Modes 1-4 of the LBO.

7.3 SLO and Surface Segmentation

Based on the second fundamental form of the surface, we generalize the definition of the STO, and introduce a new geometric operator named the SLO. Different from the LBO eigenfunctions, the SLO eigenfunctions are sensitive to concave creases/regions and convex ridges, providing a powerful basis for surface segmentation.

7.3.1 Generalized STO and SLO

We generalize Eq. (7.3) by replacing K with Ψ, and define a ***generalized STO (GSTO)***,

$$\begin{aligned} \blacklozenge f &= [\mathbf{x}_u, \mathbf{x}_v]\, \Psi \left[b^{\alpha\beta} \right] [f_u, f_v]^{\mathrm{T}} \\ &= g_u^{\blacklozenge} f_u + g_v^{\blacklozenge} f_v, \end{aligned} \tag{7.13}$$

where

$$g_u^{\blacklozenge} = \frac{\Psi}{b} \left(b_{22} \mathbf{x}_u - b_{12} \mathbf{x}_v \right) \text{ and } g_v^{\blacklozenge} = \frac{\Psi}{b} \left(b_{11} \mathbf{x}_v - b_{12} \mathbf{x}_u \right).$$

Here we choose $\Psi = 1$, e^K or e^{-H} for different purposes in surface segmentation, see Sec. 7.3.2 for details. From Eq. (7.13), $\forall f, h \in C^1(\mathbf{S})$, we can derive that

$$\begin{aligned} \langle \blacklozenge f, \blacklozenge h \rangle &= [f_u, f_v]\, \Psi^2 \left[b^{\alpha\beta}\right]\left[g_{\alpha\beta}\right]\left[b^{\alpha\beta}\right] [h_u, h_v]^{\mathrm{T}} \\ &= [f_u, f_v]\, \mathbf{A}\, [h_u, h_v]^{\mathrm{T}}, \end{aligned} \tag{7.14}$$

where

$$\mathbf{A} = \Psi^2 \left[b^{\alpha\beta}\right]\left[g_{\alpha\beta}\right]\left[b^{\alpha\beta}\right] = \left[a_{\alpha\beta}\right],$$

with

$$\begin{aligned} a_{11} &= \frac{\Psi^2}{b^2}\left(b_{22}^2 g_{11} - 2b_{22}b_{12}g_{12} + b_{12}^2 g_{22}\right), \\ a_{12} &= -\frac{\Psi^2}{b^2}\left(b_{22}b_{12}g_{11} - \left(b_{11}b_{22} + b_{12}^2\right) g_{12} + b_{11}b_{12}g_{22}\right), \\ a_{22} &= \frac{\Psi^2}{b^2}\left(b_{11}^2 g_{22} - 2b_{11}b_{12}g_{12} + b_{12}^2 g_{11}\right). \end{aligned}$$

To detect curvature-related features, we define a new geometric operator based on the second fundamental form of the surface, namely the ***secondary Laplace operator (SLO)***. Given $f \in C^2(S)$, the SLO $\blacktriangle$ is defined implicitly which satisfies

$$\int_S \left(h \blacktriangle f + \langle \blacklozenge f, \blacklozenge h \rangle\right) \mathrm{dA} = 0, \tag{7.15}$$

$\forall h \in C^1 S$. The inner product of f and $\blacktriangle f$ is

$$\int_S (f \blacktriangle f)\, \mathrm{dA} = -\int_S \langle \blacklozenge f, \blacklozenge f \rangle\, \mathrm{dA} = -\int_S \left\| \blacklozenge f \right\| \mathrm{dA} \leq 0, \tag{7.16}$$

therefore the SLO is a semi-negative definite geometric operator. The SLO eigenfunctions should satisfy

$$\blacktriangle f = -\lambda_s f, \tag{7.17}$$

where λ_s are the eigenvalues. By plugging Eq. (7.17) into Eq. (7.15), we have

$$\int_S \langle \blacklozenge f, \blacklozenge h \rangle \, \mathrm{dA} = \lambda_s \int_S hf \mathrm{dA}. \tag{7.18}$$

Similar to Eq. (7.12), the eigenfunctions of SLO can be obtained by solving an eigenproblem

$$M^S W = \lambda_s C W, \tag{7.19}$$

where $M^S = \left[m_{ij}^S\right]$ and $m_{ij}^S = \int_S \left\langle \blacklozenge \varphi_i, \blacklozenge \varphi_j \right\rangle \mathrm{dA}$. Note that the explicit definition of the SLO is not required during the eigenfunction computation.

As defined using the second fundamental form of the surface, the SLO eigenfunctions are sensitive to curvature changes on the surface. By choosing different values for Ψ, different surface features such as concave creases/regions and convex ridges can be detected using the SLO eigenfunctions. In the following, we introduce three different choices of Ψ for surface segmentation.

7.3.2 Surface Segmentation

Due to the second fundamental form in the definition of SLO, its eigenfunctions reflect the curvature-related surface features, which provide a convenient basis for surface segmentation. Basically, surface segmentation consists of two main steps: computing eigenfunctions and mapping vertices onto a k-dimensional space using the first k modes; and clustering vertices into a series of groups or surface patches. In this chapter, point clustering is fulfilled using Prediction Analysis for Microarrays (PAM) [121]. The clustering result can be evaluated using the Davies-Bouldin value [114], which determines the optimal group number.

With different choices of Ψ, the SLO eigenfunctions tend to emphasize different types of surface features such as concave creases/regions and convex ridges. In this section, we use the mean curvature H and Gaussian curvature K to define Ψ. To avoid generating

singular matrices, we restrict

$$\Psi = \begin{cases} \psi_l & \text{if } \Psi < \psi_l \\ \psi_u & \text{if } \Psi > \psi_u \end{cases},$$

where ψ_l and ψ_u are the lower and upper bounds, respectively. Here we choose $\psi_l = 0.5$ and $\psi_u = 10$ in the computation. Three different values are adopted in this chapter:

1. $\Psi = 1$ for segmenting smooth faces with both concave creases and convex ridges. For example in Fig. 7.3, the three cylindrical components contain concave creases at the root and convex ridges at the top. When Ψ is a constant, the eigenfunction value varies intensively across both concave creases and convex ridges and stays similar in smooth regions surrounded by them, so each smooth region can be segmented as a patch. Since the choice of the constant value does not change the distribution of eigenfunctions, we simply choose $\Psi = 1$ in this chapter;

2. $\Psi = e^K$ for segmenting components with concave creases only. In many objects, components with physical meanings are usually connected with concave creases rather than convex ridges. For example in Fig. 7.4, the connection regions between the horn, four wheels and the body of the Elk model are all concave creases. These concave creases have negative K while convex ridges have positive K. In addition, the SLO eigenfunctions tend to be more sensitive to regions with a smaller Ψ, where the eigenfunction value varies intensively. For the regions with a larger Ψ, the eigenfunction value stays similar. Therefore, we choose $\Psi = e^K$, so the intensive variation of eigenfunctions only happens across concave creases and the regions surrounded by them are segmented as patches; and

3. $\Psi = e^{-H}$ for segmenting concave regions from smooth surfaces. Some concave regions we want to segment may not be surrounded by obvious concave creases or convex ridges, such as the three shallow concave regions in the Disk model in Fig. 7.3.

These concave regions have negative H, so we choose $\Psi = e^{-H}$ to set a large Ψ in them and a small Ψ for the remaining surface. As a result, the eigenfunction value stays similar inside the concave regions, and we segment each of them as a patch.

Note that each input mesh is scaled to make the maximal length unit in x, y and z directions. For a different scaling, the choice of Ψ needs to be adjusted accordingly. Fig. 7.3(b-d) shows the first three SLO eigenfunctions for the Disk model with three different choices of Ψ, and (i-l) are the corresponding segmentation results. When $\Psi = 1$ in (b, g), each smooth face is segmented as a patch, and each cylindrical component is separated into two patches (top face and circumferential face). On the contrary when $\Psi = e^{K}$ in (c, h), these cylindrical components are segmented as single patches because $\Psi = e^{K}$ cannot detect convex ridges. When $\Psi = e^{-H}$ in (d, i), three concave regions are detected and segmented by the SLO eigenfunctions. Since these regions are smoothly embedded in a planar region without any obvious concave creases or convex ridges, they are ignored by the eigenfunctions in (b) and (c). As shown in Fig. 7.3(e), the spectra of SLO are convex while the spectra of LBO tend to be flat. (a, f) show the LBO eigenfunctions and the corresponding segmentation result. It is obvious that the SLO eigenfunctions yield good segmentation results with concave and convex features detected, while the LBO eigenfunctions cannot.

Compared with the LBO eigenfunctions in Fig. 7.2(a), the SLO eigenfunctions in Fig. 7.4(a, b) are more sensitive to the variation of surface curvatures, which perform better in determining the patch boundaries in the segmentation results in (c, d). For a certain object, the choice of Ψ depends on specific purposes. For example when $\Psi = 1$, the horn of Elk is separated into two planar regions in (c), which may be desired for design or manufacturing. But for structure recognition, the result in (d) may be preferred because the horn is a single solid component.

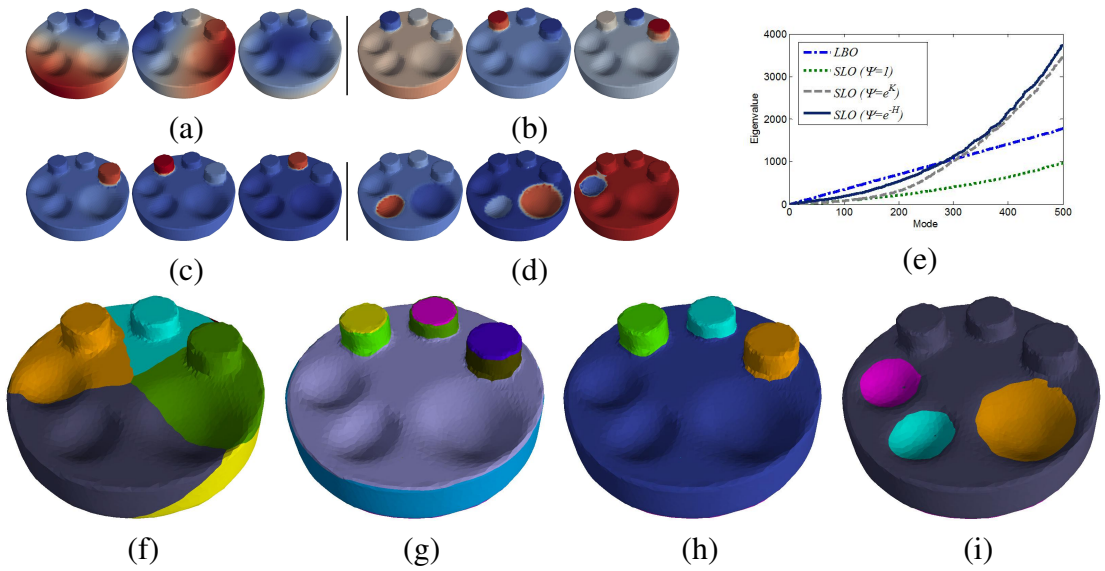


Figure 7.3: Disk model. (a) Modes 1-3 of the LBO eigenfunctions; (b-d) Modes 1-3 of the SLO eigenfunctions when $\Psi = 1$, $\Psi = e^{K}$ and $\Psi = e^{-H}$, respectively; (e) the spectra of LBO and SLO; and (f-i) the corresponding surface segmentation results of (a-d).

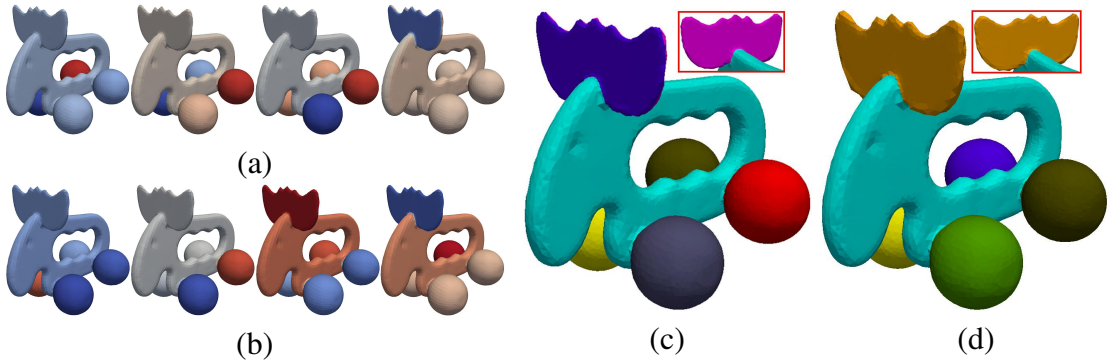


Figure 7.4: Elk model. (a, b) Modes 1-4 of the SLO eigenfunctions when $\Psi = 1$ and $\Psi = e^{K}$, respectively; and (c, d) the corresponding segmentation results of (a) and (b), respectively. The red windows in (c, d) show the back face of the horn.

7.4 GGHO and Geometric Flow

Analogy to the definition of GHO in Eq. (7.6), we introduce a ***generalized Giaquinta-Hildebrandt operator (GGHO)*** , which is defined as

$$\blacksquare f = div_S(\blacklozenge f). \tag{7.20}$$

Then we have

$$\blacksquare f = \frac{1}{\sqrt{g}}\left[\frac{\partial}{\partial u}, \frac{\partial}{\partial v}\right]\left[\sqrt{g}\,\Psi\left[b^{\alpha\beta}\right][f_u, f_v]^{\mathrm{T}}\right]. \tag{7.21}$$

Let $\mathbf{v} = (v_1, v_2, v_3)^T \in \mathbb{R}^3$ be a vector field, we define $\blacksquare\mathbf{v} = (\blacksquare v_1, \blacksquare v_2, \blacksquare v_3)^T \in \mathbb{R}^3$. Acting the GGHO on the coordinates of the surface point, we have

$$\begin{aligned}
\blacksquare\mathbf{x} &= \tfrac{1}{\sqrt{g}}\left(\left[\tfrac{\partial}{\partial u}, \tfrac{\partial}{\partial v}\right]\left[\sqrt{g}\,\Psi\left[b^{\alpha\beta}\right][\mathbf{x}_u, \mathbf{x}_v]^{\mathrm{T}}\right]\right)^T \\
&= \tfrac{1}{\sqrt{g}}\left(\left[\tfrac{\partial}{\partial u}, \tfrac{\partial}{\partial v}\right]\tfrac{\Psi}{K}\left[\sqrt{g}K\left[b^{\alpha\beta}\right][\mathbf{x}_u, \mathbf{x}_v]^{\mathrm{T}}\right]\right)^T \\
&= \tfrac{\Psi}{K}\tfrac{1}{\sqrt{g}}\left(\left[\tfrac{\partial}{\partial u}, \tfrac{\partial}{\partial v}\right]\left[\sqrt{g}K\left[b^{\alpha\beta}\right][\mathbf{x}_u, \mathbf{x}_v]^{\mathrm{T}}\right]\right)^T + \\
&\quad \left(\left[\tfrac{\partial}{\partial u}\left(\tfrac{\Psi}{K}\right), \tfrac{\partial}{\partial v}\left(\tfrac{\Psi}{K}\right)\right]\left[K\left[b^{\alpha\beta}\right][\mathbf{x}_u, \mathbf{x}_v]^{\mathrm{T}}\right]\right)^T \\
&= \tfrac{\Psi}{K}\square\mathbf{x} + \diamond\left(\tfrac{\Psi}{K}\right) = 2\Psi\mathbf{n} + \diamond\left(\tfrac{\Psi}{K}\right),
\end{aligned} \tag{7.22}$$

where $\mathbf{n}$ is the normal. A geometric flow can be defined as

$$\frac{\partial\mathbf{x}}{\partial t} = \blacksquare\mathbf{x} + \lambda\mathbf{v}_T, \tag{7.23}$$

where $\mathbf{v}_T$ is a vector in the tangential direction regularizing the quadrilateral elements, and λ controls the strength of regularization (e.g. $\lambda = 0.05$). $\mathbf{v}_T$ is usually defined based on the geometric center of the 1-ring neighborhood of a node, but features might be blurred in this way. In this chapter, we define an anisotropy in the 1-ring neighborhood based on the principal curvature directions, and $\mathbf{v}_T$ is an weighted average of the neighbors. Let Node j be a neighbor of Node i, and κ_j^1 and κ_j^2 be the principal curvature directions. The feature direction κ_j is defined as the larger one between κ_j^1 and κ_j^2. The weight of Node j is defined as

$$w_{ij} = \left|\left\langle\kappa_i, \kappa_j\right\rangle \cdot \left\langle\mathbf{n}_i, \mathbf{n}_j\right\rangle\right| + \epsilon_A, \tag{7.24}$$

where ϵ_A is a predefined parameter and here we choose $\epsilon_A = 0.05$ in our computation. We obtain similar weights for the neighbors with a weak anisotropy, such as the flat regions.

We have

$$\mathbf{v}_T = \frac{\sum\limits_{j\in N(i)} w_{ij}\mathbf{x}_j}{\sum\limits_{j\in N(i)} w_{ij}} - \mathbf{x}_i, \tag{7.25}$$

where $N(i)$ is the 1-ring neighborhood of Node i. With Eq. (7.24), large weights are assigned to the neighbors located along the same concave or convex edge with Node i. Nodes across concave or convex edges have little influence to each other. If Node i is not on the sharp edge, $\mathbf{v}_T$ will be the geometric center of its 1-ring neighbors. $\blacksquare\mathbf{x}$ consists of two components in the normal and tangential directions, respectively.

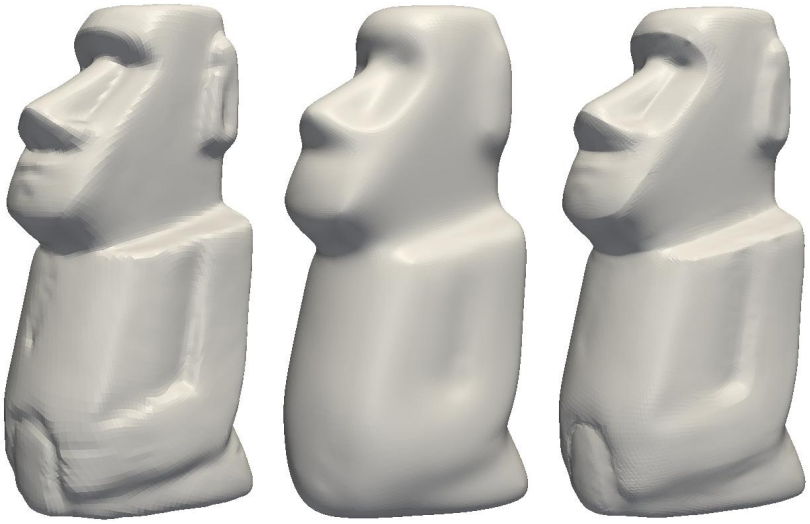
To strengthen concave creases and convex ridges on the surface, we choose $\Psi = \frac{HK^2}{\sqrt{K^2+\delta^2}}$ in this chapter. In this way, these creases and ridges can be characterized by the sign of ψ, negative for concave creases and positive for convex ridges. Then we have

$$\blacksquare\mathbf{x} = \frac{2HK^2}{\sqrt{K^2+\delta^2}} \cdot \mathbf{n} + \diamond\left(H\frac{K}{\sqrt{K^2+\delta^2}}\right), \tag{7.26}$$

where δ is a small constant (e.g. $\delta = 0.1$). Since GGHO is defined based on the second fundamental form of the surface, it is more sensitive to the curvature-related features compared with LBO, which is defined based on the first fundamental form. Fig. 7.5 shows the smoothing results for the Moai model based on the LBO and GGHO. Compared with LBO, some sharp edges such as the lower jaw and the eyebrow are preserved and sharpened using GGHO.

7.5 Results and Discussion

In this section, various models are tested using our surface segmentation and geometric flow methods. All the results are generated using a computer with an Intel Xeon E5-1620 CPU, a Nvidia GeForce GTX680 graphic card, and 16GB of memory. Tab. **??** shows a summary of the computation of eigenfunctions and segmentation results for LBO and SLO.



(a) (b) (c)

Figure 7.5: Smoothing results of geometric flow for the Moai model using different geometric operators. (a) The original model; (b) the result from LBO; and (c) the result from the GGHO (Iteration: 50; Step size: 0.02).

Surface segmentation using $\Psi = e^K$. For objects with only concave creases, we choose $\Psi = e^K$. To reveal the differences between the SLO and other operators, we adopt the PAM clustering method to compute the segmentation results from the eigenfunctions of LBO, concavity-aware Laplacian (CL) [127] and SLO. Triangle meshes are generated for the CL eigenfunction computation by splitting each quadrilateral element into two triangles. Since the vertex number is not changed, the computational time for the CL is similar with LBO and SLO. The segmentation results can be evaluated quantitatively using the Princeton benchmark and software [17], by comparing them with the ground truth from human. There are four metrics [17] to measure the quality of segmentation results:

1. the cut discrepancy e_D, which is the sum of distance between points along the cuts in the computed segmentation to the closest cuts in the ground truth segmentation, and vice-versa;

2. the Hamming distance e_H, which measures the overall difference between the patches in different segmentation results;

3. the Rand index e_R, which measures the likelihood that a random pair of elements are in the same patch or not in different segmentation results; and

4. the consistency error e_C, which measures the hierarchical similarities and differences of different segmentation results.

Basically, the smaller these metrics are, the better the segmentation is. Figs. 7.6 shows eigenfunctions and segmentation results from the LBO, CL and our SLO, together with the segmentation result from the shape diameter function (SDF) method [110]. The metrics shown in the figures indicate that the SLO yields the best segmentation results. Concave creases are completely ignored by the LBO eigenfunctions, so the resulting segmentation results miss the connection regions of different components, see (e). On the base of the Bust model, all the patches shown in the ground truth in (d) can be obtained using the SLO eigenfunctions as shown in (g), while some of them are missed in (f) and none is captured in (e). Compared with [127], we obtain different segmentation results for the Bust using the CL eigenfunctions. This is because here we use the PAM clustering for segmentation, while in [127] an advanced segmentation method named the single segmentation field was adopted. Using the SLO eigenfunctions, we obtain a lower e_R for the Bust model (0.116 vs 0.286).

Protein pocket detection using $\Psi = e^{-H}$. Protein pockets are concave regions on the biomolecular surface, which determine the interactions between the protein and other molecules. Detecting and segmenting these pockets are essentially important to predict the behavior of biomolecules during the biological process. Since the pockets are concave regions on the smooth protein surface, we adopt $\Psi = e^{-H}$ to detect and segment them.

Fig. 7.7 shows the results of 1BYH, a protein widely used for enzyme behavior study. (a, b) show the eigenfunctions of LBO and SLO, respectively. Their segmentation results are shown in (c, d). It is obvious that the SLO eigenfunctions segment the concave and convex regions well, while the LBO eigenfunctions cannot. In (d), we choose the patch

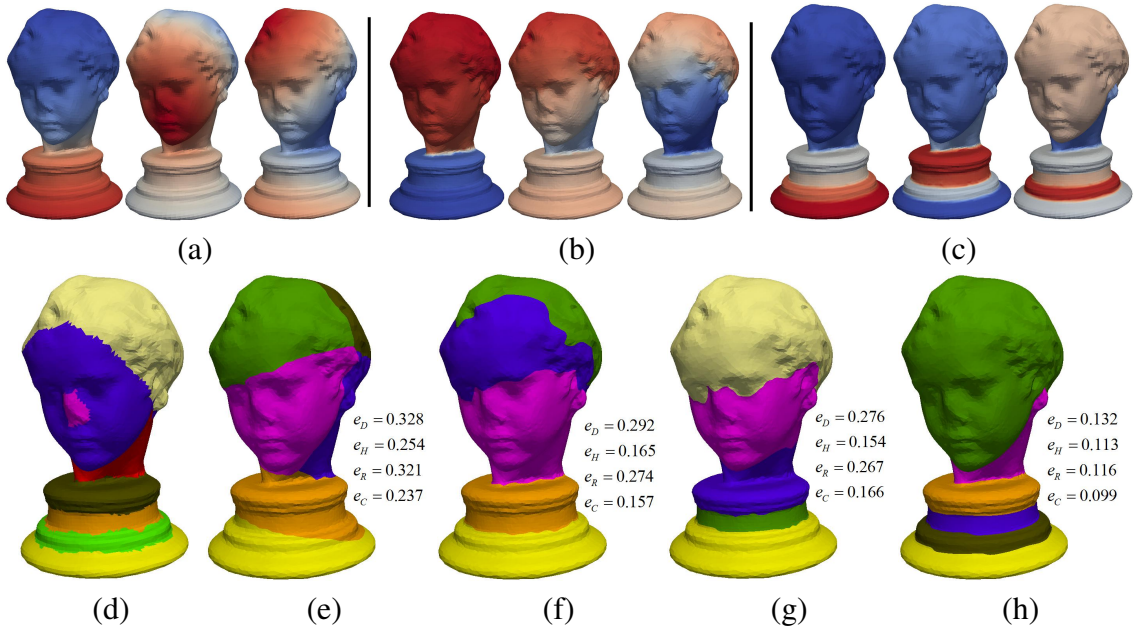


Figure 7.6: The eigenfunctions and segmentation results of the Bust model. (a-c) Modes 1-3 of the LBO, CL and SLO, respectively; (d) the ground truth for segmentation [17]; (e, f) segmentation results from LBO and CL eigenfunctions; (g) the result from the SDF method; and (h) the result from SLO eigenfunctions.

with negative average mean curvature as the pocket (Patch D), which is further subdivided into two smaller patches in (e) when two more eigenfunctions are considered.

The protein pockets are candidates for *active sites* where the protein binds with other molecules. We can use the bound structure to determine the real active sites, and also use them to validate our segmentation results. Fig. 7.7(f) shows a known binding between 1BYH and a small molecule BETA-D-GLUCOSE. Let $\mathbf{x}_j$ be a node on the protein surface, the Gaussian density at $\mathbf{x}_j$ can be computed using a summation of Gaussian kernel functions [143, 75],

$$G(\mathbf{x}_j) = \sum_{i=1}^{M} e^{\kappa\left(\|\mathbf{x}_i - \mathbf{x}_j\|^2 - r_i^2\right)}, \tag{7.27}$$

where M is the atom number of the small molecule, κ is the decay rate, and $(\mathbf{x}_i, r_i)$ are the center and radius of Atom i. The average Gaussian density in a pocket can be computed as $\bar{G} = \frac{1}{N_P} \sum_{j=1}^{N_P} G(\mathbf{x}_j)$, where N_P is the vertex number in the pocket. The pocket with the largest average Gaussian density is identified as the active site, see the orange patch in Fig. 7.7(g).

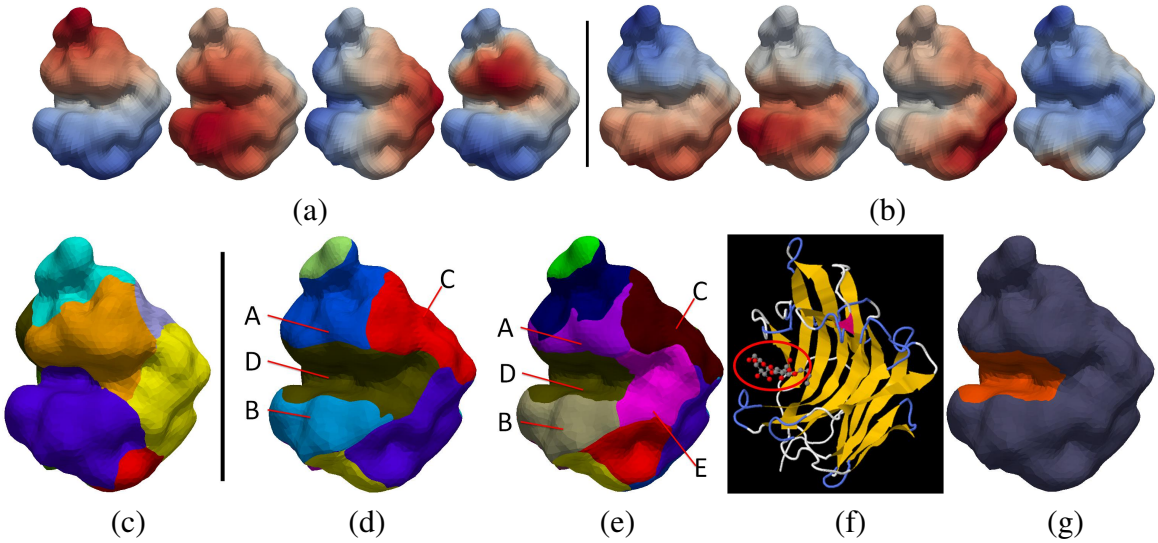


Figure 7.7: The first four eigenfunctions and segmentation results for the 1BYH model from the LBO and SLO with $\Psi = e^{-H}$. (a, b) Modes 1-4 eigenfunctions of the LBO and SLO; (c) the segmentation result from Modes 1-4 of the LBO; (d, e) segmentation results from Modes 1-2 and 1-4 of the SLO; (f) a known binding between 1BYH and BETA-D-GLUCOSE (red circle); and (g) the active site (orange) corresponding to Patch D in (e).

Fig. 7.8 shows the segmentation results for another polymorphic enzyme named 1C2B. According to the combined structure, the red patch in (e-f) is detected as an active site.

Limitations. Since the SLO and GGHO are defined based on the second fundamental form, a high-order representation of the surface is required to compute the eigenfunctions of SLO and perform the GGHO-based smoothing, which are more complicated in computation compared with other LBO-based methods. In this chapter, we use the Catmull-Clark basis functions and quadrilateral control meshes to represent the surface, so a remeshing process is required for other forms of input meshes. The cross field-based surface parameterization may fail in generating quadrilateral meshes when the input triangle meshes contain many complicated features, which limits the applications of our method.

7.6 Conclusion and Future Work

In this chapter, two new geometric operators, namely the secondary Laplace operator (SLO) and generalized Giaquinta-Hildebrandt operator (GGHO), have been introduced based on

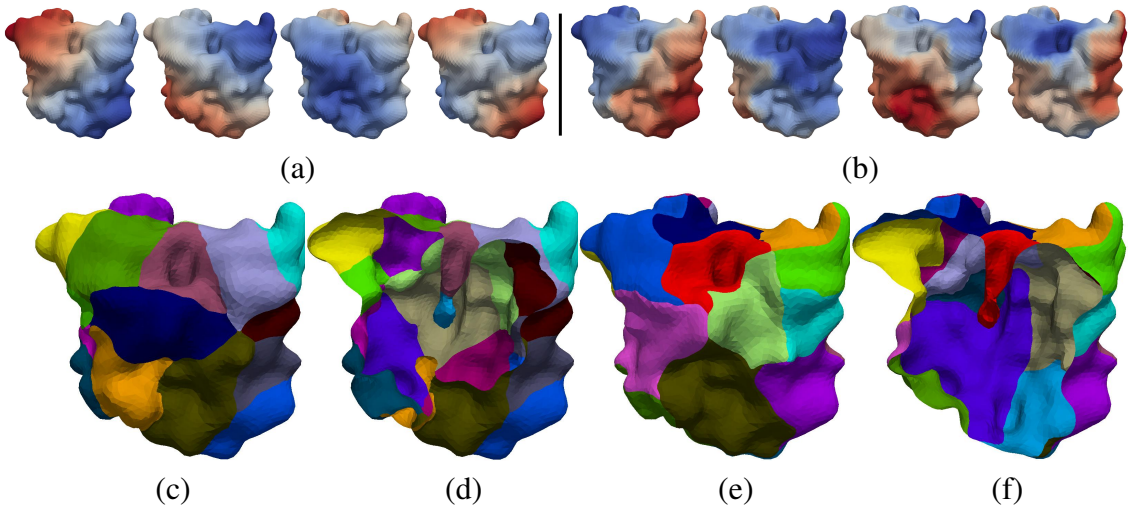


Figure 7.8: The eigenfunctions and segmentation results for the the MAChE (1C2B) model from the LBO and SLO with $\Psi = e^{-H}$. (a, b) Modes 1-4 eigenfunctions of the LBO and SLO; (c) the segmentation result from Modes 1-4 of the LBO; (e) the segmentation result from Modes 1-4 of the SLO; and (d, f) the crosssections of (c) and (e), respectively.

the second fundamental form of the surface. The eigenfunctions of SLO are sensitive to curvature-related surface features, which segment the surface with concave creases/regions and convex ridges detected. Based on the GGHO, a new geometric flow method has been developed for surface smoothing, which preserves and strengthens sharp features on the surface.

In this chapter, surface segmentation is introduced mainly to show the special properties of the SLO eigenfunctions. In the future, we plan to develop more robust segmentation algorithms and explore other applications of the SLO eigenfunctions. On the other hand, since most of the surfaces are still represented by triangle meshes, we also plan to investigate the implementation of these new operators on triangle meshes.

Chapter 8

Conclusion and Future Work

8.1 Conclusions

In this thesis, an efficient geometric modeling approach has been developed for the biomolecular complexes. The biomolecular surface is represented by a Gaussian density map, where the local resolution control is enabled. The computational efficiency for the construction of Gaussian density map is improved using a combination of neighboring search, KD-tree structure and bounding volume hierarchy. Besides, an error-bounded atom simplification method is designed to speedup the computation by reducing the atom number. Moreover, CPU- and GPU-assisted parallel computation techniques are used to further improve the computational speed. With a quality improvement process, high-quality triangle and tetrahedral meshes can be generated quickly for large biomolecular complexes.

To enable the quadrilateral mesh generation, a surface parameterization method has been developed based on the cross filed. A novel approach has been developed to define the guidance directions for the construction of cross field. Two methods are designed to combine the gradient information from different eigenfunctions of the Laplace-Beltrami operator to capture the main structure of an object. As a result, the parametric lines are

aligned with the structural features at multiple scales. Besides, an anisotropy is enabled by adapting the cross field to non-uniform parametric line spacings. Addition to the quadrilateral mesh, surface parameterization can also be used for T-mesh generation, which provides an efficient tool in representing the multi-resolution biomolecular surfaces. An extended cross field-based parameterization is introduced to generate adaptive and anisotropic T-meshes, which can be used further for T-spline surface construction. In addition, a new gradient flow-based method is introduced for T-mesh quality improvement, preserving the anisotropy in the input T-mesh.

Besides geometric modeling, eigenfunctions also provide a powerful tool for the shape analysis of biomolecules. A new shape correspondence analysis method has been introduced based on the volumetric eigenfunctions from the joint graph of different biomolecular shapes. A two-step method has been developed to improve the shape correspondence, which eliminates the node assignments against the main structure and preserves the local neighborhood of each node by minimizing the distortion energy in the correspondence-based deformation. Based on the shape correspondence analysis, a shape approximation and prediction, and shape comparison method are designed, which can be used for analyzing the behavior of biomolecules based on the variation of shape geometry.

Geometric operators have been applied in various operations for biomolecular surfaces. Two new geometric operators, namely the secondary Laplace operator (SLO) and generalized Giaquinta-Hildebrandt operator (GGHO), have been introduced based on the second fundamental form of the surface. The eigenfunctions of SLO are sensitive to curvature-related surface features, which segment the surface with concave creases/regions and convex ridges detected. Based on the GGHO, a new geometric flow method has been developed for surface smoothing, which preserves and strengthens sharp features on the surface.

8.2 Future Work

In the future, it is worth to apply our efficient multiscale modeling methods to various applications of biomolecules. Based on our T-mesh generation, T-spline representations can be built for biomolecular surfaces, enabling the isogeometric analysis. Another future direction can be solving the Poisson-Boltzmann equation using the high-order basis of the T-spline surface.

Some of our geometric modeling and shape analysis algorithms are designed for general objects, instead of for biomolecules only. Since our methods are flexible, it is easy to further enhance them specifically for biomolecules by including more biochemical information. For example for the anisotropic surface parameterization and T-mesh generation, the alignment of parametric lines can be guided by various scalar fields such as the distribution of electrostatic potential, which improves the computational accuracy for specific studies. For the shape correspondence analysis, important chains can be used to define the initial correspondence, leading to more realistic deformation tracking and shape comparison results. Additionally, the electrostatic potential can be combined with the SLO eigenfunctions to improve the accuracy of detecting and segmenting protein pockets. Moreover, during shape approximation, physical constraints such as pure bending can be set in deforming the biomolecular shapes.

Analogy to the LBO eigenfunctions which describe the stationary wave on the surface, the SLO eigenfunctions may also correspond to a special physical phenomenon. Studying the physical meaning of the SLO eigenfunctions can help us better understand the nature of this new geometric operator and further explore its applications. Furthermore compared with quadrilateral meshes, triangle meshes are more commonly used for surface representation. It is worth to study the computation of SLO eigenfunctions over the Loop's subdivision surfaces.

Bibliography

[1] L. Albou, B. Schwarz, O. Poch, J. Wurtz, and D. Moras. Defining and characterizing protein surface using alpha shapes. *Proteins*, 76(1):1–12, 2009.

[2] S. Artemova, S. Grudinin, and S. Redon. A comparison of neighbor search algorithms for large rigid molecules. *Journal of Computational Chemistry*, 32(13):2865–2877, 2011.

[3] C. L. Bajaj, J. Castrillon-Candas, V. Siddavanahalli, and Z. Xu. Compressed representations of macromolecular structures and properties. *Structure*, 13:463–471, 2005.

[4] C. L. Bajaj, V. Pascucci, and D. Schikore. Seed sets and search structures for optimal isocontour extraction. Technical report, Texas Institute of Computational and Applied Mathematics, 1999.

[5] C. L. Bajaj, V. Pascucci, A. Shamir, R. J. Holt, and A. N. Netravali. Multiresolution molecular shapes. Technical report, TICAM Technical Report, 1999.

[6] C. L. Bajaj, V. Pascucci, A. Shamir, R. J. Holt, and A. N. Netravali. Dynamic maintenance and visualization of molecular surfaces. *Discrete Applied Mathematics*, 127(1):23–51, 2003.

[7] C. L. Bajaj and V. Siddavanahalli. Fast error-bounded surfaces and derivatives computation for volumetric particle data. Technical report, ICES 06-03, 2006.

[8] Y. Bazilevs, V. Calo, J. Cottrell, J. Evans, T. Hughes, S. Lipton, M. Scott, and T. Sederberg. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5–8):229–263, 2010.

[9] M. Belkin, J. Sun, and Y. Wang. Discrete Laplace operator on meshed surfaces. *Symposium on Computational Geometry*, pages 278–287, 2008.

[10] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[11] D. Bommes, M. Campen, H. Ebke, P. Alliez, and L. Kobbelt. Integer-grid maps for reliable quad meshing. *ACM Transactions on Graphics*, 32(4):98:1–98:12, 2013.

[12] D. Bommes, B. Lévy, N. Pietroni, E. Puppo, C. Silva, and D. Zorin. Quad-mesh generation and processing: a survey. *Computer Graphics Forum*, 32(6):51–76, 2013.

[13] D. Bommes, H. Zimmer, and L. Kobbelt. Mixed-integer quadrangulation. *ACM Transactions on Graphics*, 28(3):1–10, 2009.

[14] D. Bommes, H. Zimmer, and L. Kobbelt. Practical mixed-integer optimization for geometry processing. In *Curves and Surfaces*, pages 193–206. Springer, 2012.

[15] M. Brovka, J. I. López, J. M. Escobar, J. M. Cascón, and R. Montenegro. A new method for T-spline parameterization of complex 2D geometries. *Engineering with Computers*, 30(4):457–473, 2014.

[16] M. Carcassoni and E. Hancock. Correspondence matching with modal clusters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(12):1609–1615, 2003.

[17] X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics*, 28(3):73–84, 2009.

[18] Y. Cheng, C. A. Chang, Z. Yu, Y. Zhang, M. Sun, T. S. Leyh, M. J. Holst, and J. A. Mccammon. Diffusional channeling in the sulfate activating complex: combined continuum modeling and coarse-grained Brownian dynamics studies. *Biophysical Journal*, 95(10):4659–4667, 2008.

[19] F. Chung. *Spectral graph theory*. American Mathematical Soc., 1997.

[20] D. Cohen-Steiner and J. Morvan. Restricted delaunay triangulations and normal cycle. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 312–321. ACM, 2003.

[21] M. L. Connolly. Analytical molecular surface calculation. *Journal of Applied Crystallography*, 16(5):548–558, 1983.

[22] M. L. Connolly. *Molecular surface: A Review*. *Network Science*, 1996.

[23] K. Crane, U. Pinkall, and P. Schröder. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics*, 32(4):61–72, 2013.

[24] J. P. D'Amato and M. Vénere. A CPU-GPU framework for optimizing the quality of large meshes. *Journal of Parallel and Distributed Computing*, (0):–, 2013.

[25] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Annual Conference on Computer Graphics*, pages 317–324, 1999.

[26] S. Dong, P. Bremer, M. Garland, V. Pascucci, and J. C. Hart. Spectral surface quadrangulation. *ACM Transactions on Graphics*, 25(3):1057–1066, 2006.

[27] S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer aided geometric design*, 22(5):392–423, 2005.

[28] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[29] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In *Advances in multiresolution for geometric modelling*, pages 157–186. Springer, 2005.

[30] R. Fonseca and P. Winter. Bounding volumes for proteins: a comparative study. *Journal of Computational Biology*, 19(10):1203 – 1213, 2012.

[31] M. Frigo and S. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[32] Z. Gao, Z. Yu, and X. Pang. A compact shape descriptor for triangular surface meshes. *Computer-Aided Design*, 53:62–69, 2014.

[33] R. Garimella, M. Shashkov, and P. Knupp. Triangular and quadrilateral surface mesh quality optimization using local parametrization. *Computer Methods in Applied Mechanics and Engineering*, 193(9):913–928, 2004.

[34] W. Geng and F. Jacob. A GPU-accelerated direct-sum boundary integral Poisson-Boltzmann solver. *Computer Physics Communications*, 184:1490–1496, 2013.

[35] W. Geng and R. Krasny. A treecode-accelerated boundary integral Poisson-Boltzmann solver for electrostatics of solvated biomolecules. *Journal of Computational Physics*, 247:62–87, 2013.

[36] W. Geng and S. Zhao. Fully implicit ADI schemes for solving the nonlinear Poisson-Boltzmann equation. *Molecular Based Mathematical Biology*, 1:109–123, 2013.

[37] J. Giard and B. Macq. Molecular surface mesh generation by filtering electron density map. *International Journal of Biomedical Imaging*, pages 263–269, 2010.

[38] J. A. Grant and B. T. Pickup. A Gaussian description of molecular shape. *Journal of Physical Chemistry*, 99(11):3503–3510, 1995.

[39] I. Guskov. An anisotropic mesh parameterization scheme. *Engineering with Computers*, 20(2):129–135, 2004.

[40] Y. He, K. Wang, H. Wang, X. Gu, and H. Qin. Manifold T-spline. In *Proceedings of Geometric Modeling and Processing*, pages 409–422, 2006.

[41] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 203–212, 2001.

[42] K. Hildebrandt, C. Schulz, C. von Tycowicz, and K. Polthier. Modal shape analysis beyond Laplacian. *Computer Aided Geometric Design*, 29(5):204–218, 2012.

[43] M. Holst, N. Baker, and F. Wang. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation algorithms I: algorithms and examples. *Journal of Computational Chemistry*, 21:1319–1342, 2000.

[44] L. Hu, D. Chen, and G. Wei. High-order fractional partial differential equation transform for molecular surface construction. *Molecular Based Mathematical Biology*, 1:1–25, 2013.

[45] J. Huang, M. Zhang, J. Ma, X. Liu, L. Kobbelt, and H. Bao. Spectral quadrangulation with orientation and alignment control. *ACM Transactions on Graphics*, 27(5):147:1–147:9, 2008.

[46] T. Hughes, J. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39–41):4135–4195, 2005.

[47] V. Jain and H. Zhang. Robust 3D shape correspondence in the spectral domain. In *Shape Modeling and Applications 2006*, pages 19–32, 2006.

[48] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of Hermite data. *SIGGRAPH*, 21:339–346, 2002.

[49] B. Jüttler, M. Kapl, D. Nguyen, Q. Pan, and M. Pauley. Isogeometric segmentation: the case of contractible solids without non-convex edges. *Computer-Aided Design*, 57(0):74 – 90, 2014.

[50] O. Van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or. A survey on shape correspondence. *Computer Graphics Forum*, 30(6):1681–1707, 2011.

[51] F. Kälberer, M. Nieser, and K. Polthier. Quadcover-surface parameterization using branched coverings. In *Computer Graphics Forum*, volume 26, pages 375–384, 2007.

[52] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[53] C. Keenan, D. Mathieu, and S. Peter. Trivial connections on discrete surfaces. *Computer Graphics Forum (SGP)*, 29(5):1525–1533, 2010.

[54] B. Kim, K. J. Kim, and J. K. Seong. GPU accelerated molecular surface computing. *Appl. Math*, 6(1S):185S–ÍC194S, 2012.

[55] D.-S. Kim, J.-K. Kim, Y Cho, and C.-M. Kim. Querying simplexes in quasi-triangulation. *Computer-Aided Design*, 44(2):85 – 98, 2012.

[56] D.-S. Kim, J. Seo, D. Kim, J. Ryu, and C.-H. Cho. Three-dimensional beta shapes. *Computer-Aided Design*, 38(11):1179–1191, 2006.

[57] L. Kim, K. Lee, and E. Rhee. α-Gauss curvature flows with flat sides. *Journal of Differential Equations*, 254(3):1172–1192, 2013.

[58] D. Kovacs, A. Myles, and D. Zorin. Anisotropic quadrangulation. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, pages 137–146, 2010.

[59] P. Laug and H. Borouchaki. Molecular surface modeling and meshing. *Engineering with Computers*, 18:199–210, 2002.

[60] H.-C. Lee, T. Liao, Y. Zhang, and G. Yang. Shape factor analysis: Structure-preserving dimension reduction on biological shape spaces. *Bioinformatics, submitted*, 2015.

[61] H.-C. Lee and G. Yang. Integrating dimension reduction with mean-shift clustering for biological shape classification. In *IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, pages 254–257, 2014.

[62] M. S. Lee, M. Feig, F. R. Salsbury, and C. L. Brooks. New analytic approximation to the standard molecular volume definition and its application to generalized born calculations. *Journal of Computational Chemistry*, 24(14):1348–1356, 2003.

[63] J. Leng, G. Xu, Y. Zhang, and J. Qian. *Quality improvement of segmented hexahedral meshes using geometric flows, Image-Based Geometric Modeling and Mesh Generation*. Springer, 2013.

[64] J. Leng, Y. Zhang, and G. Xu. A novel geometric flow-driven approach for quality improvement of segmented tetrahedral meshes. *Computer-Aided Design*, 45:1182–1197, 2013.

[65] B. Lévy. Laplace-Beltrami eigenfunctions: towards an algorithm that understands geometry. In *IEEE International Conference on Shape Modeling and Applications*, pages 13–21, 2006.

[66] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics*, 21(3):362–371, 2002.

[67] B. Lévy and B. Vallet. Spectral geometry processing with manifold harmonics. *Computer Graphics Forum*, 2(27):251–260, 2008.

[68] W. Li and S. McMains. Voxelized Minkowski sum computation on the GPU with robust culling. *Computer-Aided Design*, 43(10):1270 – 1283, 2011.

[69] W. Li, N. Ray, and B. Lévy. Automatic and interactive mesh to T-spline conversion. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, pages 191–200, 2006.

[70] Z. Li, S. Qin, Z. Yu, and Y. Jin. Skeleton-based shape analysis of protein models. *Journal of Molecular Graphics and Modelling*, 53:72–81, 2014.

[71] Z. Lian, A. Godil, B. Bustos, M. Daoudi, J. Hermans, S. Kawamura, Y. Kurita, G. Lavoué, H. Van Nguyen, and R. Ohbuchi. A comparison of methods for non-rigid 3D shape retrieval. *Pattern Recognition*, 46(1):449–461, 2013.

[72] T. Liao, X. Li, G. Xu, and Y. Zhang. Secondary Laplace operator and generalized Giaquinta-Hildebrandt operator with applications on surface segmentation and smoothing. *A Special Issue of SIAM Conference on Geometric & Physical Modeling in Computer Aided Design, doi: 10.1016/j.cad.2015.07.009*, 2015.

[73] T. Liao, W. Wang, and Y. Zhang. Adaptive and anisotropic T-mesh generation from cross field. In *Workshop on Structured Meshing: Theory, Application and Evaluation, the 27th Conference on Computer Animation and Social Agents (CASA 2014), Houston, TX. May 26-28*.

[74] T. Liao, G. Xu, and Y. Zhang. Structure-aligned guidance estimation in surface parameterization using eigenfunction-based cross field. *Graphical Models*, 76:691–705, 2014.

[75] T. Liao, Y. Zhang, P. Kekenes-Huskey, Y. Cheng, A. Michailova, A.D. McCulloch, M. Holst, and J.A. McCammon. Multi-core CPU or GPU-accelerated multiscale modeling for biomolecular complexes. *Molecular Based Mathematical Biology*, 1:164–179, 2013.

[76] R. Ling, J. Huang, F. Sun, B. Juttler, H. Bao, and W. Wang. Spectral quadrangulation with boundary conformation. Technical report, TR-2011-13, The University of Hong Kong, 2011.

[77] A. Liu and B. Joe. Relationship between tetrahedron shape measures. *BIT Numerical Mathematics*, 34:268–287, 1994.

[78] R. Liu and H. Zhang. Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum*, 26(3):385–394, 2007.

[79] H. Lombaert, L. Grady, J. Polimeni, and F. Cheriet. Fast brain matching with spectral correspondence. In *Information Processing in Medical Imaging*, pages 660–673, 2011.

[80] H. Lombaert, L. Grady, J. Polimeni, and F. Cheriet. Focusr: Feature oriented correspondence using spectral regularization – a method for precise surface matching. *Pattern Analysis and Machine Intelligence*, 35(9):2143–2160, 2013.

[81] H. Lombaert, J. Sporring, and K. Siddiqi. Diffeomorphic spectral matching of cortical surfaces. In *Information Processing in Medical Imaging*, pages 376–389, 2013.

[82] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. *SIGGRAPH*, 21(4):163–169, 1987.

[83] I. Lotan, F. Schwarzer, D. Halperin, and J. Latombe. Algorithm and data structures for efficient energy maintenance during Monte Carlo simulation of proteins. *Journal of Computational Biology*, 11(5):902 – 932, 2004.

[84] B. Lu, X. Cheng, and J. A. McCammon. "New-version-fast-multipole-method" accelerated electrostatic interactions in biomolecular systems. *Journal of Computational Physics*, 226:1348–1366, 2007.

[85] M. Marinov and L. Kobbelt. Automatic generation of structure preserving multiresolution models. In *Computer Graphics Forum*, volume 24, pages 479–486, 2005.

[86] D. Mateus, R. Horaud, D. Knossow, F. Cuzzolin, and E. Boyer. Articulated shape matching using Laplacian eigenfunctions and unsupervised point registration. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[87] D. Mateus, R. Horaud, D. Knossow, F. Cuzzolin, and E. Boyer. Articulated shape matching using Laplacian eigenfunctions and unsupervised point registration. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[88] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, pages 113–134, 2002.

[89] A. Myles, N. Pietroni, D. Kovacs, and D. Zorin. Feature-aligned T-meshes. *ACM Transactions on Graphics*, 29(4):98:1–98:12, 2010.

[90] A. Myles and D. Zorin. Global parametrization by incremental flattening. *ACM Transactions on Graphics*, 31(4):109:1–109:11, 2012.

[91] A. Myles and D. Zorin. Controlled-distortion constrained global parametrization. *ACM Transactions on Graphics*, 32(4):105:1–105:13, 2013.

[92] M. Niethammer, M. Reuter, F. Wolter, S. Bouix, N. Peinecke, M. Koo, and M. Shenton. Global medical shape analysis using the Laplace-Beltrami spectrum. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI*, pages 850–857. 2007.

[93] D. Panozzo, Y. Lipman, E. Puppo, and D. Zorin. Fields on symmetric surfaces. *ACM Transactions on Graphics*, 31(4):111:1–111:12, 2012.

[94] S. Pavanaskar and S. McMains. Filling trim cracks on GPU-rendered solid models. *Computer-Aided Design*, 45(2):535 – 539, 2013.

[95] N. Pietroni, M. Tarini, and P. Cignoni. Almost isometric mesh parameterization through abstract domains. *Visualization and Computer Graphics*, 16(4):621–635, 2010.

[96] N. Pietroni, M. Tarini, O. Sorkine, and D. Zorin. Global parametrization of range image sets. *ACM Transactions on Graphics*, 30(6):149:1–149:10, 2011.

[97] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental mathematics*, 2(1):15–36, 1993.

[98] H. Zhang R. Liu. Segmentation of 3D meshes through spectral clustering. In *Pacific Conference on Computer Graphics and Applications*, pages 298–305, 2004.

[99] N. Ray, W. Li, B. Lévy, A. Sheffer, and P. Alliez. Periodic global parameterization. *ACM Transactions on Graphics*, 25(4):1460–1485, 2006.

[100] M. Reuter. *Laplace Spectra for Shape Recognition*. Books on Demand, 2006.

[101] M. Reuter. Hierarchical shape segmentation and registration via topological features of Laplace-Beltrami eigenfunctions. *International Journal of Computer Vision*, 89(2-3):287–308, 2010.

[102] M. Reuter, S. Biasotti, and D. Giorgi. Discrete Laplace-Beltrami operators for shape analysis and segmentation. *Computers and Graphics*, 33(3):381–390, 2009.

[103] M. Reuter, F. Wolter, and N. Peinecke. Laplace-spectra as fingerprints for shape matching. In *Proceedings of the ACM symposium on solid and physical modeling*, pages 101–106, 2005.

[104] M. Reuter, F. Wolter, and N. Peinecke. Laplace-Beltrami spectra as "Shape-DNA" of surfaces and solids. *Computer-Aided Design*, 38(4):342–366, 2006.

[105] J. Ryu, R. Park, and D.-S. Kim. Molecular surfaces on proteins via beta shapes. *Computer-Aided Design*, 39(12):1042–1057, 2007.

[106] M. F. Sanner, A. J. Olson, and J. C. Spehner. Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers*, 38(3):305–20, 1996.

[107] L. Schmitz, L. F. Scheidegger, D. K. Osmari, C. A. Dietrich, and J. L. D. Comba. Efficient and quality contouring algorithms on the GPU. *Computer Graphics Forum*, 29:2569 – 2578, 2010.

[108] Guy. Scott and H. C. Longuet-Higgins. An algorithm for associating the features of two images. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 244(1309):21–26, 1991.

[109] J.-K. Seong, N. Baek, and K.-J. Kim. Real-time approximation of molecular interaction interfaces based on hierarchical space decomposition. *Computer-Aided Design*, 43(12):1598 – 1605, 2011.

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

[111] L. Shapiro and J. Michael Brady. Feature-based correspondence: an eigenvector approach. *Image and vision computing*, 10(5):283–288, 1992.

[112] A. Sharma, R. Horaud, D. Knossow, and E. Von Lavante. Mesh segmentation using Laplacian eigenvectors and Gaussian mixtures. In *AAAI Fall Symposium on Manifold Learning and its Applications*, pages 50–56, 2009.

[113] Yonggang Shi, Rongjie Lai, Sheila Krishna, Nancy Sicotte, Ivo Dinov, and Arthur W. Toga. Anisotropic Laplace-Beltrami eigenmaps: bridging Reeb graphs and skeletons. In *Computer Vision and Pattern Recognition Workshops*, pages 1–7, 2008.

[114] M. Smolkin and D. Ghosh. Cluster stability scores for microarray data in cancer studies. *BMC Bioinformatics*, 4(1):36–42, 2003.

[115] Y. Song, Y. Zhang, C. L. Bajaj, and N. A. Baker. Continuum diffusion reaction rate calculations of wild-type and mutant mouse acetylcholinesterase: adaptive finite element analysis. *Biophysical Journal*, 87(3):1558–1566, 2004.

[116] Y. Song, Y. Zhang, T. Shen, C. L. Bajaj, J. A. McCammon, and N. A. Baker. Finite element solution of the steady-state Smoluchowksi equation for rate constant calculations. *Biophysical Journal*, 86(4):2017–2029, 2004.

[117] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *ACM International Conference Proceeding Series*, volume 257, pages 109–116, 2007.

[118] J. E. Stone, D. J. Hardy, I. S. Ufimtsev, and K. Schulten. GPU-accelerated molecular modeling coming of age. *Journal of Molecular Graphics and Modelling*, 29(2):116 – 125, 2010.

[119] M. Tarini, N. Pietroni, P. Cignoni, D. Panozzo, and E. Puppo. Practical quad mesh simplification. In *Computer Graphics Forum*, volume 29, pages 407–418, 2010.

[120] D. Thomas, M. Scott, J. Evans, K. Tew, and E. Evans. Bézier projection: a unified approach for local projection and quadrature-free refinement and coarsening of NURBS and T-splines with particular application to isogeometric design and analysis. *Computer Methods in Applied Mechanics and Engineering*, 284:55–105, 2015.

[121] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu. Class prediction by nearest shrunken centroids, with applications to DNA microarrays. *Statistical Science*, 18(1):104–117, 2003.

[122] Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun. Designing quadrangulations with discrete harmonic forms. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 201–210, 2006.

[123] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(5):695–703, 1988.

[124] A. Varshney and F. P. Brooks, Jr. Fast analytical computation of Richards's smooth molecular surface. *Proceedings of the IEEE Visualization*, pages 300–307, 1993.

[125] A. Vuong, C. Giannelli, B. Jüttler, and B. Simeon. A hierarchical approach to adaptive local refinement in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 200(49):3554–3567, 2011.

[126] C. L. Wang and D. Manocha. GPU-based offset surface computation using point samples. *Computer-Aided Design*, 45(2):321 – 330, 2013.

[127] H. Wang, T. Lu, O. Au, and C. Tai. Spectral 3D mesh segmentation with a novel single segmentation field. *Graphical Models*, 76(5):440–456, 2014.

[128] L. Wang, X. Gu, K. Mueller, and S. Yau. Uniform texture synthesis and texture mapping using global parameterization. *The Visual Computer*, 21(8-10):801–810, 2005.

[129] Q. Wang, J. JaJa, and A. Varshney. An efficient and scalable parallel algorithm for out-of-core isosurface extraction and rendering. *Journal of Parallel and Distributed Computing*, 67(5):592–603, 2007.

[130] W. Wang, Y. Zhang, M. Scott, and T. Hughes. Converting an unstructured quadrilateral mesh to a standard T-spline surface. *Comput. Mech.*, 48(4):477–498, 2011.

[131] W. Wang, Y. Zhang, G. Xu, and T. Hughes. Converting an unstructured quadrilateral/hexahedral mesh to a rational T-spline. *Comput. Mech.*, 50(1):65–84, 2012.

[132] M. Wardetzky, M. Bergou, D. Harmon, D. Zorin, and E. Grinspun. Discrete quadratic curvature energies. *Computer Aided Geometric Design*, 24(8):499–518, 2007.

[133] G. Wei, Y. Sun, Y. Zhou, and M. Feig. Molecular multiresolution surfaces. *arXiv math-ph/0511001*, 2005.

[134] X. Wei, Y. Zhang, T. Hughes, and M. Scott. Truncated hierarchical Catmull-Clark subdivision with local refinement. *Computer Methods in Applied Mechanics and Engineering*, 291:1–20, 2015.

[135] T. Wilson, J. Sarrate, X. Roca, R. Montenegro, and J. Escobar. Untangling and smoothing of quadrilateral and hexahedral meshes. In *Proceedings of the 8th international conference on engineering computational technology, Paper 36*, 2012.

[136] Y. Xie, J. Cheng, B. Lu, and L. Zhang. Parallel adaptive finite element algorithms for solving the coupled electro-diffusion equations. *Molecular Based Mathematical Biology*, 1:90–108, 2013.

[137] G. Xu. Discrete Laplace-Beltrami operators and their convergence. *Computer Aided Geometric Design*, 21:767–784, 2004.

[138] G. Xu. Consistent approximations of several geometric differential operators and their convergence. *Applied Numerical Mathematics*, 69:1–12, 2013.

[139] G. Xu, Q. Pan, and C. Bajaj. Discrete surface modelling using partial differential equations. *Computer Aided Geometric Design*, 23(2):125–145, 2006.

[140] Z. Yu, Michael J. Holst, Y. Cheng, and J. A. McCammon. Feature-preserving adaptive mesh generation for molecular shape modeling and simulation. *Journal of Molecular Graphics and Modeling*, 26(8):1370–1380, 2008.

[141] D. Zhang, J. Suen, Y. Zhang, Y. Song, Z. Radic, P. Taylor, M. J. Holst, C. L. Bajaj, N. A. Baker, and J. A. McCammon. Tetrameric mouse acetylcholinesterase: continuum diffusion rate calculations by solving the steady-state Smoluchowski equation using finite element methods. *Biophysical Journal*, 88(3):1659–1665, 2005.

[142] J. Zhang, J. Zheng, C. Wu, and J. Cai. Variational mesh decomposition. *ACM Transactions on Graphics*, 31(3):21–31, 2012.

[143] Y. Zhang and C. Bajaj. Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. *Computer Methods in Applied Mechanics and Engineering*, 195(9):942–960, 2006.

[144] Y. Zhang, C. Bajaj, and G. Xu. Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. *Communications in Numerical Methods in Engineering*, 25(1):1–18, 2009.

[145] Y. Zhang, C. L. Bajaj, and B. Sohn. 3D finite element meshing from imaging data. *Computer Methods in Applied Mechanics and Engineering*, 194(48–49):5083–5106, 2005.

[146] Y. Zhang and J. Qian. Resolving topology ambiguity for multiple-material domains. *Computer Methods in Applied Mechanics and Engineering*, 247–248:166–178, 2012.

[147] Y. Zhang, G. Xu, and C. L. Bajaj. Quality meshing of implicit solvation models of biomolecular structures. *Computer Aided Geometric Design*, 23(6):510–530, 2006.

[148] H. Zhao and G. Xu. Triangular surface mesh fairing via Gaussian curvature flow. *Journal of Computational and Applied Mathematics*, 195(1):300–311, 2006.

[149] Q. Zheng, S. Yang, and G. Wei. Biomolecular surface construction by PDE transform. *International Journal for Numerical Methods in Biomedical Engineering*, 28(3):291–316, 2012.

[150] K. Zhou, H. Bao, and J. Shi. 3D surface filtering using spherical harmonics. *Computer-Aided Design*, 36(4):363–375, 2004.