GEOMETRIC AND TOPOLOGICAL MODELING TECHNIQUES FOR LARGE AND COMPLEX SHAPES

By

Xin Feng

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science — Doctor of Philosophy

2014

ABSTRACT

GEOMETRIC AND TOPOLOGICAL MODELING TECHNIQUES
FOR LARGE AND COMPLEX SHAPES

By

Xin Feng

The past few decades have witnessed the incredible advancements in modeling, digitizing and visualizing techniques for three-dimensional shapes. Those advancements led to an explosion in the number of three-dimensional models being created for design, manufacture, architecture, medical imaging, etc. At the same time, the structure, function, stability, and dynamics of proteins, subcellular structures, organelles, and multiprotein complexes have emerged as a leading interest in structural biology, another major source of large and complex geometric models. Geometric modeling not only provides visualizations of shapes for large biomolecular complexes but also fills the gap between structural information and theoretical modeling, and enables the understanding of function, stability, and dynamics.

We first propose, for tessellated volumes of arbitrary topology, a compact data structure that offers constant-time-complexity incidence queries among cells of any dimensions. Our data structure is simple to implement, easy to use, and allows for arbitrary, user-defined 3-cells such as prisms and hexahedra, while remaining highly efficient in memory usage compared to previous work. We also provide the analysis on its time complexity for commonly-used incidence and adjacency queries such as vertex and edge one-rings.

We then introduce a suite of computational tools for volumetric data processing, information extraction, surface mesh rendering, geometric measurement, and curvature estimation for biomolecular complexes. Particular emphasis is given to the modeling of Electron Microscopy Data Bank (EMDB) data and Protein Data Bank (PDB) data. Lagrangian and Cartesian representations are discussed for the surface presentation. Based on these representations, practical algorithms are developed for surface area and surface-enclosed volume calculation, and curvature

estimation. Methods for volumetric meshing have also been presented. Because the technological development in computer science and mathematics has led to a variety of choices at each stage of the geometric modeling, we discuss the rationales in the design and selection of various algorithms. Analytical test models are designed to verify the computational accuracy and convergence of proposed algorithms. We selected six EMDB data and six PDB data to demonstrate the efficacy of the proposed algorithms in handling biomolecular surfaces and explore their capability of geometric characterization of binding targets. Thus, our toolkit offers a comprehensive protocol for the geometric modeling of proteins, subcellular structures, organelles, and multiprotein complexes.

Furthermore, we present a method for computing "choking" loops—a set of surface loops that describe the narrowing of the volumes inside/outside of the surface and extend the notion of surface homology and homotopy loops. The intuition behind their definition is that a choking loop represents the region where an offset of the original surface would get pinched. Our generalized loops naturally include the usual $2g$ handles/tunnels computed based on the topology of the genus-g surface, but also include loops that identify chokepoints or bottlenecks, i.e., boundaries of small membranes separating the inside or outside volume of the surface into disconnected regions. Our definition is based on persistent homology theory, which gives a measure to topological structures, thus providing resilience to noise and a well-defined way to determine topological feature size.

Finally, we explore the application of persistent homology theory in protein folding analysis. The extremely complex process of protein folding brings challenges for both experimental study and theoretical modeling. The persistent homology approach studies the Euler characteristics of the protein conformations during the folding process. More precisely, the persistence is measured by the variation of van der Waals radius, which leads to the change of protein 3D structures and uncovers the inter-connectivity. Our results on fullerenes demonstrate the potential of our geometric and topological approach to protein stability analysis.

To my family

ACKNOWLEDGEMENTS

During my five years study in the PhD program, I received numerous help from mentors, collaborators and friends.

First, I would like to express my sincere gratitude towards my advisor Professor Yiying Tong, who has been a great advisor and mentor over the five years. I would not be able to finish my PhD program and thesis research without his continual guidance, help and encouragement. I thank the opportunity he gave me to pursue my research dreams in computer science. From him, I learn a great amount of knowledge in geometric modeling and topological analysis. Professor Tong always encourages me to touch the latest advancements in this field by providing any chances he could. I still remember my fresh and exciting trip to SIGGRAPH 2010 which generously paid by him. He is also a great educator, being so patient to explain the very details to me when I am confused with a theory or concept. He also has his simple ways to illustrate the connections between the abstract theories and the common sense in life. Over the years, I have learned skills to perform research work, as how to find a good research topic to begin with, how to tackle and solve a problem, and how to express the thoughts and ideas into publication. Another skill I have learned from him is "multitasking" yourself. Being a multitasking researcher himself, he guides several projects in computer graphics research, which lead to very productive achievements. The days which we were together to explore the research ideas are the most valuable experience in my five years study. I would also like to thank Professor Tong and Mrs. Tong for providing the whole group with delicious party food on those holidays. I admire him as a great researcher, an inspiring advisor, and an easy-going friend as well.

I would also like to thank Professor Guowei Wei, a great mentor and collaborator during my PhD work. His mathematical biology research fascinates me by applying mathematical theories to biological problems. His research has a great impact on my biomolecular modeling research. He also generously sent me to several mathematical biology conferences to meet talented people in

this research field. He even sent me to study the newest developments in another research group at University of Michigan for weeks.

During each phrase of my life, my parents and sister give their unconditional love and support no matter what happens. They are always there for me when I need them, especially when I go through some difficult situations. I would also like to thank Shuai Yuan from my deepest heart. She often makes delicious food for me. She also takes care of me well when I am sick.

I would like to thank the PhD committee members Professor Eric Torng, Professor Charles Owen and Professor Guowei Wei for their insightful comments and guidance regarding the thesis. I would like to thank Dr. Kelin Xia, who is my collaborator on biomelecular modeling. The discussions with him are extremely helpful for me to understand the biomolecular systems. I would like to thank Beibei Liu, who assists me on choking loops project. I would like to thank Dr. Yuanzhen Wang for collaborating on compact volume mesh project. I would like to thank Xiaojun Wang for inspiring discussions and collaborations on many interesting course projects. I would also like to thank Dr. Qiong Zheng for collaborating initial work on biomelecular modeling.

Last but not least, I would like to thank my friends whom I share the fun and joy during the five years.

Time flies so fast. I do not even realize that I have been at this land for five years. It feels like yesterday that I got off the plane at Detroit airport. I wish all the best to the people I meet here. I will always be a Spartan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Background

The past few decades have seen incredible advancement in modeling, digitizing and visualizing techniques for three-dimensional shapes. With increasingly popular consumer level 3D scanners and novel interactive tools available to the public, construction of detailed three-dimensional (3D) models becomes cost effective and practical even for non-expert users. The inexpensive graphics hardware devices nowadays also help increase the demands for 3D models. All of these factors contributed to an explosion in the number of 3D models created each year.

The number of geometric models available on the Internet to scientists and engineers for research and manufacturing design also grows fast. Numerous 3D model warehouses, repositories are created and made available to the scientific and industrial community. For instance, the website GRABCAD (www.grabcad.com), which started in 2009, already gathered around 381,000 high quality free computer-aided design (CAD) models. Another example is the Trimble 3D Warehouse (formerly Google 3D Warehouse) created in 2006, which accumulated a large number of geometric data for 3D objects, especially landmark architecture models. For analysis of the various geometric properties of three-dimensional shapes, some benchmark repositories were also created for scientists and researchers [157].

In the meantime, biological sciences are undergoing the transition from an empirical, qualitative and phenomenological discipline to a comprehensive, quantitative and predictive one [186], producing a huge amount of three-dimensional chemical and biomolecular information to be further studied. New compounds and new structures along with their geometric information are discovered every day, even though millions have already been in the record—according to Chemical Abstracts Service (CAS)(www.cas.org), an authority for chemical information of the world, there are already 70 million organic and inorganic substances. The rate at which new substances are

discovered is also record high. According to the CAS website, the 70 millionth substance was discovered just 18 months after the 60 millionth substance had been found. At the time of this writing, approximately 15,000 new substances are added into their records every day [150]. Protein Data Bank (www.rcsb.org) is another repository where a huge amount of biomolecular information has been gathered. It contains 3D structural information of large biological molecules, e.g., proteins, obtained typically by X-ray crystallography. Currently the Protein Data Bank has more than 97,000 structures. Figure 1.1 shows the number of searchable structures per year in Protein Data Bank.

With the overwhelming amount of 3D information, scientists need efficient tools to study their chemical and physical properties. Unlike CAD models, which are more likely to have regular shapes and can be stored in vector format, biomolecular models often have extremely large storage size, complex geometric shape and nontrivial topology. Most objects studied in biomolecular field have tens of thousands or even millions of atoms, which create great challenges to the numerics. The topology of chemical substances structures found is often extremely complicated. Many biochemical properties of biomolecules have close relation to their geometric shapes [35, 63, 194]. Analyzing the geometric shapes of complex shapes to discover the intrinsic relationships between shapes and properties also brings tremendous challenges as well as opportunities for current researchers.

The objective of this thesis is thus to design efficient computational modeling and analysis techniques for large and complex shapes through a combination of geometry processing and computational topology devices. The target objects include the aforementioned proteins, subcellular structures, organelles, and large multiprotein complexes, as well as graphics models. We first present a novel efficient representation for 3D domains to render algorithms based on such 3D volume representations scalable, and to facilitate the subsequent geometric or topological modeling and analysis processes. Then, we synthesize and adapt existing geometric modeling techniques into a complete toolkit specifically designed for geometry processing of biomolecular surfaces, encompassing the pipeline from model generation, smoothing, to curvature analysis and binding

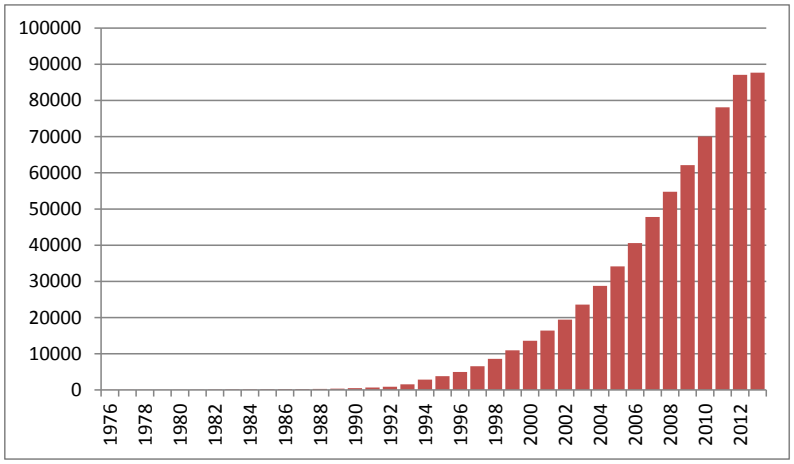

Figure 1.1: Number of searchable structures per year in Protein Data Bank (www.rcsb.org). Horizontal axis: year number. Vertical axis: number of structures.

site prediction. Furthermore, based on persistent homology theory, we defined a set of topological features and designed an algorithm for their detection on complex surfaces, which can be applied to identifying important structures, e.g., ion channel detection in biomolecular surfaces. Last, we explore the application of our topological algorithm in the study of protein stability with test results on fullerenes.

1.2 Compact Mesh Representations

Shapes in geometric modeling are often discretized as meshes, i.e., surfaces or volumes tessellated into collections of smaller cells. Meshes comprised of regular or irregular polygons (2D) or polyhedra (3D) are often constructed by mesh generation algorithms, and generally fall into two categories: surface mesh and volume mesh. Meshes have been extensively used in geometric modeling due to their efficient and flexible forms to represent shapes [3, 5, 13, 14, 17, 22, 26, 43, 54, 57, 61, 104]. Surface meshes are used to represent the boundary of 3D objects or thin shells, which are widely used for geometric modeling purposes, e.g. in computer animation and game industry. In contrast, in the scientific computing fields, where interior regions of 3D objects

or domains are analyzed (e.g. finite element analysis and finite volume analysis) volume meshes form the foundation to build algorithms on. Figure 1.2 shows some example volume meshes. Adaptively refinement of volume meshes is often the key to efficient generation of surface meshes through, for instance, marching tetrahedra algorithms [87, 172]. In our geometric and topological analysis of biomolecules, volume meshes are also indispensable.

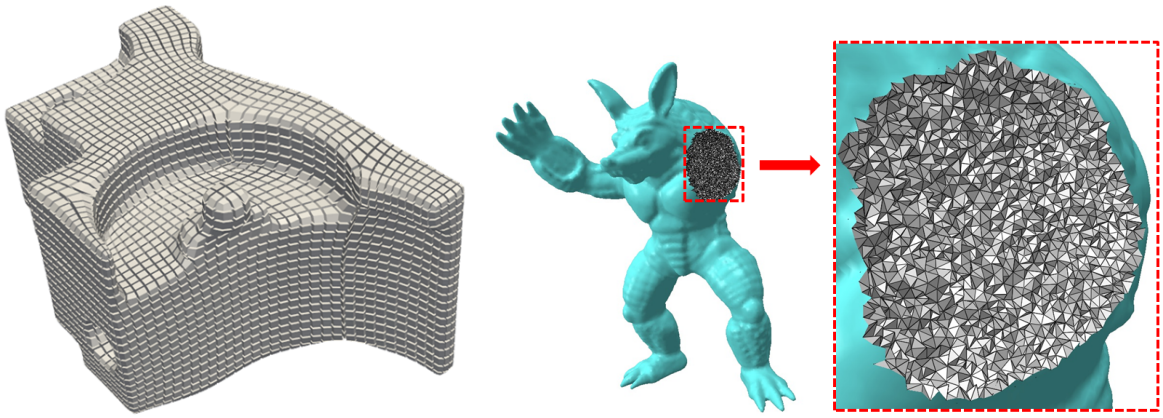

Figure 1.2: Two volume meshes. Left: a CAD workpiece hexahedral model [93]. Right: an armadillo model with a cutaway view of left arm.

We limit our discussion of previous work in this section to closely related 3D data structures. For a survey of 2D mesh data structures, see, e.g., [159, 149].

1.2.1 Existing Work and Challenges

In scientific computing, 3D volumes are often assumed to be 3-manifolds, i.e., shapes without degenerate structures such as "shark-fin" or "hanging rod". Under this assumption, a table of mesh element connectivity that maps volume cells to their corner vertices provides *complete* information about incidence among vertices, edges, faces, and volume cells. While this can be sufficient for various geometry processing algorithms[167, 126], many computational applications require constant-time upward or downward incidence queries (to access lower dimension cells from higher dimension cells, or vice versa), which cannot be achieved without auxiliary connectivity information. This requirement was referred to as *comprehensiveness* in [3]. Some applications may

only require certain incidence queries, e.g., cell-face relations in ray-tracing [123], and cell-vertex relations in Delaunay tetrahedralization. On the other hand, many Finite Element Method-based applications may need all incidence queries, including face-edge relations [45].

To tackle these issues in practice, several different data structures were proposed to store necessary adjacency information to meet the comprehensiveness requirement. Guibas and Stolfi [88] proposed a face-edge data structure. Brisson proposed a more abstract "cell-tuple" data structure based on the idea of boundary representation, which implies that it is theoretically possible to "order" all the $k-1$-cells and k-cells around a $k-2$-cell on the boundary of a $k+1$-cell. Another data structure, Combinatorial Map, originally defined for polygonal meshes [57], can also represent orientable quasi-manifolds. It can be extended to generalized d-maps to encode non-orientable manifolds [112]. Beall and Shephard [13] proposed another topology-based mesh data structure, which stores the adjacency information, at the cost of large additional memory. To alleviate the problem of a large memory footprint, compression techniques are introduced to reduce the storage space for generalized d-maps [136]. However the adjacency information is only available after decompression.

Recently, a number of memory compact data structures have been proposed. For example, [17] designed a data structure with a full list of connectivity and adjacency information using only 7.5 bytes per tetrahedron. However the method can only be applied to tetrahedral meshes, which limits its utility when other additional types of meshes are needed. Alumbaugh et al [3] introduced a compact array-based data structure of 3D orientable manifold cell complexes. They defined the concept of anchored half-faces to compute incident adjacent cells in constant time. Their concept and data structure work well for most of the common queries needed in scientific computing, e.g., when an edge is represented by two vertex indices. However, with their data structure it is impossible to allocate a unique identifier for the edge using the proposed connectivity representation. Another weakness in their representation is the lack of direct face-edge incidence access, necessary for comprehensiveness. [26] independently created a similar data structure by storing mesh elements with predefined mesh element types. They use bit flags to keep the edges

and faces uniquely represented without corruption. They also use "reverse indices" to further improved the speed of adjacency queries.

From the perspective of scientific computing, array-based data structure is often more convenient for languages like FORTRAN than linked-list-based representations. It is also easier to parallelize algorithms using such data structures across multiple CPUs [3].

In addition to existing research work, a number of libraries providing practical implementations of volume mesh data structures are released online for public use. The widely used Computational Geometric Algorithms Library (CGAL) [43] already includes an implementation of Combinatorial Maps; OpenVolumeMesh [104], released recently, is based on OpenMesh [22], which stores incidence information between k-cells and $k-1$-cells; libMesh [100] uses a complete but *not* comprehensive data structure; CGoGN [168] implements the Generalized d-Maps. However, none of these existing software packages is optimized for both memory usage and queries efficiency. A data structure with small memory footprint that can efficiently handle queries of incidence and adjacency would thus benefit a wide range of applications in graphics and scientific computing in general.

1.3 Geometric Modeling on Biomolecular Data

1.3.1 Importance of PDB and EMDB Data

Structural biology is an essential part of modern biological sciences. A basic role of structural biology is to provide structural information of biological macromolecules, especially proteins and nucleic acids, and the interpretation of macromolecular structures, namely, structure-function correlations.

Macromolecular 3D shapes can be indirectly obtained from a number of experimental means, including macromolecular X-ray crystallography, nuclear magnetic resonance (NMR), electron paramagnetic resonance (EPR), cryo-electron microscopy (cryo-EM), multiangle light scattering, confocal laser-scanning microscopy, small angle scattering, and ultra fast laser spectroscopy. The

main workhorses for single macromolecules are crystallography and NMR. The advanced X-ray crystallography technology is able to provide decisive structural information at angstrom and sub-angstrom resolutions, while NMR experiments often offer structural information under physiological conditions. Both X-ray crystallography and NMR are technologically relatively well developed, except for their applications in special tasks, such as the crystallization of membrane proteins. Macromolecular structural data are deposited at the Protein Data Bank (PDB), which is a major source for much biophysical modeling, simulation and analysis.

One most important new trend in structural biology is the study of large protein complexes and subcellular organelles, which plays an essential role in many key biological processes, including genome replication, transcription, translation, protein-folding, signal transduction, and viral infection. The structural information of large protein complexes and subcellular organelles is crucial for exploring the molecular mechanisms behind complex biological processes. Unfortunately, most conventional experimental means and imaging modalities well-suited for relatively small proteins do not work well for multiprotein complexes and subcellular organelles. Recently, electron tomography, especially cryo-electron microscopy (cryo-EM) [175], has become a powerful tool for revealing 3D structures of macromolecular complexes in different functional or biological states. The feasible resolution of cryo-EMs ranges from 80 to 2Å, capable of bridging the gap between live-cell imaging and atomic resolution structures. Its sample is bombarded by electron beams at cryogenic temperatures to improve the signal to noise ratio (SNR). Its working principle is based on the projection (thin film) specimen scans collected from many different directions around one or two axes, and the creation of 3D images by using the Radon transform. Cryo-EM allows the imaging of specimens in their native environment and is capable of providing 3D mapping of entire cellular proteomes together with their detailed interactions at a nanometer resolution [127, 139, 109, 169]. Structures determined by cryo-EMs are deposited at the EM DataBank (EMDB), a significant resource for global deposition and retrieval of cryo-EM data. Unlike PDB, which usually contains information about structures of proteins, nucleic acids, and complex assemblies obtained from X-ray crystallography or NMR spectroscopy at the atomic level resolu-

tion, EMDB typically provides information about multiproteins, organelles, cell and tissue from cryo-EMs at the molecular level resolution.

Since most biological specimens are extremely radiation sensitive, they can only sustain the illumination of a limited electron dose. As a result, cryo-EM images are inevitably of low SNRs, which lead to limited resolutions [175]. In practice, cryo-EM maps often do not contain adequate information to offer unambiguous atomic-scale structural reconstruction of biological specimens. Additional information obtained from other techniques, such as crystallography, NMR and computer simulation, is utilized to interpret the cryo-EM maps. However, the resolution of cryo-EM maps has improved dramatically over the past few years, owing to the technical advances in experimental hardware, noise reduction and image segmentation techniques. By further taking the advantage of symmetric averaging, many cryo-EM based virus structures have already achieved a resolution that can be interpreted in terms of an atomic model. Therefore, it is time to utilize cryo-EM images for molecular and atomic scale mathematical modeling and computer simulation of subcellular structures, organelles and large multiprotein complexes.

Most 3D imaging data obtained from cryo-EM and many other tomographic modalities are currently presented in a digitized format, such as a volumetric density distribution, where each Cartesian grid point is assigned with a scalar value associated with the local electron scattering power. For the purpose of visualization, one needs to display them in the form of a series of 2D rasterized images, by rendering the 3D shapes generated by isosurface extraction, or by direct volumetric rendering. For the purpose of geometric analysis, structural features in the complex settings of cellular landscapes are further characterized in terms of surface areas, surface enclosed volumes, and Gaussian and mean curvatures. For the purpose of mathematical modeling and computation, the resulting 3D geometric shape is to be further described in either the Lagrangian representation or the Eulerian representation. The Lagrangian representation is a basis for the material formulation of the biological evolution, in which surface elements are directly evolved according to a governing equation or a set of rules [35, 202]. Similarly, the Eulerian representation facilitates the spatial formulation of the biological dynamics, in which the biological shape is embedded

through a hypersurface function, or a level set function, and such a function is then evolved under prescribed physical and/or biological principles [10, 34, 33]. Both Lagrangian and Eulerian approaches have their own pros and cons, and serve their purposes in mathematical modeling and computation. Figure 1.3 shows two cryo-EM models in meshes form.

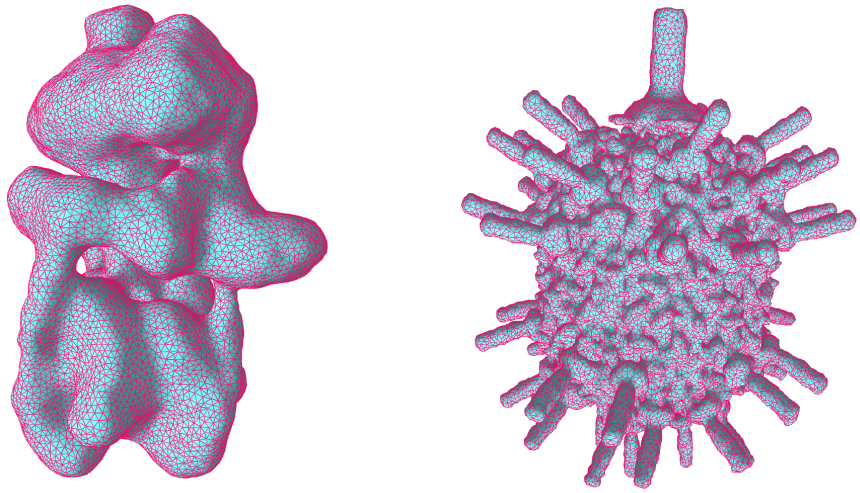

Figure 1.3: Two Cryo-EM data. Left:EMD1590, Manduca sexta vacuolar ATPase complex. Right: EMD1265, Bacteriophage $\phi 29$ (a viral DNA-packaging motor)

1.3.2 Existing Work and Challenges

Some methods in image processing, geometry processing, or signal processing in general can be applied to biomolecular data, including PDB and EMDB data. Here we briefly discuss the various methods for preprocessing of the noisy data acquired through various sources, and also those for subsequent geometric modeling and analyses.

It remains a great challenge to quantitatively model and predict the structure, function, dynamics and transport of complex self-organizing biological systems. Geometric modeling not only bridges the gap between biomolecular data and biological conceptualization and interpretation, but also provides a basis for mathematical modeling, analysis and computation [202, 64]. In 1953, Corey and Pauling proposed the atom and bond model of molecules [41], which has since become a cornerstone in physical science. Numerous other models, including the van der Waals

surface (vdWS), the solvent-excluded surface (SES) (also known as molecular surface (MS)), and the solvent-accessible surface (SAS) have been proposed [108, 137]. The combination of these biomolecular surfaces with the calculated electrostatic potentials on and around them, has become an important procedure in the analysis of biomolecular structure, function, and interaction, such as ligand-receptor binding, protein specification, drug design, macromolecular assembly, protein-nucleic acid and protein-protein interactions, and enzymatic mechanism [140]. A variety of physical and geometric models are developed during the past few decades.

The widely applied biomolecular surfaces, especially SESs, have known drawbacks in their definitions. One of these problems is the admission of geometric singularities, i.e., tips, cusps and self-intersecting facets, which lead to computational instabilities and induce excessive numerical errors [40, 58, 82, 144]. Another defect is that these surfaces are simply ad hoc divisions of a biomolecule from its surroundings, without the consideration of the physical laws of surface energy minimization and surface evolution under the interaction with the aqueous environment. At the fundamental level, there is no sharp division between solvent and solute because their electron densities overlap with each other. In the past few years, many theoretical models have been proposed to address these problems [183, 11, 10, 9, 210].

1.3.2.1 Noise Removal, Surface and Meshing

Currently, the SNR of 3D imaging data for subcellular structures, organelles and large multiprotein complexes is typically in the neighborhood of 0.01 dB [175]. To make the situation worse, the image contrast, which depends on the difference between electron scattering cross sections of cellular components, is also very low in most biological systems. Consequently, appropriate noise reduction is an indispensable process in the structure reconstruction from 3D imaging data. To improve the SNR and image contrast, researchers have employed a wide variety of denoising schemes, including wavelet transform techniques [164], nonlinear anisotropic diffusions [71, 67] or Beltrami flow [66], bilateral filter [170, 95, 132], and iterative median filtering [173]. Despite much effort, noise-reduction remains a challenging task and is far from adequate, due to the ex-

tremely low SNR and other technical complications [175]. Innovative mathematical approaches are necessary to further tackle this problem.

Geometric flows, in which the flow motion is governed or influenced by geometric properties, such as curvatures, have become an established approach to image analysis and surface generation in the past few years. Particularly, mean curvature flows have been a popular subject in applied mathematics for image analysis, material design [161, 129, 151] and surface processing [206]. The first use of partial differential equations (PDEs) for image analysis dates back to 1983 [190]. Witkin noticed that the evolution of an image under a diffusion operator is formally equivalent to the standard Gaussian low-pass filter for image denoising [190]. Perona and Malik introduced an anisotropic diffusion equation [134] to protect image edges during the diffusion process. The Perona-Malik equation stimulated much interest in applied mathematics [134, 163, 184, 29, 188]. Over the past two decades, many related mathematical techniques, such as the level set formalism devised by Osher and Sethian [131, 151], Mumford-Shah variational functional [124], and the total variation (TV) minimization [142], have been widely used for image analysis [18, 25, 130, 145, 146].

To improve the efficiency of noise removal, Wei introduced the first family of arbitrarily high order nonlinear PDEs for image denoising and restoration in 1999 [184]. Many fourth-order evolution equations were introduced in the literature for image analysis [29, 199, 166, 116, 84]. These equations were proposed either as a high-order generalization of the Perona-Malik equation [184, 9] or as an extension of the TV formulation [29, 199, 166, 116]. The essential assumption in these high order evolution equations is that high-order diffusion operators are able to remove high frequency components more effectively. High order geometric PDEs have been widely applied to image and surface analysis [28, 184, 29, 199, 166, 84, 116, 85, 9]. Due to the stiffness of high order nonlinear PDEs, computational techniques for solving higher order geometric PDEs are of great importance. For instance, alternating direction implicit (ADI) schemes are developed in the literature for integrating high order nonlinear PDEs [189, 9].

Image processing PDEs of the Perona-Malik type and total variation type are mostly designed

to function as nonlinear low-pass filters. In 2002, Wei and Jia [188] introduced coupled nonlinear PDEs to behave as high-pass filters. These coupled nonlinear PDEs are demonstrated for image edge detection. The essential idea behind these PDE based high-pass filters is that when two Perona-Malik type of PDEs evolve at dramatically different speeds, the difference of their solutions gives rise to image edges. This follows from the fact that the difference between an all-pass filter (i.e., identity operator) and a low-pass one is a high-pass filter [188]. The speeds of evolution in these equations are controlled by the appropriate selection of the diffusion coefficients. These PDE-based edge detectors have been shown to work extremely well for images with a large amount of texture [165, 188]. Most recently, the PDE transform is introduced for functional mode decomposition [180, 179] based on arbitrarily high order PDE high-pass filters. Such an approach has significantly extended the utility of PDEs for image, surface and data analysis. Similar to wavelet transform, the PDE transform has controllable time-frequency location and perfect reconstruction. The PDE transform has found its success in molecular surface generation of proteins [210].

The use of curvature controlled PDEs for biomolecular surface construction was initiated in 2005 [183]. Atomic coordinate information of a protein is embedded in 3D Eulerian grids to undergo geometric flow evolution before the protein surface is extracted via the marching cubes method from a level-set type of hypersurface function. This approach was combined with a variational procedure to generate the first variational biomolecular surface model, the minimal molecular surface, for proteins [10]. Molecular interactions were further incorporated in this approach to develop potential and curvature driven geometric flows for the construction of biomolecular surfaces [9]. Recently, many variational multiscale models have been introduced based on the geometric-flow separation of solvent and solute domains [186, 34, 33, 208].

After the surface construction, a further issue in geometric modeling is the surface and volumetric (i.e., boundary and interior) meshing [146, 51, 118]. There are a wide range of methods that can be used for this purpose. Numerous elegant methods, such as the probabilistic methods for centroidal Voronoi tessellations [52, 96], the optimal Delaunay triangulation and graph cut based variational surface reconstruction [177], and other surface remeshing enhancement meth-

ods and techniques [147, 143, 146, 182, 77], have been developed for surface reconstruction or surface remeshing during the past two decades. In general, high quality triangle surface meshes must be low-noise, low memory cost, near 60° for the majority of element angles and aligned with the physical features. [202, 203] discussed the use of adaptive feature-preserving methods for biomolecular surface meshing. They used the constrained Delaunay triangulation implemented in TetGen [158] for volumetric meshing [202].

One of the most important geometric analyses of surfaces is curvature estimation. Curvature is a measure of how much a curve deviates from being straight or a surface from being flat [102]. Curvature has been used to analyze the stereospecificity of molecular surfaces [38]. The essential idea is that geometries of binding partners are locally complementary to each other at the binding site(s). Curvature is also used as a geometric descriptor to characterize the shape of known protein binding sites so as to identify matching site(s) in other proteins and ligands. However, in real world cases, the effect of stereospecificity may be offset by hydrogen bond, polarization, electrostatics, solvation and allosteric modulation.

Although there are many existing methods to tackle one or two specific problems, there is no existing research work on the systematic treatment on geometric modeling of subcellular structures, organelles and large multiprotein complexes. Many novel, efficient and testified computational algorithms developed in the computational geometry and geometric modeling community have not been adapted to large biomolecular data to help the advances in experimental data collection, such as Cryo-EM.

Since PDB and EMDB data are often extremely large, reconstruction of biological structures from noisy 3D imaging data needs robust and efficient algorithms. Using discrete geometric representations to accurately calculate surface areas and surface enclosed volumes of the biological structure requires testified algorithms developed by computational geometry community. Evaluation of higher level geometric properties of the shape, e.g. curvatures of macromolecules, also calls for advanced and well-developed computational algorithms. In converting of the data to geometric representations, discretization of the data domain can also bring large errors to the results.

As many modeling algorithms have related parameters to tune to generate reasonable results, invaluable empirical rules on parameter setting can be obtained, if the effects of the parameters are thoroughly experimented for PDB and EMDB data.

1.3.2.2 Solvation Model

In a physiological environment, up to 65%-90% of human cellular mass is water. Consequently, almost all the biological processes in cell, such as signal transduction, transcription, translation, protein folding, protein ligand binding, and charge and mass transport, occur in aqueous surroundings. Therefore, the understanding of the solvation is of fundamental importance for quantitative modeling and analysis of all the above-mentioned processes. Explicit solvent models and implicit solvent models are two major approaches for solvation analysis [141, 155, 160]. For explicit solvent models, both the solvent and the solute are described in atomic detail and extensive sampling is required. Implicit solvent models are designed to reduce the number of degrees of freedom by using a dielectric continuum to describe the solvent while admitting a microscopic atomic description for the biomolecules [197, 8, 15]. Due to their fewer degrees of freedom, implicit solvent models, such as the Poisson-Boltzmann (PB) model or the Poisson equation (PE) model when there is no salt in the solvent, are widely used [6, 7, 135]. The coupling of the PB or PE with the generalized Laplace-Beltrami flow has the potential of describing the formation of molecular surface in realistic solvation environments. Conceptually, a solvation free energy can be divided into two major parts: a nonpolar part associated with inserting an uncharged solute into the solvent [37] and a polar part associated with charging the solute in vacuum and solvent [119, 34]. The nonpolar free energy and polar free energy can be represented by a total free energy functional [185]. By using the variational principle, a new geometric flow equation is generated that controls the biomolecular surface formation and evolution via curvature and potential driven [9, 34, 35, 36]. This model takes into consideration of the surface energy minimization and also the solvent-solute interaction, and gives a multiresolution representation of biomolecular surfaces in their native environment. Additionally, the external potential term can be used to incorporate different kinds of effects, such

as chemical reaction, fluid flow, and elastic description of macromolecules [187]. Thus, such solvation models not only require both the implicit and explicit geometric models through Lagrangian and Eulerian representations, but also rely on analysis of curvature on the surfaces in both representations in combination with other physical quantities, such as charge density.

1.4 Topological Feature Detection of Shapes

One often needs to investigate not only in modeling and analyzing local geometric properties of large complex shapes, e.g. cryo-EM data, but also in analyzing the complex shapes to extract structural information about the data. Topological feature detection, which analyzes both geometric and topological information of the shapes, arises naturally in further analysis of the essential structures of complex shapes.

Both geometry and topology measure and classify shapes. Geometry studies the invariants of a model under rigid body transformation, while topology studies invariants under continuous transformations of the model. For example, moving a teapot model with a handle from one place to another place does not change its geometry, e.g. surface curvatures. Stretching the teapot gradually and deforming its surface smoothly into a donut shape does not change its topology, e.g. genus. Thus, roughly speaking, geometry provides floating point numbers for local measurements, while topology provides integers measuring global structures.

Recently, the development of the persistent homology theory [56] in topology study provides a way to measure sizes of topological features, which also enables robust tracking and analyses of connected components, handles loops and tunnels loops, cavities of the volume. The idea of the persistence can be elucidated in its application to analysis of the height field of a terrain. Given a fixed threshold of height, we consider the connectivity of the regions with the height below the threshold. As the threshold changes, the connectivity of the targets regions also changes. Connected components that remain intact with a large change in the threshold are regions representing separate peaks instead of a bump on the road in the terrain.

The persistence endows all such topological structures with a measure, providing a continuous

importance scale for topological features as well as resilience to noise. Topological persistence measures are no strangers to computational science, graphics, and visualization. For example, the use of 3D Morse-Smale complexes in the construction of a clean distance field from noisy data is in fact using persistence to filter out critical points that cancel out in pairs through perturbation of the original field [89]. The persistence concept applied to volumetric data can also be used to extract medial structures more robustly than previous methods [113]. The topological filtering algorithm based on Reeb graphs [191] can also be seen as using persistence from height functions, although height is often not the most natural choice, even for the tiny nontrivial loops representing topological noise.

1.4.1 Topological Features

Loops that cannot shrink to a point by deforming over the surface play an important role in topology. In practice, such non-contractible loops have a multitude of potential applications in segmentation, parameterization, topological simplification and repair, path planning, detection of geometrical and topological features, biomedical imaging, and determining integrability of partial differential equations; see, e.g., [110, 191, 23, 49, 45, 20, 204, 98]. A number of algorithms based on surface homology (equivalence classes of such loops, equivalent when they form the boundary of a patch) and homotopy (equivalence classes of such loops, equivalent when they can deform continuously from one to the other) have been proposed. Most of them find $2g$ such loops that form a set of generators for the first homology or homotopy groups of a surface with genus g. Some algorithms can provide geometrically shortest loops for such bases, and some can further classify the loops in a basis into g handles (loops around the solid inside) and g tunnels (loops around the void outside). However, although $2g$ loops are enough to form a basis, there are often still a lot more than $2g$ nontrivial loops that are candidates for topologically and geometrically important structures of the object in various applications. For example, the genus for the buckyball model as shown in Figure 1.4 is 31, but there are 90 equally short loops that can replace any of those in the handle-type homology basis.

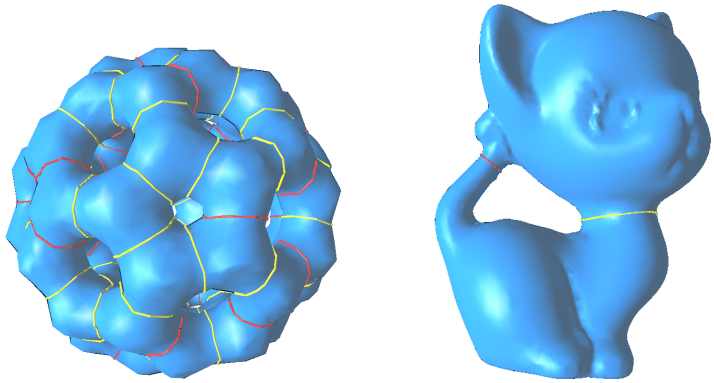

Figure 1.4: Left: C_{60} "buckyball" is of genus-31, but there are 90 equally short loops. Right: Kitten model with two loops as topological features corresponding to the narrow parts of the shape.

One way to find these additional loops is to examine different homology classes spanned by combinations of the loops in the basis. A few algorithms allow for the discovery of the shortest loop within a single homology class (i.e., loops that correspond to the sum of several loops in the basis). However, there is no predetermined way of telling which combinations should be used or where to start the search. Even for objects with the same genus, there can in fact be different numbers of useful nontrivial loops depending on the geometry. Even if one opts to count all possible combinations of the $2g$ loops and finds an oracle that distinguishes useful loops, it will still miss handle-like or tunnel-like structures of a genus-0 object: in this case, the basis contains not a single loop to begin with.

One possible solution is to allow the test of whether a loop is contractible to be performed in a local region, for instance, the intersection of the surface with a ball-like local neighborhood of a certain point. This may solve the problem when the global topology of the surface is trivial as it discovers loops that are non-trivial locally. However, the location of the center point of the designated neighborhood is not easy to determine automatically. Another issue with this method is that we may potentially find a lot of nearby locally nontrivial loops even if we add the constraint that they must be also locally shortest, for example, by considering long cylinder-like handles (such as the tail of kitten in Figure 1.4).

Finding a complete set of shortest nontrivial loops is considered an open problem [48]. However, finding such a set of loops is desirable in many occasions. For instance, when editing or filtering topological noises, such as a thick membrane (like the buckyball), with 32 tiny holes on it, filling the 31 tunnel loops in the homology basis can only turn the model to a genus-0 structure, instead of fixing the topology to a membrane enclosing a ball-like empty space inside. Similarly, objects may be incorrectly connected by a thin tube-like topological noise to form a genus-0 model, and the empty homology basis will not help find a separating loop. To detect where a certain sized object will get stuck at some bottleneck location inside a volume enclosed by a surface can be crucial to motion planning, going through short loops in a homology basis is insufficient, e.g., in a genus-0 model. When geometrically analyzing the easy-to-break handle-like structures in a mechanical part, the handles in the homology basis will only provide a subset of these structures. In biomolecules, finding tunnel-like structures is important to identify ion channels, crucial in determining biological functions and in drug design [211]. As shown in the 1mag model above, shortest tunnel loops actually miss both loops in the ion channel. In all these cases, to be able to detect a complete set of bottleneck loops is a prerequisite. Other potential applications in defining and computing such a full set of loops include analysis of shape and topology of 3D objects, surface parameterization, meshing, and feature detection.

1.4.2 Existing Work and Challenges

There have been a wide variety of algorithms proposed for the purpose of computing homology basis. Some of these methods compute nontrivial loops on the surface mesh directly. The greedy homotopy and homology algorithm [59] gives an optimal solution in theory. Other methods rely on a tetrahedralization of the interior/exterior volume; the HanTun algorithm [49] is the first of these volumetric methods to show results with automatic detection of loops on surfaces, categorized into either handles or tunnels taking geometric measurements into consideration. Our method is also volume-based as we need to identify chokepoints.

A number of algorithms have been proposed to find the shortest loop within a single homology

class. The problem has been proven NP-hard with coefficients of the homology taken as integer modulo 2 [31]. However optimal codimension-1 cycle in integer homology classes can be computed in polynomial time for manifold meshes of arbitrary dimension [47]. Given constant genus, the shortest cycle in Z_2-homology classes can be found in $O(n \log\log n)$ time [27, 94]. Most of these methods find shortest loops restricted to closed paths along mesh edges (including, e.g., [24, 44, 107]), with the exception of a few recent methods for computing local minimal loops within a given homotopy class. One of them computes geodesic loops using a discrete geodesic curvature flow [192] based on level set functions. Another method is based on iteratively computing the shortest path inside a triangle strip loop and updating the triangle strip; it is highly efficient with an empirical time complexity of $O(mk)$, where m is the number of vertices in the original loop, and k is the average number of edges in the sequences of edges the loop swept through during the shortening process [198]. We use the latter, but only to refine our results. The constriction loops in [90] are also defined as geodesic loops, but instead of detecting the true narrowing of the volume inside, they find initial vertices on the surface with large negative Gaussian curvature or through a progressive surface simplification [91].

As mentioned above, to the best of our knowledge, no existing method attempts to compute the set of all possible candidates of topologically nontrivial loops that are reasonably apart from each other, or even just to formulate them mathematically. Segmentation is a potential application, where the topologically relevant loops can be incorporated as part of patch boundaries; segmentation methods also implicitly generate boundary loops (e.g., [153, 83, 98, 30]). [30] actually employs 0-th persistent homology of a filtration based on a scalar function defined on a point cloud. However, the emphasis of other segmentation methods is often more on surface geometric features such as ridges and valleys (or peaks), and a thorough discussion on these methods is beyond the scope of this thesis.

1.4.2.1 Application in Molecule Stability Analysis

We study one particular biomolecular problem as our sample application of the topological features, and give a brief introduction to the problem here. Protein folding is the process through which the randomly coiled polypeptide assumes its three-dimensional functional structure. There are two long-held views in this process: one is that the protein's native structure is determined only by the its amino acid sequence, as suggested by the Anfinsen's dogma [4]; the other is that only the well-defined structure of protein is essential to its function [195]. Both views are challenged by the recent discovery, that the folding process depends on solvent, ion concentrations, pH value, temperature, and sometimes the presence of cofactors and protein machinery—the molecular chaperones. Further, many partially folded or unstructured proteins can still remain functional. In cell environment, folding process is rather complex. Polypeptide chain, which is translated from the mRNA in ribosome, can be formed into various intermediate conformation states and transferred from one state to another. These unstable conformation states contain few persistent structures and are easy to aggregate, especially when their concentrations increase above certain thresholds. Some of the initial disordered aggregates simply dissociate. Others may reorganize to form structured aggregates and further grow into fully mature fibrils. Under some pathological conditions, the β-structure aggregates occur and self-associate to form the amyloid fibril, which is associated with the so-called protein misfolding (or protein conformational) disease, such as Alzheimer's disease, Parkinson's disease, and Mad Cow disease. In a well-functional cell, all of these different conformational states and the transitions among them are rigorously regulated and monitored by the biological environment, particularly, the molecular chaperones, which can bind to and stabilize the favored intermediate states to prevent the formation of misfolded protein structures.

Proteins' biological functions are closely related to their structures, which are formed under the interactions such as hydrogen bonding, ionic interaction, van der Waals force, and hydrophobic interaction. Experimentally, tools such as X-ray crystallography, NMR spectroscopy, and Cryo-electron microscopy, have been used to explore these specific spatial conformations. Theoretically, different scales of representations and multiscale models are employed. Due to the constant effort,

Protein Data Bank, which is the major resource for experimentally-determined structures of proteins, nucleic acids, and complex assemblies, now stores about 97 thousand structures. Hundreds of software packages are designed based on the theoretical models to evaluate the physical properties of the proteins.

Despite the progress in the studies of the protein structure, the mechanism behind how the polypeptide coils into its native conformation remains largely elusive. This is mainly due to the complexity and the stochastic dynamics involved in the process. So far, experimental tools such as atomic force microscopy, laser optical tweezers, and biomembrane force probe, can only shed lights on some stable intermediate structures. Steered molecular dynamics pushed one step further and can simulate some possible folding pathways.

However, the cost to run nonlinear dynamics can be prohibitively high. Since the protein functions are largely determined by the shape, which in turn depends on the proximity information between atoms, an efficient geometric and topological approach may be an effective alternative to give first-order estimates.

1.5 Contributions and Proposed Methods

Facing those challenges discussed in previous sections, we first propose a new compact and comprehensive data structure to support the geometric modeling problems for large complex shapes. Our data structure provides an efficient way to store all the required combinatorial maps for darts in volume meshes and a straightforward way of attaching attributes to k-cells ($k \in \{0,1,2,3\}$). Our data structure also has a constant time complexity access to incidence/adjacency information, including *face-edge* incidence. We show that it can also be extended to higher dimensions.

Built on the efficiency of our data structure, many testified geometric modeling techniques are implemented to model and analyze large complex models, specifically adapted for PDB and EMDB data. Although there is a large amount of existing research work on each of processing phrases of those data, the field lacks the information on comparisons among various algorithms and the proper combinations of the optimal choices into a coherent framework, which can be easily adapted

to specific tasks. We constructed such a framework, with proven correctness and efficiency in theories and experiments for processing, visualizing and analyzing the data. Due to the complexity in the data size, geometric and topological properties, providing such an integrated framework of efficient numerical algorithms may greatly benefit the whole biomolecular community. The geometric modeling techniques introduced in our framework are based on the latest advancements in computational geometry, applied mathematics and medical image processing.

With the readily available geometry processing tools to prepare the surfaces, we then address the problems of analyzing topological features of the complex shapes. First, we give a mathematical definition of choking loops as the narrowing of inside/outside volumes through persistent homology [56]. Both our definition of choking loops and the associated algorithm to compute them are based on the measurement of the life span of each non-contractible loop or membrane when the mesh is incrementally built starting from a single vertex to the full-blown volume. We first detect the topological features through detecting their "seed" faces, and then trace the boundary of such faces through the volume back to the original surfaces to determine the final shape of the choking loops. As a sample application, we explore a novel protein stability estimate based on the number of such loops.

To summarize, the main contributions of our work include,

- an efficient (in both space and time) and comprehensive data structure to store and perform computation on volume meshes;
- a toolkit incorporating many efficient geometric modeling algorithms on top of our mesh data structure for modeling large complex shapes, mainly on PDB and EMDB data in biomolecular science.
- an efficient method to detect geometry-aware topological features for large complex shapes, and its application in molecule stability.

Chapter 2

BACKGROUND

The primary target of modeling and analyses in this thesis is large and complex shapes. Such a shape can be considered as a set of points, sometimes referred to as a *space*. A generic space cannot be effectively handled due to its lack of structure. One important structure that can be endowed to spaces is *geometry*, providing continuous measurements such as length, angle, area, and curvature, all of which are invariant under a chosen set of transformations (such as rigid motion in Euclidean geometry). A more stable structure, invariant even under deformation, can be described through *topology*, leading to robust discrete measurements, such as the genus of a surface (number of holes). We also use the concept of persistent homology in computational topology, which can provide continuous measurements for topological structures. As an example illustrating the geometric, topological, and persistent homology structures, we perform different operations on a cup with a handle: translating and rotating the cup does not alter its geometry, e.g. surface curvatures; stretching the cup or deforming it smoothly into a donut shape does not change its topology, e.g. the number of holes; and offsetting the surface of the cup gradually can provide an offset distance, at which the hole is filled.

In the following, we provide the background of the geometric and topological concepts employed in our modeling techniques. We also discuss their discretization, including polygonal mesh representations of shapes, curvature estimation, and persistent homology theory, which relates the topological features with different spatial resolutions.

2.1 Introduction to Differential Geometry of Surfaces

In practice, most geometric models, including mechanical parts, objects in virtual reality, and biomolecular shapes, are treated as surfaces embedded in three-dimensional (3D) Euclidean space, as they form the boundary of non-degenerate 3D objects. Mathematically, they are described as

2-manifolds, which are spaces locally similar to 2D linear spaces. For a thorough discussion, refer to [128].

2.1.1 Tangent Plane and Normals

For simplicity, we first assume that the surface can be represented as the set of points, where a function S defined a 3D domain is a given constant m, i.e. the surface is $\{(x,y,z)|S(x,y,z)=m\}$. It is *smooth* if the gradient $\nabla S = (S_x, S_y, S_z)$ of any point on surface is continuous and non-singular ($\|\nabla S\| \neq 0$). Consider any curve that lies on the surface $c(t) = ((x(t), y(t), z(t)))$ passing through a given point P. As the points of the entire curve also lie on the surface, the surface equation stands. Thus, taking derivative of surface equation with regard to t at P leads to

$$(S_x, S_y, S_z) \cdot (x(t)', y(t)', z(t)') = 0. \tag{2.1}$$

This means that the gradient of S at P is perpendicular to the tangent of any curve. Thus, all tangents of such surface curves span a plane perpendicular to the gradient at P. This plane is called the *tangent plane* at P. The *normal* of the surface at P is defined as the unit normal of the tangent plane,

$$\mathbf{n}(P) = \nabla S / \|\nabla S\| \tag{2.2}$$

2.1.2 Curvatures

Curvatures describe the rate of change of the normal field near a surface point P when moving along tangent directions. These measurements determine the local shape, since any surface with the same curvatures can be locally approximated by a quadric surface with the same curvatures to second order accuracy (in terms of the distance from P). For smooth 2D surfaces, it can be represented as a two-by-two matrix, i.e., the Jacobian of the normal field with respect to motions in the 2D tangent plane.

If we choose an orthonormal coordinate frame in the plane, there are only two invariants under the rotation of the frame within the plane, namely Gaussian and mean curvatures (determinant and one half of the trace of the aforementioned Jacobian, respectively).

In the following, we first give a brief overview on how the curvature characterizes the local shape, and then discuss the calculation of curvatures through the concepts of first fundamental form and second fundamental form [10, 35, 181].

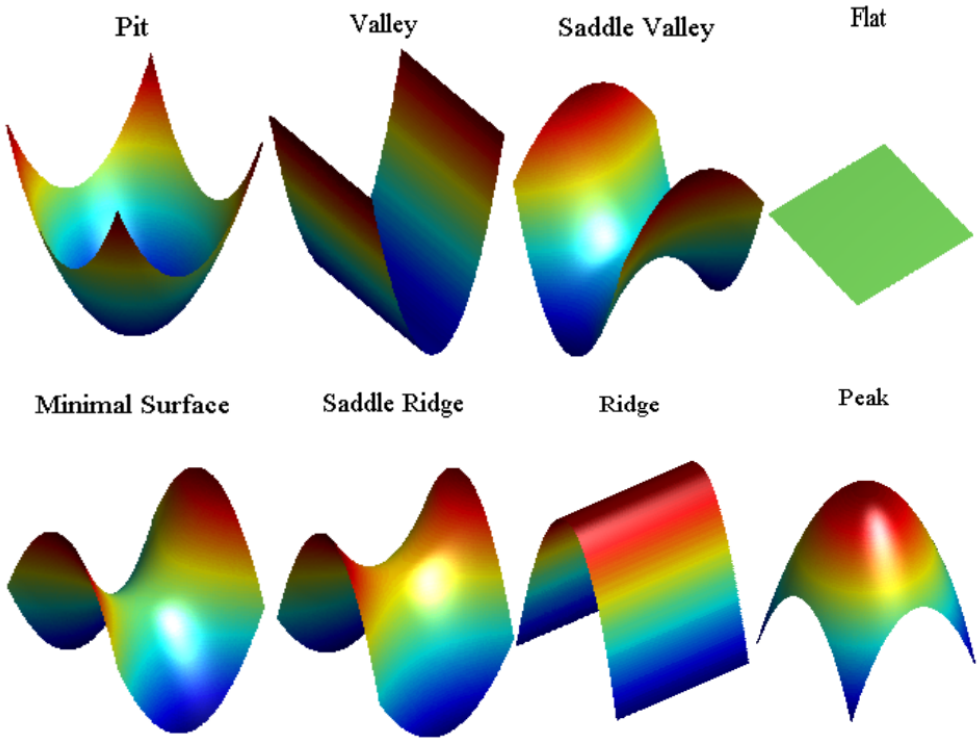

Figure 2.1: Representative image gallery of surface types based on signs of Gaussian curvature and mean curvature listed in Table 2.1.

2.1.2.1 Curvature as a Shape Descriptor

The local shape of a surface patch can be depicted by curvatures. A detailed description of curvatures in terms of differential geometry theory can be found in [10]. The curvature for a point on a curve represents how fast the tangent direction turns, or more precisely, the magnitude of the second derivative of the curve in its arc-length parameterization. For a point on the surface, one

Table 2.1: Surface types based on signs of Gaussian curvature and mean curvature as illustrated in Fig. 2.1.

	$K>0$	$K=0$	$K<0$
$H>0$	Peak	Ridge	Saddle ridge
$H=0$	None	Flat	Minimal surface
$H<0$	Pit	Valley	Saddle valley

can create a planar curve through the intersection of the surface and the local plane spanned by the surface normal and a tangent direction. The curvature of this planar curve is called the normal curvature along the chosen tangent direction. We can denote the maximum curvature among these normal curvatures by κ_1, and the minimum curvature by κ_2. These two curvatures are called *principal curvatures*, and the tangent directions associated with them are called *principal directions*. Note that these two directions are always orthogonal to each other. It can be further shown that the normal curvature κ along an arbitrary direction can be determined by κ_1 and κ_2 (the principal curvatures) and the angle θ that the chosen tangent direction makes with the maximum curvature direction

$$\kappa = \cos^2(\theta)\kappa_1 + \sin^2(\theta)\kappa_2. \tag{2.3}$$

The second order approximation of the neighborhood around a point is a quadric surface patch completely determined by the two principal curvatures, up to a global translation and rotation. To see this, we describe the neighborhood of a point on the surface by the deviation of the surface from the tangent plane, i.e. a height function $z = f(x,y)$ in a local coordinate system with the origin aligned to the point and the xy-plane aligned to the tangent plane. Such a height function always exists due to the implicit function theorem applied to $S(x,y,f(x,y)) = m$. The second order approximation of f is

$$z = \frac{1}{2}\begin{pmatrix} x & y \end{pmatrix} \mathbf{Hess} \begin{pmatrix} x \\ y \end{pmatrix} + o(d^3), \tag{2.4}$$

where $d = \|(x,y)\|$ is the distance from the center point to the projection of the surface point, and **Hess** is the Hessian (the symmetric second derivative matrix) of f,

$$\mathbf{Hess} = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}. \tag{2.5}$$

The actual shape of the second order approximation depends only on the eigenvalues of **Hess**, because applying a rotation in the tangent plane can diagonalize the Hessian. Thus we can align two local approximation shapes through a rotation, as along as the diagonalized Hessians are the same. By the definition of curvature, one can immediately see that these eigenvalues are $-\kappa_1$ and $-\kappa_2$. Here, we follow the convention in which bending towards the normal indicates a negative curvature, and bending away from the normal indicates a positive curvature. In this way, the curvatures for spheres will be positive. Note that some authors use the opposite sign.

Alternatively, it is often advantageous to use the Gaussian curvature and the mean curvature defined by

$$K = \kappa_1 \kappa_2, \tag{2.6}$$

$$H = \frac{1}{2}(\kappa_1 + \kappa_2). \tag{2.7}$$

where K is the Gaussian curvature, and H is the mean curvature. They correspond to, respectively, the determinant and half of the trace of the above Hessian matrix, which is another way of prescribing the rotation invariants.

Based on the signs of the Gaussian curvature and the mean curvature, the neighborhood of a surface point can be roughly classified as one of the eight different shapes, namely, pit, valley, saddle ridge, flat, minimal surface, saddle valley, ridge, and peak. In Table 2.1, we specify the type of shapes for each possible combination of signs. The actual shapes can be found in Fig. 2.1.

Considering the quadratic approximations they represent, we can see that local shapes with opposite signs of mean curvatures (indicating concave and convex pairs) and same signs of Gaussian curvatures may fit together.

To give intuitive descriptions of the shape, another pair of continuous invariants are sometimes used, namely, the shape index s and the curvedness c as defined in [102]

$$s = -\frac{2}{\pi}\arctan\left(\frac{\kappa_1+\kappa_2}{\kappa_1-\kappa_2}\right) \tag{2.8}$$

$$c = \sqrt{\frac{1}{2}(\kappa_1^2+\kappa_2^2)}. \tag{2.9}$$

Here s describes the relation between the principal curvatures and c describes how non-flat the shape is.

2.1.2.2 First Fundamental Form

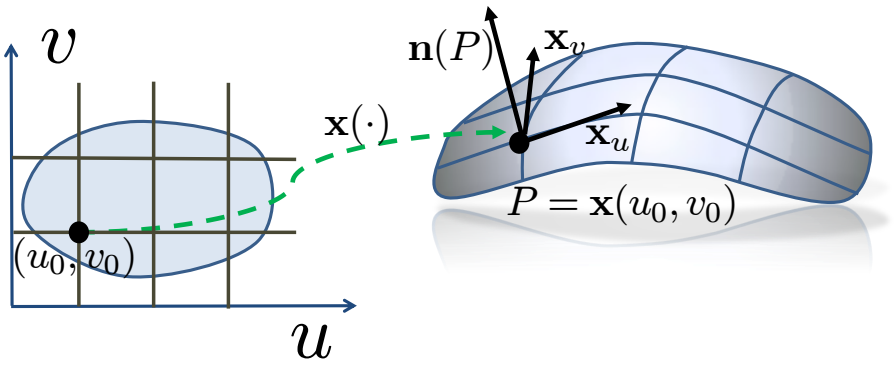

Figure 2.2: A parameterization of the surface patch shown to the right.

A generic surface patch M is often described by a parameterization mapping 2D regions to 3D Euclidean space (Figure 2.2),

$$\mathbf{x}(u,v) = (x(u,v), y(u,v), z(u,v))^T \tag{2.10}$$

We can construct a basis $(\mathbf{x}_u, \mathbf{x}_v)$ for the tangent space $T_P M$ spanned by the two tangent vectors at point P. Here, we have

$$\mathbf{x}_u = \frac{\partial \mathbf{x}}{\partial u}, \text{ and } \mathbf{x}_v = \frac{\partial \mathbf{x}}{\partial v}. \tag{2.11}$$

The first fundamental form is a quadratic form describing inner product of tangent vectors through the inner products between the basis vectors (namely, $\mathbf{x}_u$ and $\mathbf{x}_v$)

$$\begin{aligned} E &= \mathbf{x}_u \cdot \mathbf{x}_u, \\ F &= \mathbf{x}_u \cdot \mathbf{x}_v = \mathbf{x}_v \cdot \mathbf{x}_u, \\ G &= \mathbf{x}_v \cdot \mathbf{x}_v, \end{aligned} \tag{2.12}$$

The above equations can be written in a matrix form,

$$I_P = \begin{pmatrix} E & F \\ F & G \end{pmatrix} \tag{2.13}$$

Through inner product, the first fundamental form provides a way to measure distance-related quantities on surface M, such as length, angle and area. Let du and dv be infinitesimal changes in u and v direction respectively in the UV parameter domain. For a point $P(u_0, v_0)$ on surface, we have Taylor's expansion at the first order approximation

$$\mathbf{x}(u_0 + du, v_0 + dv) = \mathbf{x}(u_0, v_0) + \mathbf{x}_u du + \mathbf{x}_v dv. \tag{2.14}$$

The length induced by (du, dv) on surface would be

$$\begin{aligned} ds &= \sqrt[2]{(\mathbf{x}_u du + \mathbf{x}_v dv) \cdot (\mathbf{x}_u du + \mathbf{x}_v dv)} \\ &= \sqrt[2]{E du^2 + 2F du dv + G dv^2} \\ &= \sqrt[2]{(du, dv) I_P (du, dv)^T}. \end{aligned} \tag{2.15}$$

The same analysis also works for area. The area of a parallelogram with corners (u_0, v_0), $(u_0 + du, v_0)$, $(u_0, v_0 + dv)$ and $(u_0 + du, v_0 + dv)$ can be approximated by

$$\begin{aligned} dA &= \|\mathbf{x}_u du \times \mathbf{x}_v dv\| \\ &= \sqrt[2]{EG - F^2}\, du dv \\ &= \sqrt[2]{g}\, du dv, \end{aligned} \tag{2.16}$$

where $g = \det(I_P)$ is the Gram determinant.

2.1.2.3 Second Fundamental Form

On the tangent plane, another quadratic form, called the second fundamental form, describes the derivatives of the normal field. The unit normal vector associated with point P on M can be determined by the cross product of the basis

$$\mathbf{n}(P) = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|}. \tag{2.17}$$

By defining the normals, we introduce a fundamental concept called the Gauss map $\mathbf{n}(\cdot)$ of the

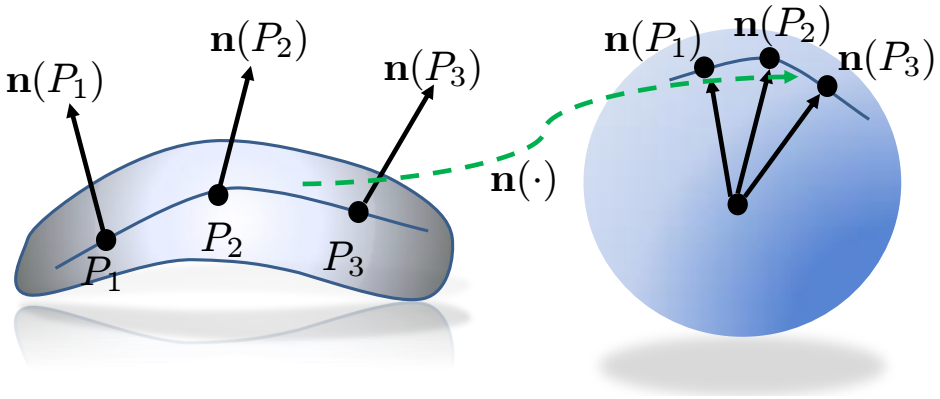

Figure 2.3: The Gauss map from the surface patch to the unit sphere.

surface M, which maps each point P on the surface M to the unit normal $\mathbf{n}(P)$ at the point, seen as a point on the unit sphere. It encodes all the geometric information related to the local shape around a point. As the normal at a point on the unit sphere centered at the origin is a vector identical to the point itself, the corresponding points on the surface and the unit sphere share the same normals. Figure 2.3 illustrates the concept of the Gauss map. We show the image of a curve on the surface patch under the Gauss map, along with the images of three sample points on that curve.

For instance, under the Gauss map, all the points of a flat plane are mapped to a single point on the unit sphere. The points on a cylinder will be mapped to a circle. The tangent planes of the point and of the image under the map are parallel to each other. By using the Taylor expansion, we have

$$\mathbf{n}(u_0 + du, v_0 + dv) = \mathbf{n}(u_0, v_0) + \mathbf{n}_u du + \mathbf{n}_v dv, \tag{2.18}$$

A tangent vector $w = (du, dv)^T$ of M is mapped to a tangent vector $\mathbf{n}_u du + \mathbf{n}_v dv$ of the sphere under Gauss map, both of which can be regarded as in the same tangent plane. We rewrite the mapping for the tangent vectors as the derivative of the Gauss map

$$d\mathbf{n}(w) = \mathbf{n}_u du + \mathbf{n}_v dv, \tag{2.19}$$

where $\mathbf{n}_u$ and $\mathbf{n}_v$ are the images of the two basis vectors $\mathbf{x}_u$ and $\mathbf{x}_v$, resp., on the tangent plane, which can be expressed in the basis of the tangent plane itself

$$\begin{aligned} \mathbf{n}_u &= a\mathbf{x}_u + c\mathbf{x}_v, \\ \mathbf{n}_v &= b\mathbf{x}_u + d\mathbf{x}_v. \end{aligned} \tag{2.20}$$

Thus, Eq. 2.19 in the matrix form representing the mapping from $T_P M$ to $T_P M$ with the basis $(\mathbf{x}_u, \mathbf{x}_v)$ is

$$d\mathbf{n} \begin{pmatrix} du \\ dv \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} du \\ dv \end{pmatrix}. \tag{2.21}$$

Expressing the terms using inner products of the basis vectors on the surface and the sphere, we can see that

$$\begin{aligned} \begin{pmatrix} a & b \\ c & d \end{pmatrix} &= \begin{pmatrix} E & F \\ F & G \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{n}_u \cdot \mathbf{x}_u & \mathbf{n}_u \cdot \mathbf{x}_v \\ \mathbf{n}_v \cdot \mathbf{x}_u & \mathbf{n}_v \cdot \mathbf{x}_v \end{pmatrix} \\ &= \begin{pmatrix} E & F \\ F & G \end{pmatrix}^{-1} \begin{pmatrix} L & M \\ M & N \end{pmatrix}, \end{aligned} \tag{2.22}$$

where L, M and N are defined as

$$\begin{aligned} L &= \mathbf{n}_u \cdot \mathbf{x}_u, \\ M &= \mathbf{n}_u \cdot \mathbf{x}_v = \mathbf{n}_v \cdot \mathbf{x}_u, \\ N &= \mathbf{n}_v \cdot \mathbf{x}_v. \end{aligned} \tag{2.23}$$

The second matrix on the right hand side is a symmetric bilinear form (quadratic form) in the tangent plane called the second fundamental form, which encodes the local shape variation around

a point on surface:

$$II_P = \begin{pmatrix} L & M \\ M & N \end{pmatrix}. \tag{2.24}$$

2.1.2.4 Gaussian Curvature and Mean Curvature

One can immediately verify that the two eigenvalues of the *shape operator* $d\mathbf{n} = I^{-1}II$ provides the principal curvatures in the surface patch, previously defined using the local height field at each surface point. The eigenvectors associated with eigenvalues are the principal directions. The formulas for Gaussian and mean curvatures can be directly expressed through the fundamental forms:

$$K = \frac{LN - M^2}{g} = \det(I_P^{-1} II_P), \tag{2.25}$$

$$H = \frac{2FM - EN - GL}{2g} = \frac{\text{trace}(I_P^{-1} II_P)}{2}. \tag{2.26}$$

The characterization of the local surface shape through this pair of curvatures is already shown in Table 2.1, which illustrates the common surface types by the signs of their associated Gaussian curvature and mean curvature values.

2.2 Discrete Surfaces and Local Shape Descriptors

We now discuss the discretization of the differential geometry of surfaces. We focus on the discrete representation of 3D shapes and the curvature estimates on their surfaces. See [86] for a general introduction on discrete differential geometry.

2.2.1 Discrete Representation of Surface Data

There is a multitude of representations of 3D objects. There are three main categories as follows:

2.2.1.1 Regular Grid Data

A large number of data generated from the scientific computing community [152, 115, 34] are stored as real-valued functions, where each discrete value sits on a regular grid cell or grid point in 2D or 3D, e.g. volumetric medical images. Each grid cell records the density or intensity of the measured physical parameters within itself. Usually the cell shapes are squares (in 2D) or cubes (in 3D). They store essentially the regular samples of the afore-mentioned function $S(x, y, z)$. They are often the input data format for geometry processing and analysis, acquired from raw device output or simulation data. It is extremely flexible in the sense that arbitrary topological change can be accommodated complicated data structure support. However, the storage requirement and temporal complexity of the algorithms relying on this type of representation may be intractable for large datasets.

2.2.1.2 Point Clouds Data

Another popular form of data to represent the shape is point clouds, which are simply sets of sample points on the boundary surface of 3D objects. Each sample point is usually stored simply by its X, Y, and Z coordinates. Such datasets are often the raw output of 3D scanners or range sensors. Some additionally include the surface normal at each sample point. Such point cloud representations are used in 3D manufactured part reconstruction, quality inspection and visualization of scenes with massive objects [176, 1, 133]. As a lot of geometric modelling techniques cannot directly apply to point clouds data, it is usually converted to other digital formats, at least locally through methods such as moving least squares [106]. In most cases, the entire surface dataset is converted to a polygonal mesh described below, through a surface reconstruction process for wide applicability.

2.2.1.3 Polygonal and Polyhedral Mesh Data

A third class of commonly-used data formats are meshes, which can be regarded as the final results of gluing a set of basic geometric elements subject to rules for forming a well-defined structure

called cell complex. Polygonal and polyhedral meshes can be built in this fashion by constructing the topology through connecting basic building blocks through incidence relationship among vertices, edges, faces and cells, while storing the geometry as the 3D spatial locations of the vertices. The edges can be either directed or undirected. The faces are polygons, including the commonly used triangles, quadrilaterals, and hexagons. Since the building blocks are often very simple by their nature, the meshes are very suitable for rendering, editing and geometric processing and analysis purposes, especially when the topology is stable.

2.2.2 Discrete Curvature Estimates on Triangle Meshes

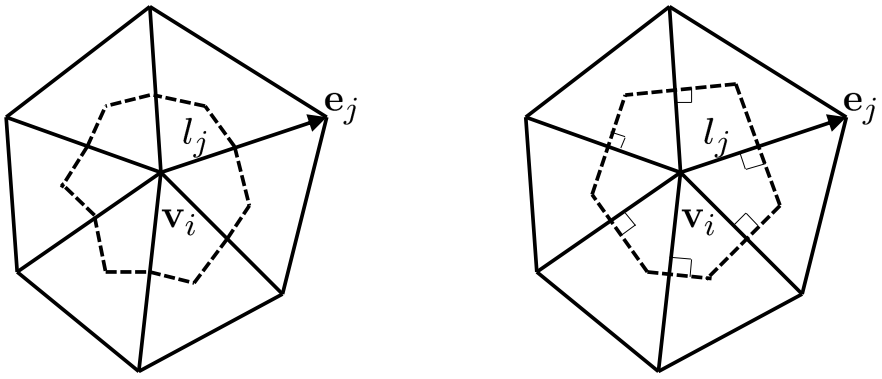

Figure 2.4: An illustration of dual cells defined around a vertex. Left: The area of the barycentric dual cell around a vertex (the cell formed by connecting consecutive barycenters of the triangles and edges incident to the center vertex $\mathbf{v}_i$), here l_j is the length of the part of edge $\mathbf{e}_j$ inside the neighborhood; Right: The area of the Voronoi dual cell of a vertex (the region containing all points closer to the center vertex $\mathbf{v}_i$ than to any other vertices).

For clarity, we only introduce the curvature analysis on triangle meshes, loosely following the notation in [46]. Estimation of curvature on Cartesian grid is discussed in 5.2.4.2. We first examine the discretization of the Gaussian curvature. The direct evaluation on a piecewise-flat triangle mesh would lead to Dirac-like distribution of Gauss curvature. Thus, a better estimate is obtained by an average over a small region. To compute the average, we first evaluate the integral of Gaussian curvature over the small region. This integral and the integral of the geodesic curvature (deviation of a surface curve from a geodesic curve, or a locally shortest curve) over its boundary sum up to 2π, according to the Gauss-Bonnet theorem. For a triangle mesh, the integral of the geodesic

curvature along the dual loop around a vertex (e.g., the loop in Fig. 2.4) is the same as the sum of the tip angles of triangles containing that vertex. Thus, the Gaussian curvature integral for a dual cell around the vertex is often estimated by the angle defect (or angle deficit), i.e. the difference between 2π and the sum, which is also called the Gauss-Bonnet scheme. To get a point-wise estimate, we can divide it by the area of the neighborhood around the vertex, as shown in Fig. 2.5.

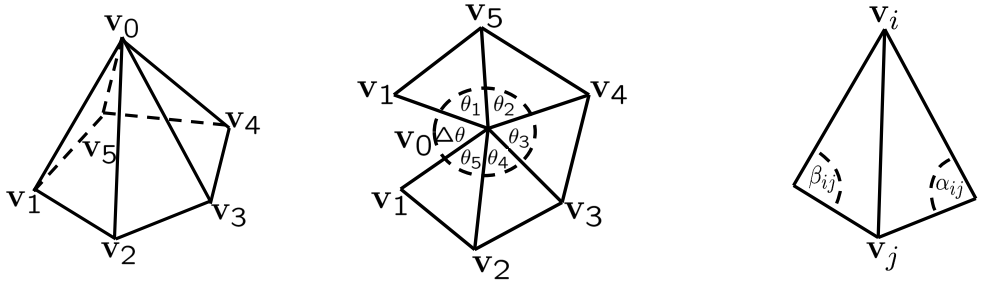

Figure 2.5: Schematic illustration of curvature algorithms. Left: A typical "one-ring" neighborhood of a vertex ($\mathbf{v}_0$); Middle: Flattening the one-ring by "cutting open" along the edge $\mathbf{v}_0\mathbf{v}_1$, we can measure the *angle deficit* used in Gaussian curvature estimates, denoted here by $\triangle\theta = 2\pi - \sum_{i=1}^{5} \theta_i$. Right: Angles used in the cotangent formula for the Laplace-Beltrami estimate of mean curvature.

There are a few different ways of determining which neighborhood area to use around the chosen vertex [46]. Once the area A_i is chosen for vertex i, the discrete estimates of the Gaussian curvature is formulated as follows

$$K_i \;=\; \frac{1}{A_i}\left(2\pi - \sum_{\theta_j \in \Theta_i} \theta_j\right), \tag{2.27}$$

where K_i is the estimated Gaussian curvature at vertex i, θ_j is the angle of triangle j at vertex i, Θ_i is a collection of all angles around vertex i as shown in Fig. 2.5. Fig. 2.4 shows a few choices of A_i, which is usually one of the dual cell areas (Voronoi dual, barycentric dual, and their mixture), those surrounded by the dashed edges in the figure. As shown in [46], the Voronoi dual cell area guarantees the least estimated errors for meshes with non-obtuse triangles. The straightforward estimation suggested by the authors for the Voronoi cell area around a vertex on mesh is

$$A_i = \frac{1}{8}\sum_{\mathbf{v}_j \in N_i} (\cot\alpha_{ij} + \cot\beta_{ij})\|\mathbf{v}_i - \mathbf{v}_j\|^2, \tag{2.28}$$

where N_i is the collection of all vertices immediately adjacent to vertex $\mathbf{v}_i$, a.k.a. the *one-ring* neighborhood as shown in Fig. 2.4. For one-ring neighborhoods containing obtuse angles, a modification called mixed area can be applied [46]. In practice, the formula using Voronoi dual area produces better results even when there are negative cotangents.

The average mean curvature for the neighborhood around a vertex is often estimated from the mean curvature normal, which is the product of H and $\mathbf{n}$. It also starts with the integral in a neighborhood region followed by a division of the area. As in the continuous theory, the mean curvature normal is computed by using the Laplace-Beltrami operator applied to the surface description [46], which is essentially an estimate of the trace of the Hessian of the local description of the surface as the distance field from the tangent plane. Intuitively speaking, the mean curvature normal is equal to the gradient of area, which represents the per unit area change around a surface point when a small perturbation is added onto the location of the point,

$$H\mathbf{n} = \lim_{A \to 0} \frac{\nabla A}{A}, \tag{2.29}$$

where A is the small area around the point on surface. It could be estimated either by barycentric dual cell area (one third of the sum of the neighbor triangles' area) or the Voronoi dual cell area (the area of the region containing points closer to the vertex than to any other vertices).

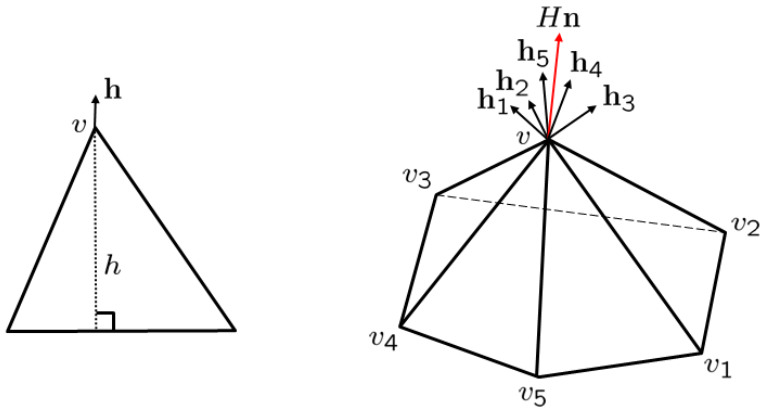

Figure 2.6: Mean curvature normal as rate of area change.

Following the area minimization concept, the mean curvature normal can be assembled for

each vertex on the mesh from the area gradient for each neighboring triangle. Figure 2.6 is an example showing this procedure to compute the mean curvature normal on a triangle mesh. The left chart is a triangle with the top vertex v and height h. The fastest way to change the triangle area fixing the bottom two vertices is by changing v along vector $\mathbf{h}$ direction. This is the gradient of the triangle area with respect to the change of vertex v. Under the same argument for the subset of the triangle mesh in the right chart, each triangle around a vertex v has its fastest direction to change the area. The weighted sum of these vectors, which is the mean curvature normal integrated in the neighborhood, is the fastest way to change the total area around vertex v. The right part of the figure shows the red mean curvature normal computed around a vertex with five neighbor triangles. The final discretized mean curvature (H_i) value at vertex i can be expressed as the *cotangent formula* [46]:

$$H_i\mathbf{n}_i = \frac{1}{4A_i}\sum_{\mathbf{v}_j\in N_i}(\cot\alpha_{ij}+\cot\beta_{ij})(\mathbf{v}_i-\mathbf{v}_j), \tag{2.30}$$

where $H_i\mathbf{n}_i$ is the mean curvature normal, and $\mathbf{n}_i$, the normalized version of the right hand side, is one commonly-used estimate for the unit surface normal at vertex i. Here A_i is the area controlled by the vertex i and N_i is the set of neighboring vertices of vertex i. Moreover, $\mathbf{v}_i$ and $\mathbf{v}_j$ are the coordinates of vertex i and j, and α_{ij} and β_{ij} are the opposite angles of the same edge in the two triangles incident to the edge. The angles used in the cotangent formula are the same as those used to compute the Voronoi area, as illustrated in Figure 2.5 right.

If an estimated curvature tensor (shape operator $I^{-1}II$) is required, a commonly used approach is to take the average of the curvature tensor evaluated on the edges inside a certain neighborhood [39]

$$\mathbf{C}(\mathbf{v}_i) = \frac{1}{A'_i}\sum_{\mathbf{e}_j\in E(\mathbf{v}_i)}\beta(\mathbf{e}_j)l_j\bar{\mathbf{e}}_j\bar{\mathbf{e}}_j^T, \tag{2.31}$$

where $\mathbf{C}(\mathbf{v}_i)$ is the estimated curvature tensor at vertex i expressed as a symmetric 3×3-matrix in the global Euclidean coordinates, and A'_i is the area of a specific neighborhood, for which the common choices include, for example, the intersection of the surface with a sphere of a give radius

around i, a geodesic disk on the surface around vertex i, or the one-ring of vertex i. Here $E(\mathbf{v}_i)$ is the set of all edges intersecting the neighborhood around vertex i, $\beta(\mathbf{e}_j)$ is the signed dihedral angle between the normals of the faces sharing edge $\mathbf{e}_j$ (negative when the faces bend towards the surface normal and positive otherwise), $\bar{\mathbf{e}}_j$ the unit direction along $\mathbf{e}_j$ (choosing either orientation of the edge will result in the same tensor), $(\cdot)^T$ denotes the matrix transpose operation, and l_j is the length of the part of edge $\mathbf{e}_j$ inside the neighborhood. To find two principal curvatures and two principal directions, one may perform an eigen-decomposition of $\mathbf{C}(\mathbf{v}_i)$

$$\mathbf{C}(\mathbf{v}_i) = \kappa_1 \mathbf{t}_1 \mathbf{t}_1^T + \kappa_2 \mathbf{t}_2 \mathbf{t}_2^T + \varepsilon \mathbf{n}\mathbf{n}^T,$$

where the eigenvalue ε with the smallest absolute value is always nearly 0, and the associated eigenvector $\mathbf{n}$ is an estimate of the local surface normal; the other two eigenvalues κ_1 and κ_2 are the principal curvatures, and their associated eigenvectors $\mathbf{t}_1$ and $\mathbf{t}_2$ are the two principal directions in the tangent plane. In an arbitrarily chosen frame for the tangent plane, e.g. $(\mathbf{x}_u, \mathbf{x}_v)$, the shape operator is the 2×2-matrix,

$$I^{-1}II = (\mathbf{x}_u, \mathbf{x}_v)^T (\kappa_1 \mathbf{t}_1 \mathbf{t}_1^T + \kappa_2 \mathbf{t}_2 \mathbf{t}_2^T)(\mathbf{x}_u, \mathbf{x}_v).$$

The larger the chosen neighborhood is, the less accurate the result is. However, choosing an overly small neighborhood results in noisy estimates when the resolution of the mesh is low.

2.3 Homology and Persistence

Instead of giving a generic overview of the continuous topology and talk about discretization, we start by directly defining topological structures on meshes. In particular, we focus on a algebraic topology concept called homology. On continuous surfaces, one may construct a concept called singular homology, instead of the simplicial homology that we use on the discrete meshes. However, the two are isomorphic to each other [42]. Thus, we only need to discuss the meshes. However, the homology groups are abstract abelian groups, which may not be robust or providing continuous measurements. Thus, we discuss the persistent homology theory, developed independently by

[138, 56], which provides a continuous measurement for measuring the persistence of topological structures, enabling both quantitative comparison and resilience to noises.

2.3.1 Simplex and Simplicial Complex

To describe the simplicial homology, we first present a formal description of the meshes mentioned in the discrete representation of surfaces and volumes. They are essentially a decomposition of the shape into elementary pieces called *simplices*.

2.3.1.1 Simplex

The *simplices* are the simplest polytopes in a given dimension, as detailed below. Let $v_0, v_1, ..v_p$ be $p+1$ vertices in a linear space. A p-simplex σ_p is the convex hull of those $p+1$ vertices, denoted as $\sigma_p = convex\{v_0, v_1, ..., v_p\}$ or shorten as $\sigma_p = \{v_0, v_1, ..., v_p\}$. A common requirement is that σ_p does not degenerate into the convex hull of a proper subset of these vertices. A more formal definition can be given as,

$$\sigma_p = \{v \mid v = \sum_{i=0}^{p} \lambda_i v_i, \sum_{i=0}^{p} \lambda_i = 1, 0 \leq \lambda_i \leq 1, \forall i\} \tag{2.32}$$

The dimension of σ_p is p since v_i spans σ_p. The most commonly used simplices in 3D are 0-simplex for vertex, 1-simplex for edge, 2-simplex for face and 3-simplex for cell.

An m-face of σ_p is the m-dimensional subset of $p+1$ vertices, where $m \leq p$. For example, an edge has two vertices as its 0-faces and one edge as its 1-face. Since the number of non-empty subsets of a set with $p+1$ vertices is 2^{p+1}, there are $2^{p+1} - 1$ faces in σ_p in total. All the faces are proper except for σ_p itself. In a triangular mesh, there are only three types of simplices, vertex(0-simplex), edge(1-simplex) and triangle(2-simplex). In a tetrahedral mesh, there is an additional simplex type called tetrahedron (3-simplex). Note that the more general mesh can include cells other than simplices, such as hexahedron and pyramid, but we restrict our discussion to simplicial meshes.

Two p-simplices σ^i and σ^j are *adjacent* to each other if they share a common face.

The boundary of σ_p, denoted as $\partial\sigma_p$, is the sum of its $p-1$-dimensional faces. Its interior is defined as the set containing all other points, denoted as $\sigma - \partial\sigma_p$.

We define the boundary operator for each p-simplex spanned by vertices v_0 through v_p as

$$\partial_p\{v_0,...,v_p\} = \sum_{i=0}^{p} \{v_0,...,\hat{v_i},...,v_p\}, \tag{2.33}$$

where $\hat{v_i}$ indicates that v_i is omitted.

2.3.1.2 Simplicial Complex

With the simplices as the basic building blocks, we define a *simplicial complex K* as a finite collection of simplices that meet the following two requirements,

- Containment: Any face of a simplex from K also belongs to K.
- Disjoint interior: The intersection of any two simplices σ_i, σ_j from K is either empty or a face of both σ_i and σ_j.

2.3.2 Homology

A powerful tool in studying the topology is the homology, which maps certain shapes in the meshes into algebraic groups. For closed smooth 2D surfaces, the homology completely describes their topology. For 3D objects, the essential topological features are the connected components, tunnels and handles, and cavities, which are exactly what is described by 0th, 1st, and 2nd homology, respectively.

2.3.2.1 Chains

The shapes to be mapped to the homology groups are constructed from *chains* defined below. Given a simplicial complex (e.g., a tetrahedral mesh) K, which, roughly speaking, is a concatenation of p-simplices (convex hulls of $p+1$ vertices, including vertices for $p=0$, edges for $p=1$, faces for $p=2$, and tetrahedra for $p=3$), we define a p-chain $c = \sum_i a_i\sigma_i$ as a formal linear combination

of all p-simplices in K, where $a_i \in Z_{/2}$ is 0 or 1 and σ_i is a p-simplex. Under such a definition, 0-chain is a set of vertices, 1-chain is a set of line segments, 2-chain is a set of triangles.

We extend the boundary operator ∂_p for each p-simplex to a linear operator applied to chains, i.e. the extended operator meet following two conditions for linearity,

$$\begin{aligned} \partial_p(\lambda c) &= \lambda \partial_p(c), \\ \partial_p(c_i + c_j) &= \partial_p(c_i) + \partial_p(c_j), \end{aligned} \tag{2.34}$$

where c_i and c_j are both chains and λ is a constant, and all arithmetic is for modulo-2 integers, in particular $1+1=0$.

An important property of the boundary operator is the following composite operation,

$$\partial_p \circ \partial_{p+1} = 0, \tag{2.35}$$

which immediately follows from the definition. Take 2-chain $c = f_1 + f_2$ as an example, which represents a membrane formed by two triangles, shown in Figure 2.7. The boundary of c is a 1-chain, which turns out to be a loop,

$$\begin{aligned} \partial_2(c) &= \{v_1, v_2\} + \{v_2, v_3\} + \{v_3, v_1\} + \{v_3, v_2\} + \{v_2, v_4\} + \{v_4, v_3\} \\ &= \{v_1, v_2\} + \{v_3, v_1\} + \{v_2, v_4\} + \{v_4, v_3\} \end{aligned} \tag{2.36}$$

The boundary of this loop is thus

$$\begin{aligned} \partial_1 \circ \partial_2(c) &= \partial[\{v_1, v_2\} + \{v_3, v_1\} + \{v_2, v_4\} + \{v_4, v_3\}] \\ &= v_1 + v_2 + v_2 + v_4 + v_4 + v_3 + v_3 + v_1 = 0 \end{aligned} \tag{2.37}$$

2.3.2.2 Homology

Homology is built on the *chain complex*, which is the sequence $(C_1, C_2, \ldots, C_n)$, where C_p is the space of all p-chains:

$$\cdots \xrightarrow{\partial_{p+1}} C_p \xrightarrow{\partial_p} C_{p-1} \xrightarrow{\partial_{p-1}} \cdots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} \emptyset \tag{2.38}$$

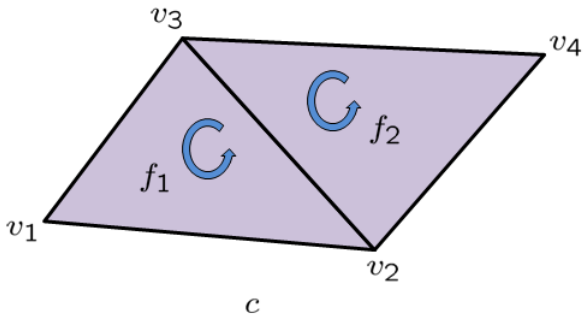

Figure 2.7: A sample 2-chain $c = f_1 + f_2$.

The p-chains in the kernel of the boundary homomorphisms ∂_p are called p-cycles (p-chains without boundary) and the p-chains in the image of the boundary homomorphisms ∂_{p+1} are called p-boundaries. The p-cycles form an abelian group (with group action being the addition of chains) called cycle group, denoted as $Z_p = Ker\ \partial_p$. The p-boundaries form another abelian group called boundary group, denoted as $B_p = Im\ \partial_{p+1}$.

Notice that $\partial_p \circ \partial_{p+1} = 0$, i.e., p-boundaries are also p-cycles (see Fig.2.8). As p-boundaries form a subgroup of the cycles group, the quotient group can be constructed through cosets of p-cycles, i.e by equivalent classes of cycles. The p-th homology, denoted as H_p, is defined as the quotient group,

$$\begin{aligned} H_p &= Ker\ \partial_p / Im\ \partial_{p+1} \\ &= Z_p / B_p, \end{aligned} \tag{2.39}$$

where p is the dimension.

Intuitively speaking, an element in the p-th homology group is an equivalent class of p-cycles. One of these cycles c can represent any other p-cycle that can be "deformed" through the mesh to c, because any other p-cycle in the same equivalence class differ with c by a p-boundary $b = \partial(\sigma_1 + \sigma_2 + \dots)$, where each σ_i is a $p+1$-simplex. Adding the boundary of σ_i has the effect of deforming c to $c + \partial\sigma_i$ by sweeping through σ_i. For instance, a 0-cycle v_i is equivalent to v_j if there is a path $\{v_i, v_{k1}\} + \{v_{k1}, v_{k2}\} + \dots + \{v_k n, v_j\}$. Thus each generator (basis) of the

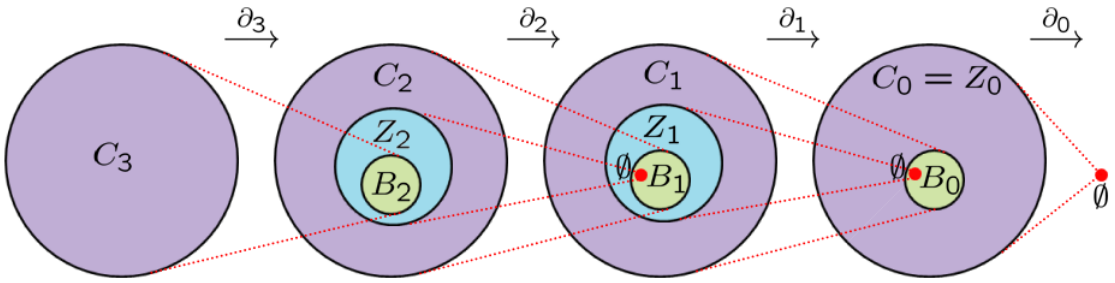

Figure 2.8: Homomorphism through the boundary operators among chain, cycle and boundary groups in 3D.

0-homology generators represents one connected component. Similarly, 1-cycles are loops, and 1st-homology generators (vectors in a basis for the linear space of 1-chains) represent independent nontrivial loops, i.e. separate tunnels; 2-homology generators are independent membranes, each enclosing one cavity of the 3D object.

Define $\beta_p = rank(H_p)$ to be the p-th Betti number. For a simplicial complex in 3D, β_0 is the number of connected components; β_1 is the number of tunnels; β_2 is the number of cavities. As H_p is the quotient group between Z_p and B_p, we can also compute the Betti numbers through,

$$rank(H_p) = rank(Z_p) - rank(B_p), \tag{2.40}$$

2.3.3 Persistent Homology

Homology generators identify the tunnels, cavities, etc. in the shape, but as topological invariants, they omit the metric measurements by definition. However, in practice, one often wants to compare the sizes of tunnels, for instance, to find the narrowest tunnel, or to filter out tiny tunnels as topological noises. Persistent homology is one method of reintroducing metric measurements to the topological structures [138, 56].

The measurement is introduced as an index i to a sequence of spaces $\{\mathbb{X}_i\}$. Such a family of

spaces form a *filtration*, if they are nested as follows,

$$\emptyset = \mathbb{X}_0 \subseteq \mathbb{X}_1 \subseteq \mathbb{X}_2 \subseteq \cdots \subseteq \mathbb{X}_m = \mathbb{X}. \tag{2.41}$$

Since each inclusion induces a mapping of chains, it induces a linear map for homology,

$$\emptyset = H(\mathbb{X}_0) \to H(\mathbb{X}_1) \to H(\mathbb{X}_2) \to \cdots \to H(\mathbb{X}_m) = H(\mathbb{X}). \tag{2.42}$$

The above sequence describes the evolution of the homology generators. We adopt the exposition in [125] and define a composition mapping from $H(\mathbb{X}_i)$ to $H(\mathbb{X}_j)$ as $\xi_i^j : H(\mathbb{X}_i) \to H(\mathbb{X}_j)$. A new homology class c is created (born) in $\mathbb{X}_i$ if it is not in the image of ξ_{i-1}^i. It is deceased (dead) in $\mathbb{X}_j$ if its image in $H(\mathbb{X}_j)$ is in the image of ξ_{i-1}^j, but its image in $H(\mathbb{X}_{j-1})$ is not in the image of ξ_{i-1}^{j-1}.

If we associate with each space $\mathbb{X}_i$ a value h_i denoting "time", we can define the duration, or the persistence of the each homology generator c as

$$persist(c) = h_j - h_i. \tag{2.43}$$

This measurement h_i is usually readily available when analyzing the topological feature changes. For instance, when the filtration arises from the level sets of a height function.

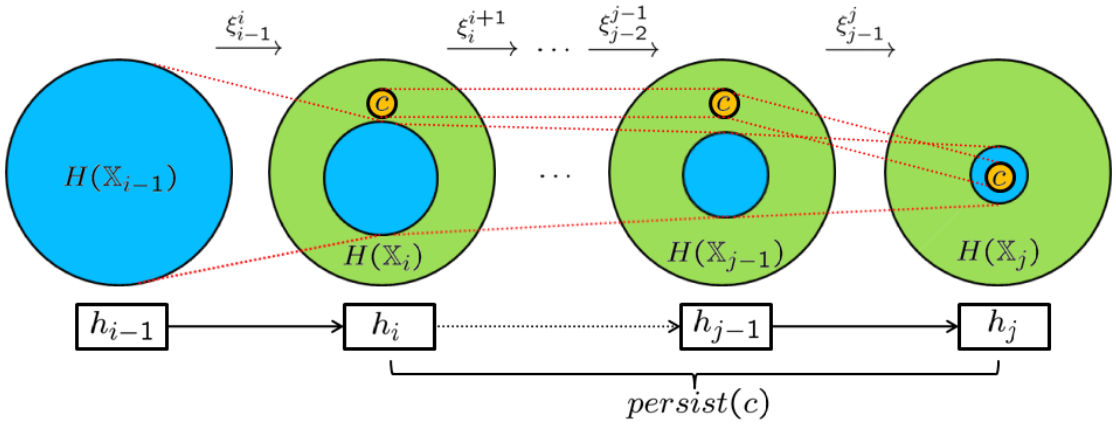

Figure 2.9: The birth and death of a homology generator c

Chapter 3

COMPACT COMBINATORIAL MAPS

3.1 Introduction

Volume meshes are now ubiquitous in solid modeling, physics-based simulation, computational science, and even rendering of translucent materials. However, the ever-increasing size and complexity of meshes impose undue stress on both memory access times and usage, especially since mesh size typically grows as a cubic function of the resolution. A data structure with small memory footprint that can efficiently handle queries of incidence and adjacency would thus benefit a wide range of applications in graphics and scientific computing in general.

We propose a novel compact data structure to meet the increasing demands for handling enormous size of volume meshes. While our data structure is based on the compact, array-based mesh data structure [3], we depart from their methods in several ways. With a simple but generic method for adding volume cell types into the data structure, our representation can define polyhedron types as required by the application. The concise local connectivity description of generic volume cell types is suitable for both file format and data structure. Our data structure also completes the data structure with a list of edges, and improves incidence queries within each volume cell.

Our main contributions include:

- a concise local connectivity description of generic 3-cell (volume cell) types, suitable for both file format and data structure;
- an efficient way to store all the required combinatorial maps for darts in volume meshes;
- a straightforward way of associating attributes to k-cells ($k \in \{0,1,2,3\}$); and
- a constant time complexity access to adjacency information, including *face-edge* incidence.

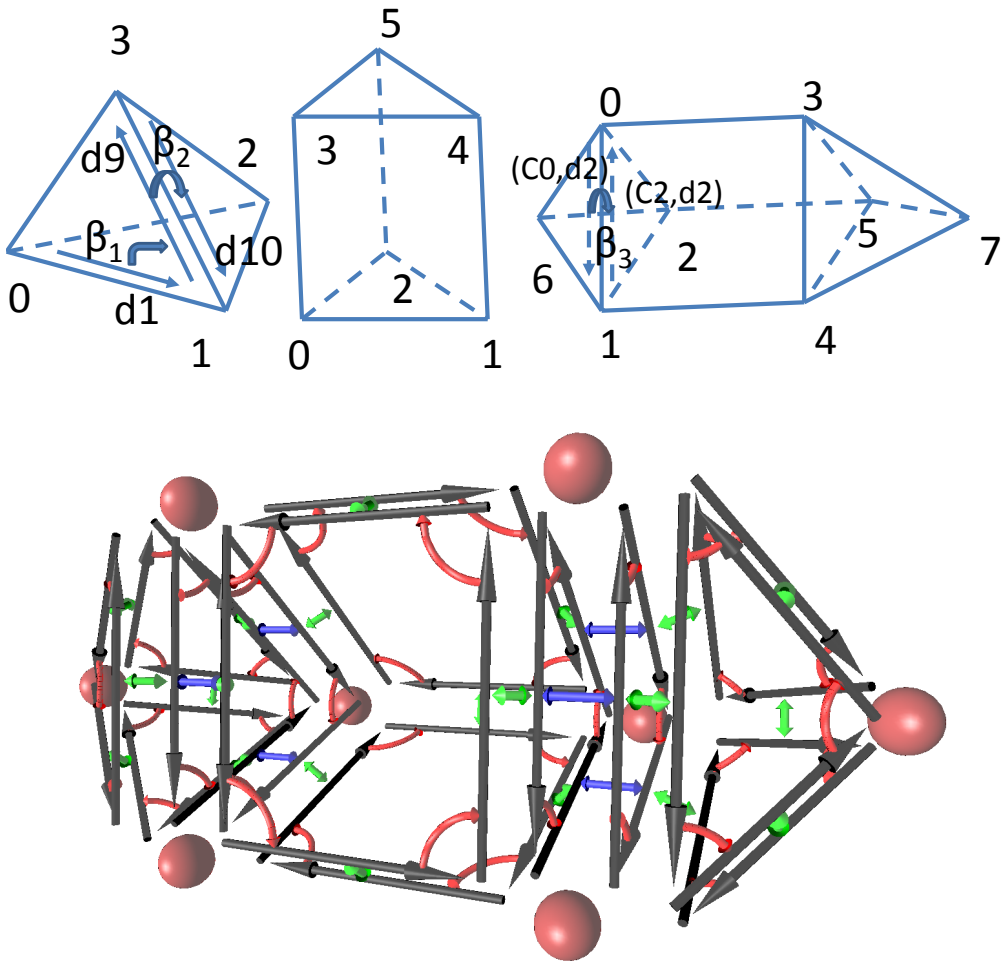

Figure 3.1: Upper: tetrahedron cell type; prism cell type; a mesh with 3 cells. Bottom: full set of combinatorial maps (β_1 in red, β_2 in green), and β_3 in blue) among darts. One example for each of the maps is given with the labels for the darts involved.

Note that unique edge identifiers and the face-edge incidence are the main missing components in the compact array-based mesh data structures [3] compared to our implementation. On the other hand, one can replace integer indices with memory pointers and use linked lists to make our data structure able to handle dynamic connectivity, at the cost of slightly increased memory usage, possible fragmentation and worse spatial consistency. Array-based data structure, however, are often more convenient in languages dedicated to scientific computing, such as FORTRAN. It is also easier to parallelize when distributed over several CPUs [3]. In fact, most of the aforementioned implementations provide the users with the option of using arrays and integers. Note that while we discuss in this chapter the details and implementation of our data structure to encode

orientable 3D manifolds, it can be generalized to orientable d-dimensional manifold meshes.

The rest of the chapter is organized as follows. In Sec. 2, we briefly introduce the combinatorial maps data structure for volume meshes. In Sec. 3, we describe our compact array-based data structure, and briefly analyze its space complexity. In Sec. 4, we discuss adjacency queries and show typical operations our data structures can efficiently handle, before concluding in Sec. 5.

3.2 Combinatorial Maps

In order to introduce the notion of combinatorial maps, we loosely follow the notation used in [43] and call k-dimensional cells *k-cells*. Hence, vertices are 0-cells, edges are 1-cells, faces are 2-cells, and volume cells (such as tetrahedra, prims, etc) are 3-cells. Two cells of different dimensions are said to be incident if one is a subset of the other. Two k-cells of the same dimension are adjacent if they share a common $(k-1)$-cell.

A combinatorial map describes the incidence and adjacency relations among cells of the mesh using a basic element called *dart*, and a group of relations between darts. For an orientable 3D manifold, a 3D dart corresponds to a cell tuple (v,e,f,c), where v is a starting vertex of an edge e that lies in a face f of 3-cell c. For 2D orientable surfaces, a 2D dart would be the same as the usual half-edge.

An abstract way to define a whole 3D combinatorial map M is to use a tuple $M = (D, \beta_1, \beta_2, \beta_3)$, with:

- D is a finite set of darts;
- for $i = 1,2,3$, $\beta_i : D \to D$ is a mapping;
- β_1 is a permutation;
- β_2, β_3, and $\beta_1 \circ \beta_3$ are involutions, i.e., $\forall d \in D$, $\beta_2 \circ \beta_2(d) = d$, $\beta_3 \circ \beta_3(d) = d$, and $(\beta_1 \circ \beta_3) \circ (\beta_1 \circ \beta_3)(d) = d$.

Intuitively speaking, β_i maps a dart to another dart with a different i-cell and a different vertex. If we identify the darts with (v,e,f,c) in the regular cell complex description, $\beta_1((v,e,f,c)) = (v',e',f,c)$, $\beta_2((v,e,f,c)) = (v',e,f',c)$, and $\beta_3((v,e,f,c)) = (v',e,f,c')$. Note that β_1 and β_2 are the 3D analogues of a half-edge's next() and opposite() operations, respectively.

In this abstract sense, we can define k-cells by orbits $\langle S\rangle(d)$, i.e., the set of darts that can be reached by arbitrary combination of maps $m \in S$:

- the 3-cell containing d is $\langle\{\beta_1,\beta_2\}\rangle(d)$;
- the 2-cell containing d is $\langle\{\beta_1,\beta_3\}\rangle(d)$;
- the 1-cell containing d is $\langle\{\beta_2,\beta_3\}\rangle(d)$,
- the 0-cell containing d is $\langle\{\beta_1\circ\beta_2,\beta_1\circ\beta_3\}\rangle(d)$.

3.3 Compact Data Structure

3.3.1 Overview

3.3.1.1 File Format

For a 2D polygonal mesh, the complete connectivity information can be encoded by a face list, with each entry corresponding to the list of vertices in the polygon face. However, for a polyhedral mesh, the same list of vertices can correspond to different polyhedra. For instance, an octahedron and a prism both have six vertices. As there are only a handful of k-cell types in most k-dimensional meshes used in practice, we opt to describe all the k-cell types in the header part of the file, and to describe each polyhedron by an ordered vertex list and its k-cell type.

3.3.1.2 Comprehensive Data Structure

All low dimensional ($\leq k-1$) relations $(\beta_1,\ldots,\beta_{k-1})$ map darts within the same k-cell. Given the type of a k-cell, we may assign each dart in that cell a local id, and the maps among the darts

can be precomputed when the k-cell type. One can easily assemble a global ID for each dart by (C,d), where C is the global ID of the k-cell, and d is the local dart ID. Additional auxiliary local incidence mapping to increase efficiency can also be created for each k-cell type at a constant memory cost (independent of the mesh size).

β_k maps a dart in one k-cell C_1 to another dart in an adjacent k-cell C_2. Noticing the relation among β's, we only store β_k for one dart in the common $k-1$-cell in C_1. Thus, the size of β_k can be reduced to one dart per pair of k-cell and $k-1$-cell.

The relation between k-cells and darts is implicitly given in the way we express a global ID for each dart (C,d). The mapping from darts to vertices (0-cells) is stored in the vertex lists for k-cells, also called the element connectivity in array-based methods such as [3], denoted $Cv2V$ below. The map from each vertex to one of its darts is stored in a table, denoted by $V2D$ below.

The above information enables constant time incidence/adjacency inquiries among vertices, k-cells, and "half"-$k-1$-cells, akin to [3] except some subtle differences. However, no unique IDs are actually given to 1-cells, 2-cells, through $k-1$-cells, hence no constant time incidence inquiries involving these cells can be achieved, without additional memory cost.

We propose to build a minimal set of additional connectivity tables to provide these incidence relations crucial to real world applications. We describe them as optional, since often one may only need some of the tables in this set, although at least one of them is, in many cases, indispensable. Here we restrict our discussion to 3D. To create a unique edge identifier we use a table called $E2D$, which maps a global edge ID to one of its darts. The map from darts back to edges can be implemented through a table $V2E$ mapping a vertex to the edge starting from it with the smallest ID, as elaborated below. Similarly, but less frequently required, we assign unique face IDs through the table $F2D$, and the backward mapping by $V2F$.

3.3.2 Details for 3D

To illustrate the detailed actual data structure, we use as a running example the description of the simple meshes shown in Figure 3.1, as found in a mesh file—skipping the list of vertex coordi-

nates since our focus is on connectivity information. As in the compact array-based half-face data structure (HFDS) [3], we leverage the fact that there are only a few types of cells typically used in engineering or graphics applications. However, unlike in HFDS, we will not limit ourselves to 3-cells used in the CFD General Notation System (tetrahedron, pyramid, prism, and hexahedron): any 3-cell type for which faces are locally defined can be specified in the header of a mesh file.

3.3.2.1 Local Information within Each 3-cell

Each 3-cell is treated locally as a 2-manifold cell complex, which can be represented by a local half-edge structure, i.e., a 2D combinatorial map. For a given type of 3-cell with n_v vertices, n_e edges, n_f faces:

- locally denote each vertex by v_i, with $i \in \{0, \ldots, n_v - 1\}$;
- locally label each face as $f_m = (v_i, v_j, v_k, \ldots)$, with $m \in \{0, \ldots, n_f - 1\}$.
- (optionally) locally label each of the $2n_e$ darts as $e_k = (v_i, v_j)$, with $k \in \{1, \ldots, 2n_e\}$;

Darts are indexed starting from 1, as 0 is reserved for boundaries.

The mesh file for Figure 3.1 would thus contain the information in Table 3.1, Table 3.2 and Table3.3.

Table 3.1: Cell type 0 (tetrahedron)

faces	0:(0,2,1)	1:(0,1,3)	2:(1,2,3)	3:(2,0,3)		
darts	1:(0,1)	2:(1,0)	3:(0,2)	4:(2,0)	5:(0,3)	6:(3,0)
	7:(1,2)	8:(2,1)	9:(1,3)	10:(3,1)	11:(2,3)	12:(3,2)

Table 3.2: Cell type 1 (prism)

faces	0:(0,2,1)	1:(0,1,4,3)	2:(1,2,5,4)	3:(2,0,3,5)	4:(3,4,5)	
darts	1:(0,1)	2:(1,0)	3:(0,2)	4:(2,0)	5:(0,3)	6:(3,0)
	7:(1,2)	8:(2,1)	9:(1,4)	10:(4,1)	11:(2,5)	12:(5,2)
	13:(3,4)	14:(4,3)	15:(3,5)	16:(5,3)	17:(4,5)	18:(5,4)

In all the tables we list, the information before ":" is for illustration purposes only, and is thus not stored in memory or files. For each 3-cell type, defining only the faces would be necessary and

Table 3.3: Sample file for the mesh in Figure 3.1

type 0	C0:(1,0,2,6) C1:(3,4,5,7)
type 1	C2:(0,1,2,3,4,5)

sufficient, since we can build the darts based on faces and give them labels. We then build a lookup table for β_1 and β_2 of all darts, with $2n_e$ entries and $2n_e$ possible values in the range for each entry. In our running example, the β_1 and β_2 tables for 3-cell type 0 are in Table 3.4.

Table 3.4: β_1 and β_2 tables for 3-cell type 0

d	1	2	3	4	5	6	7	8	9	10	11	12
$\beta_1(d)$	9	3	8	5	12	1	11	2	6	7	10	4
$\beta_2(d)$	2	1	4	3	6	5	8	7	10	9	12	11

Here the rows labeled β_1 and β_2 contain the images of the darts of the same column in the rows labeled with d, e.g. $\beta_1(1) = 9$ and $\beta_2(1) = 2$.

Assuming a small number of 3-cell types compared to mesh size, these type specifications only use a negligible amount of memory. In fact, storing all the *local* incidence and adjacency information directly for improved speed only requires an additional constant memory cost. We denote local incidence mappings as follows:

- $d2f(d)$ maps a dart d to its local face ID;
- $f2d(f,i)$ is the i-th dart of the local face f;
- $d2v(d)$ maps a dart d to its starting vertex.

We use lower (resp., upper) case in the name of a map to denote whether the index is local (resp., global).

3.3.2.2 Global Information

We load the connectivity table that contains, for each 3-cell, the global indices of its vertices. We denote this table by $Cv2V(C,v)$ since it maps the v-th vertex of 3-cell C to its global index V. Note that this corresponds to the usual way of storing the bare minimum connectivity information of 2D

polygonal meshes in files. Similarly, one can organize the file by listing vertex lists of every 3-cell in their order of enumeration; that is, the file first lists the descriptions of 3-cell types, followed by vertex lists of all 3-cells of the first type, the second type, etc. Once we have the 3-cell connectivity, a dart can be globally indexed by an ordered pair $D = (C, d)$, where C is the global 3-cell ID, and d is the local dart index. Note that instead of using a local face index with a starting vertex (called anchored half face) as in HFDS, we use local indices of darts; for the common case of tetrahedron meshes, this means we can cope with meshes twice as large for the same amount of memory.

To complete incidence and adjacency information in the combinatorial map, we need to construct β_3. We save space by noticing that $\beta_3 = \beta_1 \circ \beta_3 \circ \beta_1$, which means that $\beta_3(D)$ can be inferred if $\beta_3(\beta_1(D))$ is known. Thus, we only store β_3 for the *first* dart in each half face $H = (C, f)$, and denote this additional table by $H2D(C, f)$. If the application requires the use of boundary darts, their β_3 can be stored in a separate list $B2D(B)$, mapping the first dart of each boundary face B to its corresponding dart in the 3-cell adjacent to it. We also need to map from a vertex to one of its darts $V2D(v)$; but the map from a dart to its starting vertex is trivially found by $D2V(C,d) = Cv2V(C, d2v(d))$.

The tables for the 3-cell example are in Table 3.5.

Table 3.5: β_3, B2D and V2D tables

β_3	C0	f0d3:(2,8)	f1d1:(0,0)	f2d7:(1,0)	f3d4:(2,0)
	C1	f0d3:(2,16)	f1d1:(3,0)	f2d7:(4,0)	f3d4:(5,0)
	C2	f0d3:(0,8)	f1d1:(6,0)	f2d7:(7,0)	f3d4:(8,0)
		f4d13:(1,2)			
B2D	BF0:(0,1)	BF1:(0,7)	BF2:(0,4)	BF3:(1,1)	BF4:(1,7)
	BF5:(1,4)	BF6:(2,1)	BF7:(2,7)	BF8:(2,4)	
V2D	V0:(0,7)	V1:(0,1)	V2:(0,4)	V3:(1,1)	V4:(1,7)
	V5:(1,4)	V6:(0,10)	V7:(1,6)		

3.3.2.3 Boundary

The map β_3 usually returns an internal dart (C, d) with $d > 0$. However, if the opposite is a boundary dart, it will return $(B, 0)$, i.e., the boundary half-face ID. We carefully choose $V2D$ so

that whether a vertex V is on boundary can be determined by examining $\beta_3(V2D(V))$. Darts belonging to boundary half-face do not need to explicitly maintained in most cases .

3.3.2.4 Edge and Face Incidence Information

If we need to use a unique edge identifier, a table for $E2D(E)$ is maintained to map an edge to one of its darts. We sort the edges in the $E2D$ table by lexicographic order of their vertices (V_{start}, V_{end}) assuming that it always points from the vertex with a smaller index to the one with a larger index. A backward mapping $D2E$ can be implemented by a table $V2E(V)$, mapping vertex V to the first edge starting from it. We can avoid sorting the edges by using a linked list at the cost of storing another n_1 integers. The map $V2E$ would then be made to map a vertex to a linked list of edges starting from it.

If only half faces need identifiers, (C, f) can be used instead. Otherwise, a table $F2D(F)$ is required. Similar to the edge case, we can sort the faces by their first three vertices, assuming vertices are in ascending order within each face F. Then the backward mapping $D2F$ can be implemented by $V2F(V)$, mapping vertex V to the first face that has V as its smallest-indexed vertex.

For our running example, the (optional) edge tables are in Table 3.6.

Table 3.6: Optional edge tables E2D and V2E

E2D	E0(V0,V1):(0,2)	E1(V0,V2):(2,3)	E2(V0,V3):(2,5)
	E3(V0,V6):(0,9)	E4(V1,V2):(0,3)	E5(V1,V4):(2,9)
	E6(V1,V6):(0,5)	E7(V2,V5):(2,11)	E8(V2,V6):(0,11)
	E9(V3,V4):(2,13)	E10(V3,V5):(1,3)	E11(V3,V7):(1,5)
	E12(V4,V5):(2,17)	E13(V4,V7):(1,9)	E14(V5,V7):(1,11)
V2E	V0:0 V1:4 V2:7 V3:9 V4:12 V5:14 V6: V7:		

3.3.2.5 Example Table Construction

The construction of most tables is straightforward since the mesh connectivity information is complete. We only give an example of how to build $E2D$ in Algorithm 1. Note that the procedure

Algorithm 1 Build $E2D$ table

```
1: init flag table visited(), E ← 0
2: for all non-boundary dart D do
3:    if visited(D) then
4:       continue
5:    end if
6:    D_0 ← D
7:    while true do
8:       D' ← β_3 ∘ β_2(D) {rotate clockwise}
9:       if Boundary(D') or D' = D_0 then
10:         break
11:      end if
12:      D ← D'
13:   end while
14:   E2D(E) ← D, E ← E + 1, D_0 ← D
15:   repeat
16:      visited(D) ← true, visited(β_2(D)) ← true
17:      D ← β_2 ∘ β_3(D) {rotate counter-clockwise}
18:   until Boundary(D) or D = D_0
19: end for
```

ensures that a quick counter-clockwise traversal of the edge's one-ring is possible even when it is on the boundary, and an easy boundary test through $\beta_3 \circ \beta_2(E)$.

3.3.2.6 Spatial Complexity

Tetrahedron meshes are the easiest to establish comparisons between various data structures: for such meshes, we can approximate all k-cell counts n_k as a function of the number of tetrahedra n_3 and boundary faces n_b—other mesh types must be analyzed using the count of darts, and its estimated relation with k-cell numbers. Following [13], we assume the average valence of a vertex is around 4π divided by the solid angle for a vertex of an equilateral tet 0.5513, i.e., 22.8. Additionally, we assume that the average solid angles at boundary nodes are about half of the average angle. Based on these assumptions, the fact that each tetrahedron has 4 vertices and 4 faces, and Euler's formula, we have

$$22.8n_0 \approx 4n_3,\ 4n_3 + n_b = 2n_2,\ n_0 - n_1 + n_2 - n_3 \approx 0. \tag{3.1}$$

The k-cell counts are therefore

$$n_0 \approx 0.175 n_3,\ n_1 \approx 1.175 n_3 + 0.5 n_b,\ n_2 = 2 n_3 + 0.5 n_b. \tag{3.2}$$

For the models shown in 3.8, these estimates are very close to the actual k-cell counts. Some of the models are shown in Figure 3.2, with cross-sections revealing the internal tetrahedral structure. The memory usage for these models in OpenVolumeMesh and CGAL combinatorial maps data structures is listed in Table 3.9.

In the following analysis, we assume that the lowest four or more bits are sufficient to encode the local dart index or the local half face index; thus we need only one integer for (C,d) or (C,f). Alternatively, for tetrahedron meshes with fixed connectivity, we can use an integer D such that it represents $C = D/12$ and $d = D\%12$. When 3-cells are sorted by type, this method can be easily extended to cope with hybrid meshes and to include boundary darts. The memory size required for the various connectivity tables are listed in Tabel 3.7.

Table 3.7: Memory size required for the various connectivity tables.

Table	V2XYZ	Cv2V	H2D	V2D	B2D
Space	$3n_0$	$4n_3$	$4n_3$	n_0	n_b
Table(optional)		E2D	V2E	F2D	V2F
Space		n_1	n_0	n_2	n_0

By tallying up these numbers, we find that $8n_3 + n_0 + n_b \approx 8.175\,n_3 + n_b$ integers are required for the basic tables, in par with the basic eight pointers per tetrahedron (pointing to adjacent tetrahedra and corner vertices) plus one pointer per node (to one incident tetrahedron) used to encode connectivity in Pyramid and CGAL, and close to [17]'s tetrahedron mesh structure prior to difference code compression. Data structures capable of handling generic polytope meshes require more memory space when used for simplicial meshes, e.g., Dobkin and Laszlo's structure [50] would require around $18n_3$ pointers, while radial-edge, cell-tuple, and G-map representations, as well as CGAL's combinatorial map, would use even more memory. If unique edge identifiers are needed, we require $n_0 + n_1 \approx 1.35\,n_3$ additional integers, which is more compact than the pure tet mesh encoding of [17] before difference coding.

HFDS [3] uses the same amount of basic space ($8.175n_3 + n_b$). However, their encoding of a local dart (anchored face) identifier (C, f, v) uses a separate local index f for a face within the tetrahedron and a local index v of a vertex within the face. Thus, it would be less memory efficient when dealing with generic 3-cells, for example, 3-cells that have 5-edge faces or more. In addition, even in the common case of tetrahedron meshes, HFDS requires 5 bits for local indices ($f = 0$ is reserved for boundary), while we only need 4 bits, enabling us to handle meshes with 256M 3-cells with a 32-bit integer representation, instead of their 128M limit.

Furthermore, and *key to runtime efficiency*, we provide a simple way to give edges and faces unique identifiers. As we elaborate upon next, this enables constant time incidence queries, and allows appending attributes to edges and faces, which are important in simulation and other computational tasks. The HFDS data structure does not actually provide any means to get unique adjacent edge IDs in constant time.

3.4 Incidence/Adjacency Queries

As our data structure can be seen as an internal representation of a combinatorial map, it can directly leverage any implementation of combinatorial maps to get incidence and adjacency information in constant time. In addition, with integer IDs, additional attributes associated to vertices, darts, half faces, cells, edges, and faces, can be directly allocated as an array with the appropriate size, making it highly efficient and flexible for static meshes. We will first give a few examples of commonly-used neighborhood constructions such as one-rings in Algorithms 2 and 3 ('.' symbol denotes member access). Assuming constant maximum valence, both algorithms run in constant

Table 3.8: Actual memory usage for a variety of meshes.

model name	n_0	n_1	n_b	n_3	+V2XYZ	est.	+Edges
1mag	95,156	648,969	48,308	529,652	19,858KB	18,625KB	23,006KB
Armadillo	189,919	1,314,767	77,704	1,085,997	39,502KB	38,103KB	44,634KB
david	140,592	965,377	65,402	792,038	29,486KB	27,824KB	33,334KB
dc-wt	550,770	3,819,288	224,024	3,156,497	111,286KB	110,742KB	125,702KB
emd1590	23,419	150,930	19,540	117,736	5,346KB	4,175KB	6,110KB
fertility	341,924	2,385,564	125,450	1,980,912	70,098KB	69,438KB	79,490KB
neptune	358,647	2,498,975	133,476	2,073,588	73,622KB	72,695KB	83,442KB

Table 3.9: Actual memory usage for the same meshes as in Table 3.8 using OpenVolumeMesh library and CGAL's combinatorial maps, respectively.

model name	OpenVolumeMesh	CGAL CM
1mag	246,284KB	334,028KB
Armadillo	502,028KB	673,587KB
david	366,988KB	483,532KB
dc-wt	1,402,900KB	1,929,379KB
emd1590	54,696KB	67,789KB
fertility	885,876KB	1,205,862KB
neptune	921,616KB	1,258,291KB

Algorithm 2 One-ring darts and cells around vertex V_0

Ensure: {darts} and {cells} contain darts and cells in the one-ring.

$C \leftarrow (V2D(V_0).C)$
Queue $Q.push(C)$, save C in {cells}
while Q not empty **do**
  $C \leftarrow Q.pop()$
  $\{D_i\} \leftarrow$ all darts starting at V_0 in C
  save $\{D_i\}$ in {darts}
  for all D in $\{D_i\}$ **do**
    $C \leftarrow (\beta_3(D).C)$
    if C not in {cells} **then**
      $Q.push(C)$, save C in {cells}
    end if
  end for
end while

time. To map a dart to a unique edge ID, we find the end vertices (V_{start}, V_{end}) with $V_{start} < V_{end}$. We then perform a linear search in $E2D$ starting from $V2E(V_{start})$, this again would terminate in constant time.

In most cases, faces do not need a unique ID, as attributes are often associated to half faces; but if needed, our $F2D$ and $V2F$ tables can be used to provide a unique face ID.

All other incidence information can be similarly assembled from the mappings between cells and darts and the mappings among darts.

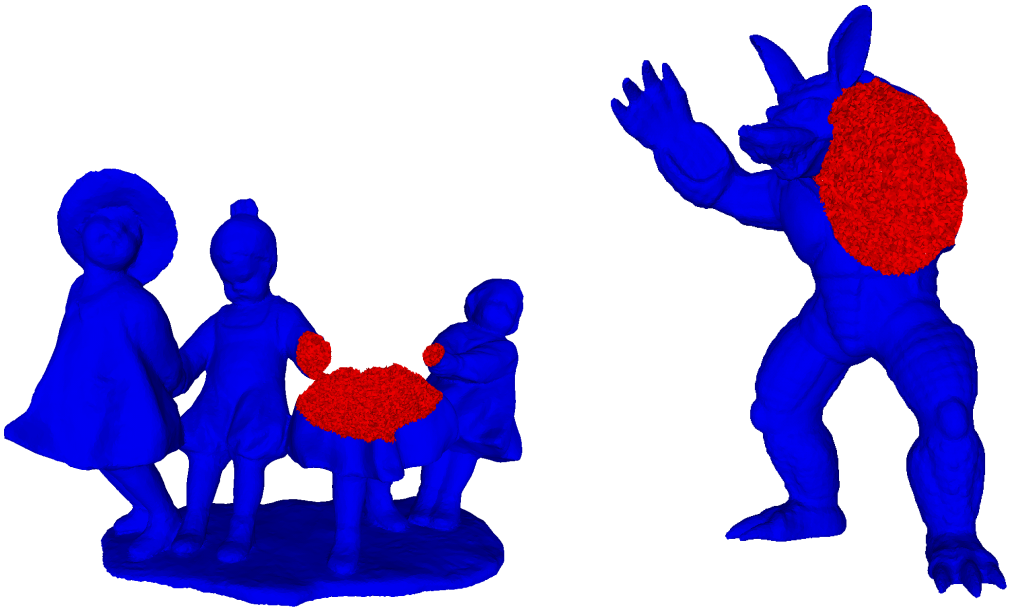

Figure 3.2: Some of the meshes in the statistics. The cross-sections reveal the internal tetrahedra. The surface triangles are rendered blue, and the internal triangles red.

3.5 Summary

We presented an efficient representation of combinatorial maps. All necessary components in combinatorial maps can be implemented in compact form. Compared to previous work, our data structure can handle arbitrary 3-cell types, and it provides adjacency and boundary inquiries in constant time. Appending attributes to *cells of any dimension* is also straightforward.

One limitation of the compact combinatorial map data structure we described is its apparent inability to deal gracefully with dynamically changing connectivity, in particular with possible changes of 3-cell types. (On the other hand, if 3D cells are kept intact as in the case of cutting or

Algorithm 3 One-ring (internal) HF around Edge E_0

Ensure: Array {HF} is the CCW ordered one-ring.

1: $D_0 \leftarrow E2D(E_0), D \leftarrow D_0$
2: **repeat**
3: $C \leftarrow (D.C), d \leftarrow (D.d)$
4: save $(C, d2f(d))$ and $(C, d2f(\beta_2(d)))$ in {HF}
5: $D \leftarrow \beta_2 \circ \beta_3(D)$ {rotate counter-clockwise}
6: **until** $Boundary(D)$ or $D = D_0$

merging meshes along faces, the mesh can be easily modified accordingly.) However, we believe that our data structure can be readily altered to efficiently handle connectivity changes as well: one could use pointers instead of integers for the IDs of 3-cells and vertices—and the last few bits of the pointer can actually be used to encode local dart index as in the integer case. The linked list version of $V2E$ will be necessary, increasing the memory space by $n_1 = 1.175\,n_3$.

Thus, a possible research direction worth exploring is the design of admissible local connectivity changes (such as edge removal or 2-3 flip) that maintain the validity of our compact data structure. Compression of neighboring information (β_3) using difference coding after sorting the cells along space-filling curves could also lead to further reduction of memory usage. Additionally, the extension to dimension $n > 3$ could be done by encoding the local connectivity $(\beta_1, \ldots, \beta_{n-1})$ of n-cell types, and store only β_n.

Another future work would be to explore the application of the data structure in tasks involving volume data, such as 3D field design and solid texturing [209, 205].

Chapter 4

GEOMETRIC MODELING ON BIOMOLECULAR MODELS — LAGRANGIAN REPRESENTATION

4.1 Introduction

One of major features of biological sciences in the 21^{st} century is their transition from an empirical, qualitative and phenomenological discipline to a comprehensive, quantitative and predictive one [186]. Indeed, theoretical description, mathematical modeling and computer simulation of biological systems have made enormous contribution to the present understanding of biological sciences. The material basis and fundamental underpinning of modern biological sciences are biological macromolecules, especially proteins and nucleic acids, which coil into specific three-dimensional (3D) shapes and are able to carry out most of the functions of cells. The goal of theoretical description, mathematical modeling and computer simulation is to understand the structure, function, dynamics and transport of biological macromolecules. A prerequisite to theoretical description, mathematical modeling and computer simulation of the structure, function, dynamics and transport of biological macromolecules is the geometric modeling based on their 3D shapes. In addition to straightforward geometric visualization, geometric modeling bridges the gap between imaging and mathematical modeling such that the structural information can be integrated into mathematical models [202].

The objective of this chapter is to explore the efficient computational methods for the geometric modeling of proteins, subcellular structures, organelles and large multiprotein complexes. Specifically, we study the reconstruction of biological structures from noisy 3D imaging data, examine the geometric representation of complex biological shapes, provide accurate calculation of surface areas and surface enclosed volumes, and investigate the computational algorithm and surface mapping of Gaussian and mean curvatures of macromolecules. Most geometric modeling is carried out in the Lagrangian representation with triangle meshes on the surface.

The rest of this chapter is organized as follows. Section 4.2 introduces the theory and formulation of variational multiscale models for macromolecular systems. We first briefly review the variational derivation of the mean curvature model which generates the minimal molecular surface(MMS). This variational approach is extended to include nonpolar and polar interactions. A density functional approach for ionic species is also discussed. Coupled governing equations are derived to describe multiresolution surfaces and associated electrostatic maps. Section 4.3 discusses computational methods and numerical algorithms for geometric modeling. We give a brief description of high order geometric PDEs and nonlinear PDE based high-pass filters. Different surface extraction schemes are discussed. Numerical algorithms for calculating surface areas and surface enclosed volumes are given in the Lagrangian representation. We introduce the state of the art techniques for volumetric meshing of subcellular structures, organelles and large multiprotein complexes. In Section 4.4, we show the results of our extensive numerical experiments to validate the proposed methods, algorithms, and schemes. We designed analytical cases to test accuracy and convergent order of the proposed algorithms for area, volume and curvature calculations. Second order convergence is found in these schemes. Finally, we apply the proposed methods to PDB and EMDB examples. Our results demonstrate the effectiveness, robustness and efficiency of the proposed approaches. This chapter ends with a summary in Section 4.5.

4.2 Theory and Models

In this section, we discuss the differential geometry based multiscale surface generation. The minimal molecular surface is constructed by using the variational principle applied to a surface free energy functional. When the nonpolar energy is considered, surface formation is governed by geometric and potential driven flows. For more realistic solvation process, multiscale models of the biomolecular system at equilibrium or non-equilibrium state are developed. Generalized Laplace-Beltrami, generalized Poisson-Boltzmann and generalized Nernst-Planck equations are derived to describe surface evolution, electrostatic potential distribution and charged species concentrations, respectively.

4.2.1 Minimal Molecular Surface

Minimal surfaces, such as the shapes of soap bubble films and of tensile membranes in architecture, are omnipresent in nature and man-made materials, as the result of surface free energy minimization to reach a stable equilibrium. Based on the energy minimization principle, the MMS is introduced to remove geometric singularities in traditional molecular surfaces, i.e., vdWS, SAS, and SES. Numerically, geometric singularities cause the computational instability. Physically, geometric singularities do not exist in biomolecular systems as atomic or molecular electron densities overlap.

In our variational models, a hypersurface function $S(\mathbf{r})$ is defined to describe the biomolecular surface. It is convenient to set $S(\mathbf{r}) = 1$ for the region inside the macromolecule and $S(\mathbf{r}) = 0$ for the solvent domain. Under the action of the Laplace-Beltrami flow described below, the hypersurface function $S(\mathbf{r})$ will gradually become continuous and carry the geometric shape of the biomolecule. The final MMS is obtained by iso-surface extraction from $S(\mathbf{r})$. We define γ as the surface tension and Area as the enclosed area of the biomolecular surface. The computational domain is represented by $\Omega \in \mathbb{R}^3$. The surface free energy can be expressed as [186]

$$G_{\text{surface}} = \gamma \text{Area} = \gamma \int_0^1 \int_{S^{-1}(c)\bigcap\Omega} d\sigma dc = \int_\Omega |\nabla S(\mathbf{r})| d\mathbf{r}, \quad \mathbf{r} \in \mathbb{R}^3, \tag{4.1}$$

where the coarea formula from the geometric measure theory [60] has been used to describe the surface area as a volume integral. The energy minimization process can be done through the Euler-Lagrange equation. By introducing an artificial time t, a generalized Laplace-Beltrami equation is obtained [10, 186, 34, 35],

$$\frac{\partial S}{\partial t} = |\nabla S| \left[\nabla \cdot \left(\gamma \frac{\nabla S}{|\nabla S|} \right) \right], \tag{4.2}$$

where $S = S(\mathbf{r}, t)$ depends on the artificial time t. The MMS can be extracted from the steady state solution under the constraint that the surface encloses vdWS[10].

4.2.2 Surfaces Derived from Nonpolar Solvation Analysis

The solvation process is of fundamental importance to the quantitative description and analyses of biomolecular systems, because almost all important biological processes, such as DNA replication, transcription and translation, protein folding, protein-protein interaction, and protein-ligand binding, occur in aqueous environment. Solvation free energy, which can be measured experimentally, is the major physical observable for a solvation process. Typically, solvation free energy consists of two parts, the polar contribution and the nonpolar contribution. The nonpolar energy can be further divided into three components, including surface energy, energy for creating a solute cavity in the solvent, and solvent-solute interaction [72, 74, 111, 73]

$$G_{\text{nonpolar}} = \gamma \text{Area} + p\text{Vol} + \int_{\Omega_s} U d\mathbf{r}, \quad \mathbf{r} \in \mathbb{R}^3, \tag{4.3}$$

where γ is the same surface tension as we mentioned above, "Area" and "Vol" are respectively the solute surface area and volume, p is the hydrodynamic pressure, and U denotes the solvent-solute non-electrostatic interactions. The integration is over the solvent domain Ω_s.

Usually, the solvent has multiple species. Therefore, the solvent-solute interaction potential U can be rewritten as the summation of all the interactions between the solvent species and the solute molecule,

$$U = \sum_{\alpha} \rho_\alpha U_\alpha, \tag{4.4}$$

where ρ_α is the density of the αth solvent component, and U_α is the interaction potential of the αth component of the solvent.

In the aqueous environment, each solvent species interacts with both solute and other solvent species. Especially for charged ions, they can form ion-water clusters and constantly influence each other. To take the general correlations into account, the interaction potential is further elaborated as,

$$U_\alpha = \sum_j U_{\alpha j}(\mathbf{r}) + \sum_\beta U_{\alpha\beta}(\mathbf{r}), \tag{4.5}$$

where $U_{\alpha j}$ is the interaction potential between the jth atom of the solute and the αth component of the solvent, and $U_{\alpha\beta}$ is the interaction potential between the αth and the βth components of the solvent. In principle, U_α can take any desirable form. In the past, the Lennard-Jones potential was used to approximate the solvent-solute non-electrostatic interactions [34, 35]. The solvent-solvent interaction can be represented by the van der Waals potential as well. The potential $U_{\alpha\beta}(\mathbf{r})$ can be expressed in an integral form,

$$U_{\alpha\beta}(\mathbf{r}) = \bar{\varepsilon}_{\alpha\beta} \int \rho_\beta(\mathbf{r}') \left[\left(\frac{\sigma_\alpha + \sigma_\beta}{|\mathbf{r} - \mathbf{r}'|} \right)^{12} - \left(\frac{\sigma_\alpha + \sigma_\beta}{|\mathbf{r} - \mathbf{r}'|} \right)^{6} \right] d\mathbf{r}' \tag{4.6}$$

where $\overline{\varepsilon}_{\alpha\beta}$ is the well-depth parameter, and σ_α and σ_β are the radii of the αth and βth solvent component.

Using the hypersurface function S defined in the previous section, the nonpolar energy can be expressed as

$$G_{\text{nonpolar}} = \int_\Omega \gamma |\nabla S(\mathbf{r})| d\mathbf{r} + \int_\Omega pS(\mathrm{r}) d\mathbf{r} + \int_\Omega (1 - S(\mathbf{r})) U d\mathbf{r}. \tag{4.7}$$

Note that the term $1 - S(\mathbf{r})$ is the indicator function of the solvent domain. By means of the Euler-Lagrange equation, we have

$$\frac{\delta G_{\text{nonpolar}}}{\delta S} \Rightarrow -\nabla \cdot \left(\gamma \frac{\nabla S}{|\nabla S|} \right) + p - U = 0. \tag{4.8}$$

With an artificial time, the above condition can be turned into the following generalized Laplace-Beltrami equation [10, 186, 34, 35]

$$\frac{\partial S}{\partial t} = |\nabla S| \left[\nabla \cdot \left(\gamma \frac{\nabla S}{|\nabla S|} \right) - p + U \right]. \tag{4.9}$$

The surface for nonpolar solvation models can be obtained by extracting a certain isovalue from the steady state solution of the above generalized Laplace-Beltrami equation.

4.2.3 Surfaces Derived from Full Solvation Analysis

In most situations, the solvation process involves also a polar contribution due to the electrostatic interactions. In the equilibrium state, the polar energy can be estimated based on the Poisson-Boltzmann theory. Since Sharp and Honing introduced the variational formulation of Poisson-Boltzmann theory in 1990 [154], several similar approaches have been discussed in the literature

[81, 53, 186]. The total polar solvation energy can be written as an integral equation. In this chapter, we modify the Boltzmann distribution of the αth solvent species as

$$\rho_\alpha = \rho_{\alpha 0} e^{-\frac{q_\alpha \Phi + U_\alpha - \mu_{\alpha 0}}{k_B T}}, \tag{4.10}$$

where k_B is the Boltzmann constant, T is the temperature, $\rho_{\alpha 0}$ and ρ_α respectively denote the reference bulk concentration and the concentration distribution of the αth solvent species, Φ is the electrostatic potential, and q_α denotes the charge valence of the αth solvent species, which is zero for an uncharged solvent component. The new term $\mu_{\alpha 0}$ is a relative reference chemical potential which reflects the difference in the equilibrium concentrations of different solvent species, i.e., $\rho_\alpha \neq \rho_\beta$, given that $\rho_{\alpha 0} = \rho_{\beta 0}$. In Section 4.2.4, it can be seen that Boltzmann distribution (4.10) occurs naturally as an equilibrium condition. Here U_α is the interaction potential of the αth component of the solvent as described in the Section 4.2.2.

With the new Boltzmann distribution formulation, the total polar energy functional can be represented as,

$$\begin{aligned} G_{\text{polar}} = \int_\Omega \Big\{ & S\Big[-\frac{\varepsilon_m}{2} |\nabla \Phi|^2 + \Phi\, \rho_m \Big] \\ & + (1-S)\Big[-\frac{\varepsilon_s}{2} |\nabla \Phi|^2 - k_B T \sum_\alpha \rho_{\alpha 0} \Big(e^{-\frac{q_\alpha \Phi + U_\alpha - \mu_{\alpha 0}}{k_B T}} - 1 \Big) \Big] \Big\} d\mathbf{r}, \end{aligned} \tag{4.11}$$

where Φ is the electrostatic potential, ε_s and ε_m are the dielectric constants of the solvent and solute, respectively, and ρ_m represents the fixed charge density of the solute. Specifically, one has the form of $\rho_m = \sum_j Q_j \delta(\mathbf{r} - \mathbf{r}_j)$, with Q_j denoting the partial charge of the jth atom in the solute.

The total solvation energy functional is the combination of polar energy (4.11) and nonpolar energy (4.7),

$$\begin{aligned} G^{\text{PB}}_{\text{total}}[S, \Phi] = \int_\Omega \Big\{ & \gamma |\nabla S| + pS + S\Big[-\frac{\varepsilon_m}{2} |\nabla \Phi|^2 + \Phi\, \rho_m \Big] \\ & + (1-S)\left[-\frac{\varepsilon_s}{2} |\nabla \Phi|^2 - k_B T \sum_\alpha \rho_{\alpha 0} \left(e^{-\frac{q_\alpha \Phi + U_\alpha - \mu_{\alpha 0}}{k_B T}} - 1 \right) \right] \Big\} d\mathbf{r}. \end{aligned} \tag{4.12}$$

We emphasize that the interactions $(1-S)U$ are not omitted here, but embedded in Boltzmann distribution. If we assume $k_B T >> q_\alpha \Phi + U_\alpha - \mu_{\alpha 0}$, the Boltzmann term can be approximated

by

$$-(1-S)k_BT\sum_\alpha \rho_{\alpha 0}\left(e^{-\frac{q_\alpha\Phi+U_\alpha-\mu_{\alpha 0}}{k_BT}}-1\right)\sim(1-S)\sum_\alpha \rho_{\alpha 0}\left(q_\alpha\Phi+U_\alpha-\mu_{\alpha 0}\right). \quad (4.13)$$

Therefore, the interactions have already been accounted for in our modified Boltzmann distribution. However, the decomposition of the total solvation energy into the polar and nonpolar parts is by no means unique. The interactions will influence the concentration distributions, especially for the charged species [186, 34].

Once the total energy functional in Eq. (4.12) has been determined, the variational principle is applied to derive governing equations,

$$\begin{aligned}\frac{\delta G^{\mathrm{PB}}_{\mathrm{total}}}{\delta S}\Rightarrow &-\nabla\cdot\left(\gamma\frac{\nabla S}{|\nabla S|}\right)+p-\frac{\varepsilon_m}{2}|\nabla\Phi|^2+\Phi\,\rho_m\\ &+\frac{\varepsilon_s}{2}|\nabla\Phi|^2+k_BT\sum_\alpha\rho_{\alpha 0}\left(e^{-\frac{q_\alpha\Phi+U_\alpha-\mu_{\alpha 0}}{k_BT}}-1\right)=0.\end{aligned} \quad (4.14)$$

$$\frac{\delta G^{\mathrm{PB}}_{\mathrm{total}}}{\delta\Phi}\Rightarrow\nabla\cdot([(1-S)\varepsilon_s+S\varepsilon_m]\nabla\Phi)+S\rho_m+(1-S)\sum_\alpha q_\alpha\rho_{\alpha 0}e^{-\frac{q_\alpha\Phi+U_\alpha-\mu_{\alpha 0}}{k_BT}}=0. \quad (4.15)$$

Equations (4.14) and (4.15) are obtained by the minimization of the total solvation free energy functional with respect to S and Φ, respectively. A coupled system, including a generalized Laplace-Beltrami equation and generalized Poisson-Boltzmann equation, is obtained,

$$\frac{\partial S}{\partial t} = |\nabla S|\left[\nabla\cdot\left(\gamma\frac{\nabla S}{|\nabla S|}\right)+V_1\right], \quad (4.16)$$

$$-\nabla\cdot(\varepsilon(S)\nabla\Phi) = S\rho_m+(1-S)\sum_\alpha q_\alpha\rho_{\alpha 0}e^{-\frac{q_\alpha\Phi+U_\alpha-\mu_{\alpha 0}}{k_BT}}, \quad (4.17)$$

where the potential driven term V_1 is given by

$$V_1=-p+\frac{\varepsilon_m}{2}|\nabla\Phi|^2-\Phi\,\rho_m-\frac{\varepsilon_s}{2}|\nabla\Phi|^2-k_BT\sum_\alpha\rho_{\alpha 0}\left(e^{-\frac{q_\alpha\Phi+U_\alpha-\mu_{\alpha 0}}{k_BT}}-1\right), \quad (4.18)$$

and $\varepsilon(S)=(1-S)\varepsilon_s+S\varepsilon_m$ is a generalized permittivity function. For the generalized Laplace-Beltrami, an artificial time is introduced as discussed in the earlier work [10, 186, 34, 35]. These coupled equations are called the Laplace-Beltrami and Poisson-Boltzmann (LB-PB) equations.

The numerical experiments demonstrated good predictions compared with the experimental results. Thus, this model can be used to describe the solvation at equilibrium.

The generalized potential in Eq. (4.5) takes into consideration of the interactions between solvent species and those between solvent and solute. Therefore, Eqs. (4.16) and (4.17) should be able to capture the detailed microstructural characteristics such as size effect and ionic double layer effect [12], as is the classical density functional theory [80].

4.2.4 Surfaces Derived from Charge Transport Analysis

Charge transport is a common phenomenon in complex physical, chemical, and biological systems and engineering devices, such as fuel cells, solar cells, battery cells, nanofluidics, transistors, and ion channels. These systems are usually far from equilibrium, and thus the models for the equilibrium state as we have discussed in the above section cannot be used. On the other hand, as a response to the perturbation, a nonequilibrium system might evolve towards the equilibrium driven by spatial gradients. In this section, a chemical potential related free energy is considered to describe multispecies mixing.

For simplicity, the flow stream velocity and chemical reaction are not considered. We define μ_α^0 as a reference chemical potential of the αth species at which the associated ion concentration is $\rho_{0\alpha}$ given $\Phi = U_\alpha = \mu_{\alpha 0} = 0$. With the consideration of the entropy of mixing and osmotic effect, the chemical potential related free energy is expressed as [69]

$$G_{\text{chem}} = \int_\Omega \sum_\alpha \left\{ -\mu_{\alpha 0}\rho_\alpha + k_B T \rho_\alpha \ln \frac{\rho_\alpha}{\rho_{\alpha 0}} - k_B T \left(\rho_\alpha - \rho_{\alpha 0}\right) \right\} d\mathbf{r}, \tag{4.19}$$

where $k_B T \rho_\alpha \ln \frac{\rho_\alpha}{\rho_{\alpha 0}}$ is the entropy of mixing, and $-k_B T \left(\rho_\alpha - \rho_{\alpha 0}\right)$ is a relative osmotic term [117]. The total free energy for a charge transport system can be expressed as the summation of

the nonpolar energy, polar energy and chemical related free energy,

$$
\begin{aligned}
G^{\mathrm{PNP}}_{\mathrm{total}}[S,\Phi,\{\rho_\alpha\}] = \int_\Omega \{&\gamma|\nabla S| + pS + (1-S)U \\
&+S\left[-\frac{\varepsilon_m}{2}|\nabla\Phi|^2 + \Phi\,\rho_m\right] + (1-S)\left[-\frac{\varepsilon_s}{2}|\nabla\Phi|^2 + \Phi\sum_\alpha \rho_\alpha q_\alpha\right] \\
&+(1-S)\sum_\alpha\left[-\mu_{\alpha 0}\rho_\alpha + k_BT\rho_\alpha \ln\frac{\rho_\alpha}{\rho_{\alpha 0}} - k_BT\left(\rho_\alpha - \rho_{\alpha 0}\right)\right]\Bigg\} d\mathbf{r}.
\end{aligned}
\tag{4.20}
$$

The total free energy functional (4.20) is a function of the surface function S, electrostatic potential Φ and the ion concentration ρ_α. By applying the variational principle with respect to S, Φ and ρ_α, one has

$$
\frac{\delta G^{\mathrm{PNP}}_{\mathrm{total}}}{\delta\rho_\alpha} \Rightarrow \mu_\alpha^{\mathrm{gen}} = -\mu_{\alpha 0} + k_BT\ln\frac{\rho_\alpha}{\rho_{\alpha 0}} + q_\alpha\Phi + U_\alpha = \mu_\alpha^{\mathrm{chem}} + q_\alpha\Phi + U_\alpha, \tag{4.21}
$$

$$
\frac{\delta G^{\mathrm{PNP}}_{\mathrm{total}}}{\delta\Phi} \Rightarrow \nabla\cdot\left([(1-S)\varepsilon_s + S\varepsilon_m]\nabla\Phi\right) + S\rho_m + (1-S)\sum_\alpha \rho_\alpha q_\alpha = 0, \tag{4.22}
$$

$$
\begin{aligned}
\frac{\delta G^{\mathrm{PNP}}_{\mathrm{total}}}{\delta S} \Rightarrow &-\nabla\cdot\left(\gamma\frac{\nabla S}{|\nabla S|}\right) + p - U - \frac{\varepsilon_m}{2}|\nabla\Phi|^2 + \Phi\,\rho_m \\
&+\frac{\varepsilon_s}{2}|\nabla\Phi|^2 - \Phi\sum_\alpha\rho_\alpha q_\alpha - \sum_\alpha\left[-\mu_{\alpha 0}\rho_\alpha + k_BT\rho_\alpha\ln\frac{\rho_\alpha}{\rho_{\alpha 0}} - k_BT\left(\rho_\alpha - \rho_{\alpha 0}\right)\right] = 0,
\end{aligned}
\tag{4.23}
$$

where $\mu_\alpha^{\mathrm{gen}}$ is the relative generalized potential of species α, and vanishes at equilibrium. Therefore, we have at equilibrium

$$
\rho_\alpha = \rho_{\alpha 0} e^{-\frac{q_\alpha\Phi + U_\alpha - \mu_{\alpha 0}}{k_BT}}. \tag{4.24}
$$

In case of nonequilibrium, Fick's first law says that the relative generalized potential $\mu_\alpha^{\mathrm{gen}}$ leads to ion fluxes $\mathbf{J}_\alpha = -D_\alpha\rho_\alpha\nabla\frac{\mu_\alpha^{\mathrm{gen}}}{k_BT}$ with D_α being the diffusion coefficient of species α. Fick's second law predicts the Nernst-Planck equation $\frac{\partial\rho_\alpha}{\partial t} = -\nabla\cdot\mathbf{J}_\alpha$. Together with the generalized Laplace-Beltrami equation and generalized Poisson equation obtained from the above Euler-Lagrange equations (4.23) and (4.22), a coupled system is obtained,

$$
\frac{\partial\rho_\alpha}{\partial t} = \nabla\cdot\left[D_\alpha\left(\nabla\rho_\alpha + \frac{\rho_\alpha}{k_BT}\nabla(q_\alpha\Phi + U_\alpha)\right)\right], \tag{4.25}
$$

$$
\frac{\partial S}{\partial t} = |\nabla S|\left[\nabla\cdot\left(\gamma\frac{\nabla S}{|\nabla S|}\right) + V_2\right], \tag{4.26}
$$

$$
-\nabla\cdot(\varepsilon(S)\nabla\Phi) = S\rho_m + (1-S)\sum_\alpha\rho_\alpha q_\alpha, \tag{4.27}
$$

where $q_\alpha\Phi + U_\alpha$ can be identified as a form of the potential of the mean field. Here, the external potential term V_2 is expressed as

$$\begin{aligned} V_2 &= -p + U + \frac{\varepsilon_m}{2}|\nabla\Phi|^2 - \Phi\,\rho_m - \frac{\varepsilon_s}{2}|\nabla\Phi|^2 + \Phi\sum_\alpha \rho_\alpha q_\alpha \\ &\quad + \sum_\alpha \left[k_B T \left(\rho_\alpha \ln \frac{\rho_\alpha}{\rho_{\alpha 0}} - \rho_\alpha + \rho_{\alpha 0} \right) - \mu_{\alpha 0}\rho_\alpha \right]. \end{aligned} \tag{4.28}$$

Note that the same technique of introducing the artificial time for the generalized Laplace-Beltrami equation is used. This coupled system is called the Laplace-Beltrami Poisson-Nernst-Planck (LB-PNP) model.

4.3 Methods

This section provides a variety of mathematical and computational methods for geometric modeling. The goal here is to introduce a repertoire of appropriate computational tools for the applications involving volumetric data and the shapes defined in cryo-EM datasets and PDB datasets.

4.3.1 Multiresolution Representations

Initial data downloaded from the Protein Data Bank (PDB) and Electron Microscopy Data Bank (EMDB) are used as inputs of the LB, LB-PB and/or LB-PNP models.

The coupled systems of LB-PB or LB-PNP are multiscale models. Partial charges in the protein molecules are explicitly described as point charges using Dirac delta functions. The charged species in the solvent are described in terms of concentrations, which either follow Boltzmann distributions or are governed by the Nernst-Planck equation. These different representations reduce the number of degrees of freedom and, at the same time, maintain certain accuracy. The multiscale surfaces can be extracted from the solution of the generalized Laplace-Beltrami equation. Appropriate initial conditions for the geometric flow equation can lead to multiresolution representations of different geometric and topological features.

For the generalized Laplace-Beltrami (LB) equation, the initial condition is set as an enlarged van der Waals surface in a 3D domain. Under the biological constraint, the hypersurface is evolved

according to the generalized LB equation. With appropriate preprocessing [32] of the data from the PDB, we obtain atom positions $\mathbf{r}_i = (r_{i,x}, r_{i,y}, r_{i,z}), i = 1, \cdots, n$, atom radii $r_i, i = 1, \cdots, n$ and also the atomic charge information. Here n is the total number of the protein atoms. It is useful to define two sets,

$$D_\chi = \cup_{i=1}^{n} \{\mathbf{r} : |\mathbf{r} - \mathbf{r}_i| < r_i\}. \tag{4.29}$$

and

$$D = \cup_{i=1}^{n} \{\mathbf{r} : |\mathbf{r} - \mathbf{r}_i| < \eta r_i\}, \tag{4.30}$$

where $\eta > 1$ is a parameter which can be adjusted to give different initial conditions. The initial value of $S(\mathbf{r}, t)$ is set to

$$S(\mathbf{r}, 0) = \begin{cases} 1 & \forall \mathbf{r} \in D \\ 0 & \text{otherwise} \end{cases} \tag{4.31}$$

We also set $S(\mathbf{r}, t) = 1 \ \forall \mathbf{r} \in D_\chi$ as a constraint. Usually, for the same number of iterations, a larger η parameter gives a “thicker" surface, which means that the fine structures are merged and a “coarser" representation of the molecular surface is obtained. A large η parameter can help us omit atomic details and focus on desirable molecular (global) features relevant to certain protein-protein interactions or protein-ligand bindings.

Another way to generate multiresolution representations is to adjust the number of iterations in solving the generalized LB equation. Instead of reaching the steady state, we stop the iteration earlier (i.e., selecting a finite total evolution time t_t in $S(\mathbf{r}, t_t)$). This procedure with different choices of t_t enables us to achieve different resolutions.

Yet another approach for multiresolution surfaces is to extract different iso-values (i.e., selecting C in $S(\mathbf{r}, t_t) = C$) of a given hypersurface function ($S(\mathbf{r}, t_t)$) as illustrated in our earlier work [183]. Typically, a lager C value gives rise to a higher resolution molecular surface, while a smaller C leads to highlighting global surface features.

The “coarse" resolution can be useful if one needs to capture some global characteristics of the protein, like holes, concave subdomains and convex regions. As the surface electrostatic distribu-

tion is calculated simultaneously, this multiscale multiresolution model can have a great potential in analyzing the protein-protein interaction and protein-ligand binding.

4.3.2 High Order Geometric Flows

Geometric flows, such as the Laplace-Beltrami flow, play a significant role in image analysis. An important aspect in the geometric flow development is the use of high order geometric PDEs for image processing or surface analysis. Willmore flow, proposed in 1920s, is a fourth order geometric PDE which locally minimizes the difference between two principal curvatures (see detailed description on principal curvatures in Section 2.1.2.1). Therefore, the Willmore flow prefers spherical shapes, which may be undesirable in general applications. Motivated by the hyperdiffusion in the pattern formation in alloys, glasses, polymer, combustion, and biological systems, Wei introduced the first family of arbitrarily high order geometric PDEs for edge-preserving image restoration in 1999, using Fick's law [184]

$$\frac{\partial u(\mathbf{r},t)}{\partial t} = -\sum_q \nabla \cdot \mathbf{j}_q + e(u(\mathbf{r},t), |\nabla u(\mathbf{r},t)|, t), \quad q = 0,1,2,\cdots \tag{4.32}$$

where the nonlinear hyperflux is given by

$$\mathbf{j}_q = -d_q(u(\mathbf{r},t), |\nabla u(\mathbf{r},t)|, t)\nabla\nabla^{2q}u(\mathbf{r},t), \quad q = 0,1,2,\cdots \tag{4.33}$$

where $\mathbf{r} \in \mathbf{R}^n$, $\nabla = \frac{\partial}{\partial \mathbf{r}}$, $u(\mathbf{r},t)$ is the processed image function, $d_q(u(\mathbf{r},t), |\nabla u(\mathbf{r},t)|, t)$ are edge sensitive diffusion coefficients and $e(u(\mathbf{r},t), |\nabla u(\mathbf{r},t)|, t)$ is a nonlinear operator. Equation (4.32) is subject to the initial image data $u(\mathbf{r},0) = X(\mathbf{r})$ and appropriate boundary conditions. The essential idea of Equation (4.32) is to accelerate the noise removal in the Perona-Malik equation [134] by higher order derivatives, which is more efficient in noise dissipation. As a generalization of the Perona-Malik equation, the hyperdiffusion coefficients $d_q(u, |\nabla u|, t)$ in Eq. (4.33) can also be chosen as the Gaussian form

$$d_q(u(\mathbf{r},t), |\nabla u(\mathbf{r},t)|, t) = d_{q0} \exp\left[-\frac{|\nabla u|^2}{2\sigma_q^2}\right], \tag{4.34}$$

where the values of constant d_{q0} depend on the noise level, and σ_0 and σ_1 were chosen as the local statistical variance of u and ∇u

$$\sigma_q^2(\mathbf{r}) = \overline{|\nabla^q u - \overline{\nabla^q u}|^2} \quad (q = 0, 1). \tag{4.35}$$

The notation $\overline{Y(\mathbf{r})}$ above denotes the local average of $Y(\mathbf{r})$ centered at position $\mathbf{r}$. The measure based on the local statistical variance is important for discriminating image features from noise. As a result, one can bypass the image preprocessing, i.e., the convolution of the noise image with a smooth mask in the application of the PDE operator to noisy images. High order geometric PDEs have found many practical applications [184, 116, 79, 78]. Arbitrarily high order geometric PDEs are modified for molecular surface formation and evolution [9]

$$\frac{\partial S}{\partial t} = (-1)^q \sqrt{g(|\nabla \nabla^{2q} S|)} \nabla \cdot \left(\frac{\nabla(\nabla^{2q} S)}{\sqrt{g(|\nabla \nabla^{2q} S|)}} \right) + P(S, |\nabla S|), \tag{4.36}$$

where S is the hypersurface function, $g(|\nabla \nabla^{2q} S|) = 1 + |\nabla \nabla^{2q} S|^2$ is the generalized Gram determinant and P is a generalized potential term, including microscopic interactions in biomolecular surface construction. When $q = 0$ and $P = 0$, Eq. (4.36) recovers the mean curvature flow used in our earlier construction of minimal molecular surfaces [10]. It reproduces the surface diffusion flow [9] when $q = 1$ and $P = 0$. It has been shown that surface generated with the fourth order geometric PDE demonstrates a morphology distinguished from that obtained with the mean curvature flow or the Laplace-Beltrami flow [9].

4.3.3 Nonlinear PDE Based High Pass Filters

Unfortunately, the studies of geometric flows have been essentially limited to the construction of nonlinear PDE based low-pass filters. From the point of view of image and signal processing, low-pass filtering is just one specific type of operations and other filters, such as high-pass filters and band-pass filters are equally important. An exception is the nonlinear PDE based high-pass filters introduced by Wei and Jia [188] for image edge detection in 2002,

$$u_t(\mathbf{r}, t) = F_1(u, \nabla u, \nabla^2 u, \ldots) + \varepsilon_u(v - u) \tag{4.37}$$

$$v_t(\mathbf{r}, t) = F_2(v, \nabla v, \nabla^2 v, \ldots) + \varepsilon_v(u - v) \tag{4.38}$$

where $u(\mathbf{r},t)$ and $v(\mathbf{r},t)$ are scalar fields, ε_u and ε_v are coupling strengths. Here F_1 and F_2 are general nonlinear diffusion operators, and can be chosen as the Perona-Malik operator $F_1 = \nabla \cdot d_1(|\nabla u|)\nabla$ and $F_2 = \nabla \cdot d_2(|\nabla v|)\nabla$. The initial values for both nonlinear evolution equations are chosen to be the same image, i.e., $u(\mathbf{r},0) = v(\mathbf{r},0) = X(\mathbf{r})$. As a nonlinear dynamic system, the time evolution of Eqs. (4.37) and (4.38) will eventually lead to a synchronization in the solution for positive nonzero coupling coefficients. For the purpose of image processing, Eqs. (4.37) and (4.38) are designed to evolve at dramatically different time scales, for example, the coefficients d_1 and d_2 are chosen as the Gaussian form in Eq. (4.34) with $d_{20} >> d_{10} \geq 0$. After finite time evolution, the image edges are obtained as the difference [188]

$$w(\mathbf{r},t) = u(\mathbf{r},t) - v(\mathbf{r},t). \tag{4.39}$$

It was shown that Eq. (4.39) behaves like a band-pass filter when $d_{20} >> d_{10} \sim 0$. The essence of this approach is that when two coupled evolution PDEs are evolving at dramatically different speeds, the difference of two low-pass PDE operators gives rise to a band-pass or high-pass filter. The coupling terms play the role of relative fidelity, and balance the disparity of two images. It has been shown that nonlinear PDE based high-pass filters work extremely well for images with a large number of textures, and outperform classical Sobel, Prewitt, and Canny operators [165, 188].

4.3.4 Lagrangian Representations and Surface Extraction

Representing 3D regions using an indicator function as in the above definitions requires intrinsically a large amount of memory storage, which scales as a cubic function of the resolution in each dimension. The difficulty can be alleviated by using adaptive data structures such as the octree. However, the implicit representation can still be inefficient to generate exact sample points and their geodesic neighborhoods on such surfaces. Thus, for geometric processing tasks that involve evaluation of properties that depend on a local neighborhood of a point on the surface, such as curvature, it is far more efficient to first convert the implicit representation to an explicit one, i.e., a Lagrangian representation.

Another advantage of the Lagrangian representation is that the sampling points can move with the surface when it undergoes geometric deformation, or simply a smoothing process. In contrast, implicit indicator functions are sampled on regular grid points fixed in 3D space, i.e., Eulerian. In addition, the Eulerian representations are prone to grid alignment artifacts in geometry processing procedures.

The shape of a nondegenerate smooth 3D object can be defined through its boundary surface. In geometric modeling, such a representation using boundary surfaces is called *boundary representation*, or B-rep for short[193]. The curved 2D surface is often tessellated into a collection of faces (2D cells) connected through common edges (1D cells) or vertices (0D cells). For efficient cell incidence and adjacency queries, there are a number of popular B-rep data structures mostly designed based on the connectivity information of each edge. We will discuss one such structure, the halfedge data structure, in the next subsection. Here, we first introduce the basic concepts and the commonly used face-based triangle mesh data structures, which are also the basic forms of most common standard file formats for the Lagrangian surface representations.

4.3.4.1 Triangle Meshes

Triangle meshes are the *de facto* standard in geometry processing. Mathematically, it can be defined as a specific type of *2D simplicial complex*. A 2D simplicial complex can be defined as a 3-tuple (V,E,F), where $V = \{v_0, v_1, ...\}$ is a set of vertices, $E = \{\{v_i, v_j\}, ...\}$ is a set of edges connecting vertices $\{v_i, v_j\}, ...$, and $F = \{(v_i, v_j, v_k), ...\}$ is a set of (counterclockwise oriented) triangles, each with 3 vertices as its corners. All edges of the triangles in F must also be in E, and all vertices of the edges in E must be in V as well. The simplicial complex provides the connectivity information of the cells. If the triangles incident to each vertex form a disk-like topology, the simplicial complex represents a 2D manifold. Assigning 3D coordinates to each vertex, using the straight line segment linking the pair of vertices to represent each edge, and using the flat triangle formed by the three vertices to represent each face, we can *embed* the simplicial complex in the 3D Euclidean space. It is called a *geometric realization* of such a simplicial complex, assuming no

self-intersection among the triangles. Such a geometric realization is called a triangle mesh.

Given any closed smooth 2-manifold embedded in the 3D Euclidean space (i.e., the boundary surface of a regular 3D object), it can always be approximated by such a triangle mesh, just as a smooth function can always be approximated by piecewise linear functions. A typical file storing such data simply consists of a list of vertex coordinates (3 floating point numbers per vertex) followed by a list of triangles (3 vertex indices per triangle).

4.3.4.2 Marching Cubes

Both the 3D imaging data from cryo-EM and the result of the aforementioned Eulerian geometric flows or PDE-based nonlinear filtering are given as functions sampled on Eulerian grids. For subsequent use in finite element methods (FEMs) or Lagrangian geometry processing, which is often much more efficient than the Eulerian representation due to better adaptivity of the irregular sampling and the reduction from 3D into curved 2D representation. In most geometry processing approaches, *h*-refinement (more segments) is preferred over *p*-refinement (degree of the polynomial in each segment), since the modern computer architectures can handle a large number of simple objects more efficiently than a small number of complex objects representing the same surface [21]. In the following, we use the extremal C^0 case, i.e., piece-wise flat surface meshes, the de-facto standard data structure in current geometry processing.

The conversion from 3D image data to 2D triangle mesh is often implemented by using the widely used marching cubes algorithm [115]. Without loss of generality, we can assume the isosurface to be extracted is the 0 level-set. The vertices can be found on edges with opposite field values on both ends. The exact location of the intersection of the isosurface on the edge can be easily computed based on the trilinear interpolation approximation of the continuous underlying field, which reduces to linear interpolation along the edge. The mesh connectivity is then established by examining each cell and constructing triangles with vertices on the edges of that cell by checking in a predetermined lookup table, which contains the connectivity information of the vertices within each cell for the 256 possible sign configurations of the 8 grid points of the cell. The actual lookup

table is reduced to 15 cases by using symmetry. For some cases, disambiguation based on actual field values is necessary [121].

We recommend the use of marching cubes for applications that do not require well-shaped elements, as this approach is highly parallelizable. On the other hand, for FEM and other applications with stringent requirement on the maximum and minimum angles of each triangle, we propose to use an alternative based on restricted Delaunay triangulation [19]. A sample implementation is available from the computational geometry algorithms library (CGAL) [43]. It distributes sample points on the surface, and then extracts an interpolating triangle mesh through the 3D triangulation of the points. These points are added iteratively following a Delaunay refinement-like step until the sizes and shapes of the mesh triangles meet specified criteria. Other choices include dual contouring for adaptive octree data [97, 16], and the extended marching cubes for models with sharp features [101], which might be rare for cryo-EM datasets though.

4.3.4.3 Dual Contouring

An alternative method called dual contouring [97] extracts an isosurface of the implicit function by first generating surface vertices in the interior of each volume cell (of a regular grid or an adaptive octree), followed by constructing a polygon per edge that intersects the isosurface.

4.3.5 Finite Element Meshing

4.3.5.1 Remeshing

For finite element analysis of molecular surfaces, it is often not enough to have just a triangle mesh, but necessary to also produce one with high quality element shapes. One practical approach is to go through a *remeshing* process on the results of the marching cubes method or its variants, where the geometric locations of the sample points (vertices) can be optimized and/or the topological connectivity is also optimized so that the mesh quality is improved for the target application. This process leads to a semi-regular mesh with most of its vertices neighboring six triangles. Al-

ternatively, meshes with well-shaped triangles can be directly produced through a constrained 3D Delaunay refinement if an implicit surface is given in the form of a level set of a 3D function [19].

4.3.5.2 Volumetric Meshing

Volumetric meshing refers to the interior meshing. It is possible to generate tetrahedron meshes directly from 3D images with theoretical bounds on dihedral angles using algorithms such as isosurface stuffing [105]. Isosurface stuffing uses regular patterns to tetrahedralize grid cells completely inside the surface, followed by a marching-cubes-like boundary treatment, which shifts some of grid points near the boundary for attaining a better element shape. The algorithm is extremely fast, and the surface can approximate smooth iso-surfaces well under reasonable assumptions, but the element shape is not optimal, and its adaptivity is restricted to octree-like structures.

Other available popular algorithms include TetGen [158] and NetGen [148], both providing user control on the size and shape of tetrahedra. The NetGen can take either a constructive solid geometry (shapes composed of primitive shapes combined through Boolean operations, i.e., union, intersection, and subtraction) or a boundary surface representation (BRep). However, NetGen can be less robust than other algorithms, such as TetGen [114]. TetGen produces tetrahedron meshes through constrained Delaunay tetrahedralization. If the tetrahedron mesh is required to conform to a boundary triangle mesh, the TetGen can be the method of choice. However, restricting the boundary to the given mesh can make the quality of the volumetric mesh dependent on the surface triangle mesh given by the user.

Another recent algorithm using interleaved Delaunay refinement and mesh optimization [171] can generate quality meshes that satisfy a set of user-defined criteria, which can be useful, for example, when importance of the sampling density is determined by the local chemical structure. In the Delaunay refinement step, sample points are inserted in order to satisfy the user-specified quality requirements. In the optimization step, a target function called the optimal Delaunay triangulation energy is used, whose minimization leads to a high quality mesh. A final step perturbing the locations of vertices of slivers (flat tetrahedra) further improves the mesh quality.

4.3.5.3 Incidence and Adjacency

Another requirement for performing finite element analysis on meshes is that incidence relations (e.g., face-edge, or edge-vertex) and adjacency information (e.g., face-face, edge-edge, or vertex-vertex) should be performed with constant time complexity. Such incidence/adjacency information is essential in constructing differential operators, solving differential equations, evaluating geometric quantities, or even simply reducing geometric noise. To provide such efficient query capability, a large number of data structures have been proposed, including winged edge, halfedge, and combinatorial maps.

Halfedge data structure is among the most popular ones in computer-aided geometric design and in geometry processing. It is based on the observation that each edge is adjacent to exactly two polygon faces for a manifold surface, so the connectivity information for each edge-face pair (halfedge) can be stored in a fixed length array. Other incidence/adjacency information can then be restored from the connectivity of halfedges in constant time. In the implementation details of halfedge data structure, each edge in the mesh is split into two halfedges with opposite orientations. Each halfedge stores the references to its incident face, incident vertex, and opposite halfedge. Each face and vertex store one reference to one of its incident halfedges. Traversal from each element to another element is achieved through halfedges.

While halfedge data structure is widely used for representing surface meshes, it is not designed for volumetric meshes. Volumetric meshes are defined as the polyhedral representation of the object's inside volume. From the data structure point of view, the main difference between volume mesh and surface mesh is whether it includes 3D cell information. Volumetric meshes can be described by combinatorial maps, which provide a way to describe the volume structure using *darts* (extension of halfedge from edge-face pair to edge-face-cell triple) and maps between darts. There are a number of existing tools to generate volumetric meshes. In this chapter, we compare two of them, which produce good polygon shapes and provide enough information to reconstruct the combinatorial maps to query the adjacency information. In our implementation, we employed a compact data structure designed for combinatorial maps in 3D [62].

4.3.6 Surface Area and Surface Enclosed Volume

Surface area and enclosed volume are crucial components in the mathematical and thermodynamical modeling of biomolecular systems [186, 187, 34, 35].

Surface area evaluation for surfaces in Lagrangian representation is straightforward. One only needs to sum up all the triangle areas. The process is essentially akin to taking the Riemann sum for evaluating the definite integral of a continuous function. Thus, it converges to the actual surface area, provided that the underlying surface is continuous. To be more specifically, given a Lagrangian mesh with piecewise flat segments, one simply sums up the area of each surface triangle,

$$A(S) \;=\; \int_S 1 \, dA \;=\; \sum_{t_l \in T} |t_l| \;=\; \sum_{t_l \in T} \frac{1}{2} |(\mathbf{v}_k - \mathbf{v}_i) \times (\mathbf{v}_j - \mathbf{v}_i)|, \tag{4.40}$$

where S is the surface of the biomolecule, $A(S)$ is the total surface area, and T contains all the surface triangles in a tessellation of S. Here t_l is a triangle mesh element, with $\mathbf{v}_i$, $\mathbf{v}_j$ and $\mathbf{v}_k$ as the coordinates of its three vertices.

To evaluate the volume of the 3D object/region enclosed by a surface, one may take the integral of the flux of one third of the coordinates field across the surface boundary. This can be proved by the divergence theorem (Gauss's law), since the divergence of one third of the coordinates field is 1. Alternative, it can be computed by summing up the signed volumes of all tetrahedra formed by boundary triangles and the origin of the 3D coordinate system. To be more specifically, one picks an arbitrary point inside or outside the mesh, for example, the origin, and sums up all the signed volume of the tetrahedron formed by the point and each triangle,

$$V(S) \;=\; \int_S \mathbf{x} \cdot \mathbf{n} \, dA \;=\; \sum_{t_l \in T} \frac{1}{6} ((\mathbf{v}_k - \mathbf{v}_i) \times (\mathbf{v}_j - \mathbf{v}_i)) \cdot \mathbf{v}_i, \tag{4.41}$$

where $V(S)$ is the total volume, $\mathbf{n}$ is the outward surface normal at a position $\mathbf{x}$ on the surface S. Here the vertices of each triangle are assumed to be listed in counterclockwise order when viewed from the outside of the surface. Even when a volumetric mesh of the inside is available, summing up the volumes of these thin tetrahedra formed by a fixed point and boundary faces is in general much more efficient than summing up the volumes all the tetrahedra of the volumetric mesh.

The accuracy of the above surface area and the volume estimates depends on the extracted mesh quality. If one computes these values on a coarse mesh, one will end up with results with a large deviation from the true value of the underlying smooth surface. However, as the discretization on the original surface becomes finer, the values of the computed area and volume will become closer to the real values of the objects that the meshes represent.

4.3.7 Electrostatic Analysis on Surface Meshes

To compute the areas of different regions defined by certain properties, such as electrostatics, associated to surface points, we can get a rough and quick estimate by classifying entire triangles into such regions, and sum up the triangle areas in each region. For example, to compute the area of the regions with positive polarity, we can classify each triangle with at least two positive vertices into such regions. For our specific analysis of protein data models, we can classify the surface of the protein model as positive charge regions, negative charge regions and neutral regions. Different types of regions of the surface with different charge densities could be used to analyze the chemical and physical properties of the surface of the protein.

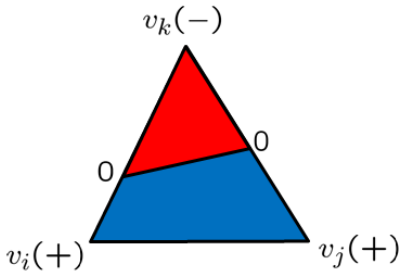

Figure 4.1: Fractional area. Red is the fractional area for the negative portion; blue is for the positive portion.

For improved accuracy, we can compute the fractional area within a triangle, assuming linear interpolation of the indicator function stored at each vertex. For instance in Figure 4.1, given the triangle with vertices v_i, v_j, and v_k, if both v_i, v_j are with positive charge density, and v_k is with negative charge density, we can compute the proportion of the negative parts of edge $v_i v_k$ and edge

$v_j v_k$. If the proportions are s and t, respectively, the area of the negative part within the triangle would be stA (in red), where A is the total area of the triangle. The rest part would be the area for the positive part (in blue). However, once our mesh refines, the computed area difference between the results produced by the above two methods will diminish.

4.4 Results and Discussion

In this section, we test the introduced modeling methods on two datasets; cryo-EM maps datasets and PDB datasets. We also investigate and discuss the pros and cons of those methods.

4.4.1 Cryo-EM Maps Datasets

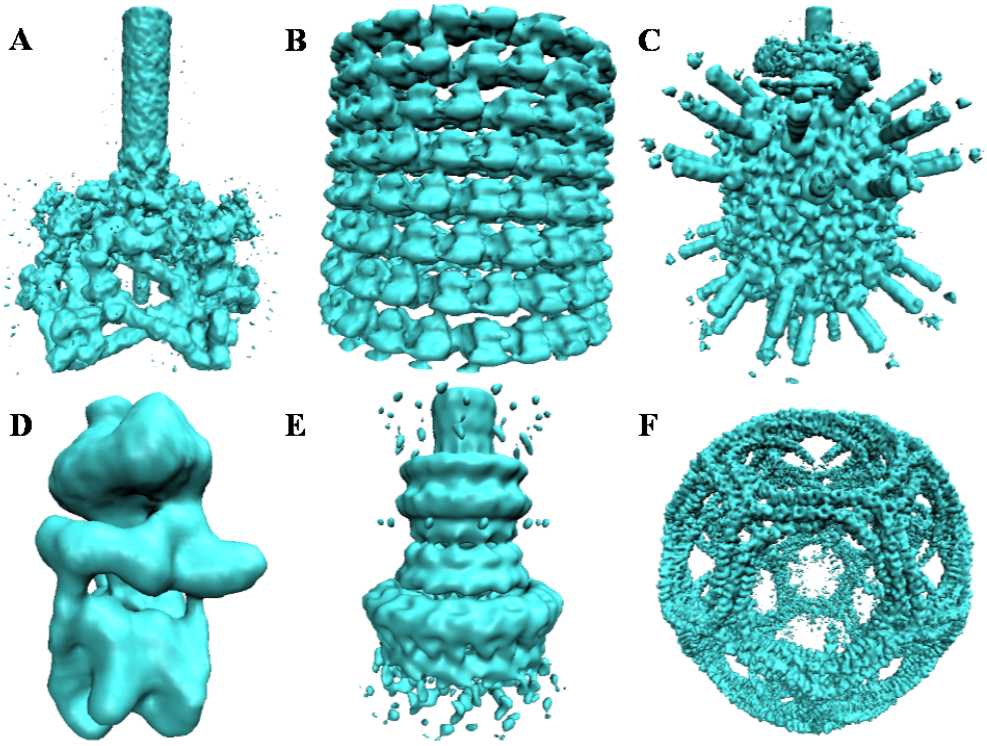

Figure 4.2: Image gallery of representative cryo-EM maps used in this study. The VMD is used for visualization.

In the present section, we consider six representative cryo-EM maps from the EMDataBank. With the help of visualization tool VMD (http://www.ks.uiuc.edu/Research/vmd/), we extract their

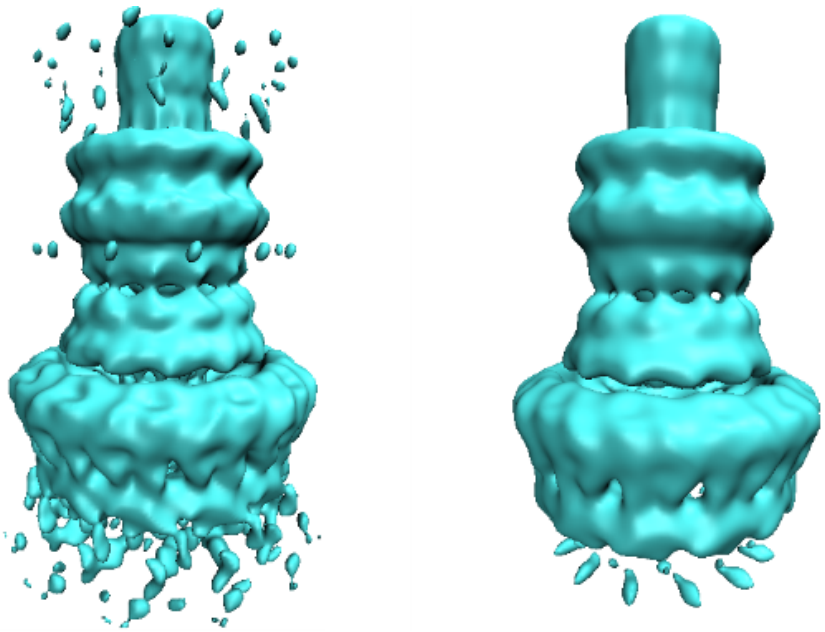

Figure 4.3: Noise reduction of EMD1617. Left: Before filtering; Right: After filtering by high order geometric PDEs.

surfaces with the recommended iso-values and the results are displayed in Fig. 4.2. The details of these data are summarized as follows.

- Fig. 4.2A(EMD1048): The baseplate of bacteriophage T4. It is a multiprotein molecular machine that controls host cell recognition, attachment, tail sheath contraction and viral DNA ejection.

- Fig. 4.2B(EMD1129): GDP-tubulin. It is a GDP-bound tubulin. The rope-like polymers of tubulin, which are components of the cytoskeleton, can grow as long as 25 micrometers and are highly dynamic.

- Fig. 4.2C(EMD1265): Bacteriophage $\phi 29$. It is a viral DNA-packaging motor, which translocates and compresses genomic DNA with tremendous velocity into a preformed protein shell (the procapsid).

- Fig. 4.2D(EMD1590): Manduca sexta vacuolar ATPase complex. It is a V-ATPase, which acidifies a wide array of intracellular organelles and pumps protons across the plasma mem-

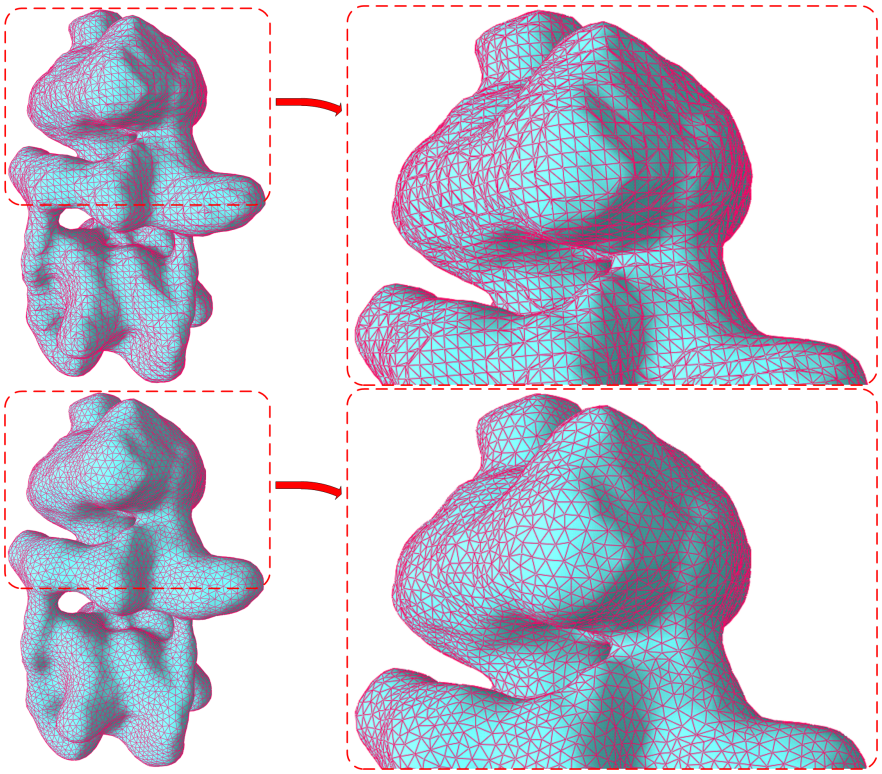

Figure 4.4: Comparison of surface meshes. Top: Marching cubes result of EMD1590; Bottom: CGAL result of EMD1590.

branes. V-ATPases couple the energy of ATP hydrolysis to proton transport across intracellular and plasma membranes of eukaryotic cells.

- Fig. 4.2E(EMD1617): Shigella flexneri T3SS needle complex. The type three secretion system (T3SS) is a protein appendage found in several Gram-negative bacteria. In pathogenic bacteria, the needle-like structure is used as a sensory probe to detect the presence of eukaryotic organisms and secrete proteins that help the bacteria infect them.

- Fig. 4.2F(EMD5119): Clathrin coats. It is a polyhedral lattice that surrounds the vesicle in order to safely transport molecules between cells. The endocytosis and exocytosis of vesicles allow cells to transfer nutrients, to import signaling receptors, and to mediate an immune response.

4.4.1.1 Data Denoising and Surface Extraction

In this part, we explore our integrated tools on EMD data sets to test the strategies proposed used for geometric modeling. Six different EMD objects shown in Fig. 4.2 are employed for the present study. It can be seen from Fig. 4.2 that the electron tomography sometimes produces extremely noisy and low contrast 3D density maps. The poor signal-to-noise ratio (SNR) hinders visualization and interpretation. Therefore, some noise filtering techniques are indispensable. Many important methods and schemes, like wavelet transform techniques, nonlinear anisotropic diffusions, Beltrami flow, bilateral filter, and iterative median filtering have been used for noise reduction [164, 65, 66, 170, 95, 132, 173]. In this chapter, to improve the noise removal, we make use of the high order geometric flows. Basically, it is a set of high order geometric PDE based low-pass filters for image processing or surface analysis. An example of noise removal of EMD1617 is demonstrated in Fig. 4.3. The basic structure of the T3SS needle complex is preserved while the noise amplitude is dramatically reduced. During the process of noise reduction, the surface of the protein is smoothed. Artificial sharp edge and sharp tips are naturally removed. From the energy minimization point of view, these features are not favorable in the biological surface formation [10]. Therefore, the loss of these features does not lead to degradation in accuracy when dealing with biological data.

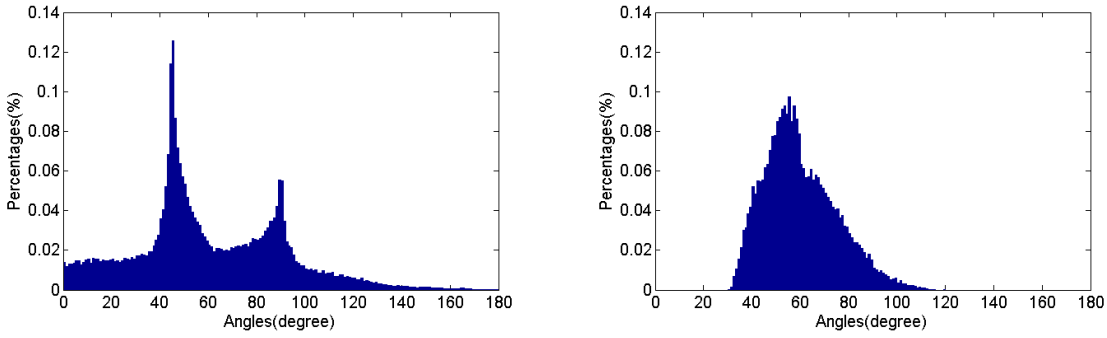

Figure 4.5: Comparison of surface mesh angle distributions. Left: Angle histogram of marching cubes result of EMD1590; Right: Angle histogram of CGAL isosurface extraction result of EMD1590.

Among those surface extraction methods we showed in Section 4.3.4, we compare the results

of the marching cubes method and CGAL's Delaunay-based method in Figure 4.4. The marching cubes method is highly parallelizable because the lookup table used is precomputed and stored. For all the files that we tested, it costs less than five seconds to process a cryo-EM map with up to $200\times200\times200$ Cartesian grids. The direct result from marching cubes for EMD1590 shows that it may have a large number of skinny triangles, and the overall shape may contain terracing artifacts for a large proportion of the triangles. Many triangles have sharp angles less than 30°. The lack of element quality control is the intrinsic weakness of the marching cubes methods. Thus, the result often needs post-processing to improve the mesh quality. It may also require a large number of triangles to store (14,170 vertices and 28,360 triangles in the example shown) at a given accuracy due to the lack of adaptivity.

On the other hand, CGAL's running time and generated surface mesh quality depend heavily on the criteria chosen for the Delaunay triangulation. The criteria are controlled by three parameters: angular bound for mesh triangles' minimal angle, radius bound for the maximum surface Delaunay balls' radius (a surface Delaunay ball circumscribes a mesh triangle and centered on surface), and distance bound for the maximum distance between triangle's circumcenter and surface Delaunay ball center. However, these parameters can usually be easily tuned to achieve proper results with appropriate triangle shapes and sizes given some domain knowledge of the intended application. We use 30° as the angular bound, 0.8 as the radius bound and 0.8 as the distance bound to directly extract the EMD1590 surface mesh from its cryo-EM map file. It takes about four seconds to run the algorithm to get the extracted surface with 10,100 vertices and 20,220 triangles. If we set smaller parameter values we will get more detailed models but may suffer from longer running time and increased mesh size due to the smaller mesh triangles. Compared to the marching cubes result for EMD1590, the CGAL result gives triangle angles always greater than 30 degrees, triangles with almost the same size, and reduced mesh sizes without losing surface shape accuracy, as shown in Fig. 4.4.

The histograms of the triangle angle for both meshes are given in Fig. 4.5. From the figure one can see that the angles in the marching cubes results have a large distribution in the low

range, which may result in accuracy problems in mathematical modeling involving PDEs. In contrast, CGAL's results are guaranteed to meet the angle requirements while reducing the mesh size significantly.

4.4.1.2 Surface Mesh Improvement

The surface generated from the marching cubes method or its variants often does not fit the need of applications relying on finite element, finite difference, or finite volume methods, such as geometric reconstruction of the internal structure or numerical simulation of electrostatics [200, 201, 186, 34, 35]. The process of creating a mesh that satisfies the new requirements while remaining close to the original mesh is called remeshing [2]. For instance, the mesh for EMD1590 produced by the marching cubes method can be remeshed into a mesh with rather uniform well-shaped triangles as shown in Fig. 4.6. Even for meshes with well-shaped elements (for triangle meshes, this means nearly equilateral triangles, measured by the ratio of the circumcircle radius to the length of the shortest edge [156]), it is still possible to perform remeshing to reduce vertex count while remaining faithful to the original underlying surface. In this case, the procedure is also called mesh simplification.

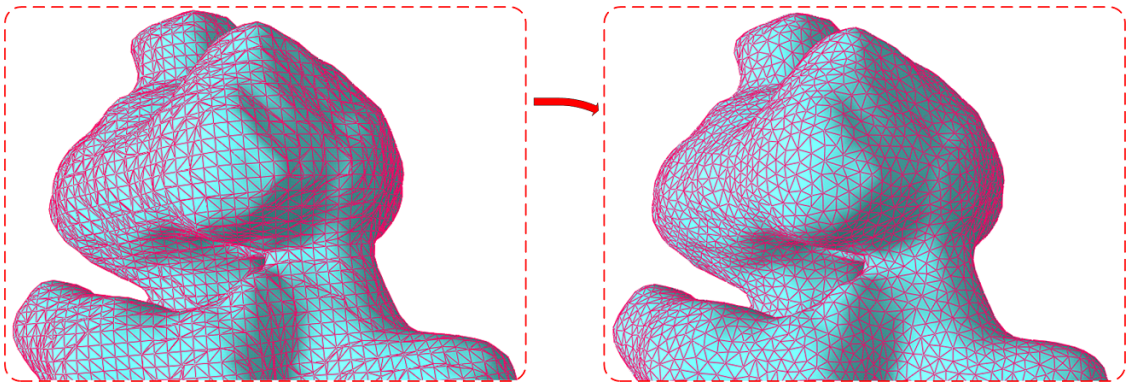

Figure 4.6: Mesh improvement with Delaunay remeshing from left (marching cubes result of EMD1590) to right.

Figure 4.7 presents a collection of meshes of six cryo-EM maps generated by using the CGAL approaches. The bacteriophage $\phi 29$ and clathrin lattice have small scale features. In particular,

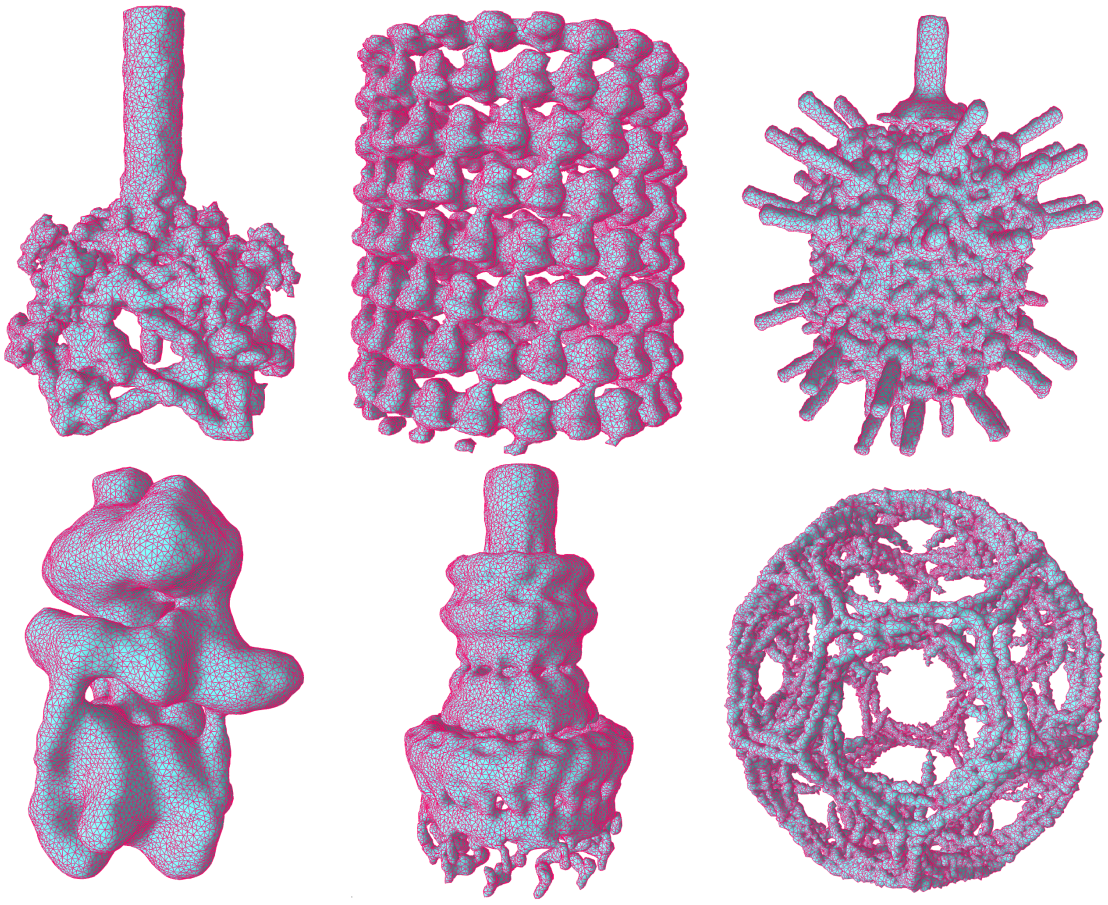

Figure 4.7: CGAL results of surface meshes. From upper left to lower right: EMD1048 [103]; EMD1129 [178]; EMD1265 [196]; EMD1590 [122]; EMD1617 [92]; EMD5119 [70].

clathrin lattice data are quite noisy. It is seen from Fig. 4.7 that the CGAL library is very robust and reliable for cryo-EM meshing.

4.4.1.3 Areas, Volumes and Curvatures

Surface areas and enclosed volumes are frequently used in mathematical models of biomolecular systems [186, 187, 34, 35]. Accurate estimation of surface areas and enclosed volumes is important in theoretical biology. The validation of the presented numerical methods is described below.

We compute surface areas and enclosed volumes for spheres with different radii by the proposed methods and give the comparison between the theoretical values and their estimates in Table 4.1. The radii used in our tests are 1, $\sqrt{2}$ and $\sqrt{3}$, whose theoretical values of area and volume

are straightforward to compute. It can be seen that the straightforward methods proposed in this chapter are accurate for the high quality meshes generated using the proposed methods.

Table 4.1: Comparison of theoretical values and computed estimate of sphere's areas and volumes.

radius	total area (est.)	total area (theo.)	total volume (est.)	total volume (theo.)
1	12.53	12.57	4.16	4.19
$\sqrt{2}$	25.09	25.13	11.81	11.85
$\sqrt{3}$	37.66	37.70	21.72	21.77

Table 4.2: The curvatures estimated using barycentric dual cell area. Here μ_K (resp., μ_H) is the average of Gaussian curvature K (mean curvature H), σ_K^2 (σ_H^2) is the standard deviation of K (H), and K_{theo} (H_{theo}) is the theoretical value of K (H).

radius	μ_K	σ_K^2	K_{theo}	μ_H	σ_H^2	H_{theo}
1	1.003239	0.026352	1	1.000031	0.018632	1
$\sqrt{2}$	0.500804	0.011344	0.5	0.707106	0.011323	0.707107
$\sqrt{3}$	0.333685	0.009292	0.333333	0.577343	0.010881	0.577350

Table 4.3: The curvatures estimated using Voronoi cell area. Here μ_K (resp., μ_H) is the average of Gaussian curvature K (mean curvature H), σ_K^2 (σ_H^2) is the standard deviation of K (H), and K_{theo} (H_{theo}) is the theoretical value of K (H).

radius	μ_K	σ_K^2	K_{theo}	μ_H	σ_H^2	H_{theo}
1	1.003238	0.010534	1	1.000030	0.002627	1
$\sqrt{2}$	0.500804	0.007382	0.5	0.707107	0.003701	0.707107
$\sqrt{3}$	0.333684	0.007158	0.333333	0.577343	0.005481	0.577350

As shown in Section 2.1.2, the curvature at a point on the surface describes the local geometric feature. Curvature analysis is useful for the identification of protein-protein and protein-ligand interaction sites. It can also be used to help understand the protein-DNA binding specificity. In this chapter, we first validate the accuracy and convergence order of the numerical methods proposed in Section 2.1.2. We then demonstrate the usefulness of these methods for cryo-EM data analysis.

As discussed in Section 2.1.2, around each vertex, the one-ring area can be chosen in two different ways, the barycentric dual cell area and the Voronoi cell area. The accuracy of these approaches are examined by spheres of radii (r) 1, $\sqrt{2}$ and $\sqrt{3}$. Their Gaussian and mean curvatures are given by $1/r^2$ and $1/r$, respectively. These spheres are tessellated with triangles of similar

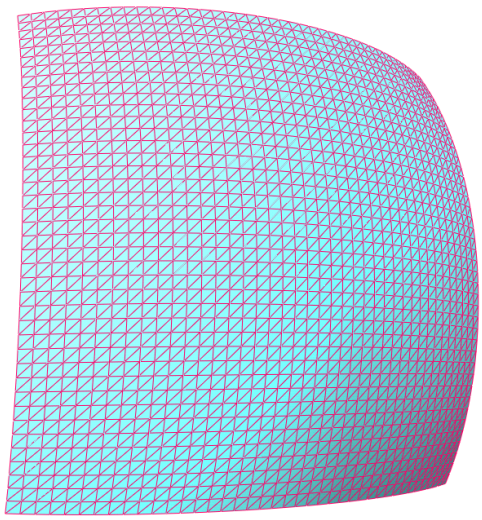

Figure 4.8: The analytical geometric model: a patch of a sphere.

sizes. The results of estimated curvatures obtained with the barycentric dual areas are shown in Table 4.2, together with theoretical values. Both our results for both Gaussian and mean curvatures are accurate in the tests. The resulting standard deviations show that the difference between the computed value and the theoretical value is small relative to the mesh size. For a comparison, we also listed our results obtained with the Voronoi dual areas in Table 4.3, under the same mesh. It is seen that the curvatures computed with the Voronoi dual areas are essentially the same as those with the barycentric dual areas. However, the Voronoi dual area approach offers smaller standard deviations in Gaussian and mean curvature estimations than does the barycentric dual area approach. Therefore, the Voronoi dual area approach performs better and is utilized in the rest of this chapter.

Table 4.4: The convergence orders for Gaussian curvatures on a patch of a sphere.

Maximal edge length	L_∞	Order	L_2	Order
1.00×10^{-1}	8.325×10^{-3}		4.815×10^{-3}	
5.00×10^{-2}	2.676×10^{-3}	1.64	1.149×10^{-3}	2.07
2.50×10^{-2}	1.031×10^{-3}	1.38	2.803×10^{-4}	2.04
1.25×10^{-2}	4.544×10^{-4}	1.18	6.920×10^{-5}	2.02

Table 4.5: The convergence orders for Mean curvatures on a patch of a sphere

Maximal edge length	L_∞	Order	L_2	Order
1.00×10^{-1}	1.275×10^{-3}		5.228×10^{-4}	
5.00×10^{-2}	3.441×10^{-4}	1.89	1.409×10^{-4}	1.89
2.50×10^{-2}	8.820×10^{-5}	1.96	3.623×10^{-5}	1.96
1.25×10^{-2}	2.226×10^{-5}	1.99	9.167×10^{-6}	1.98

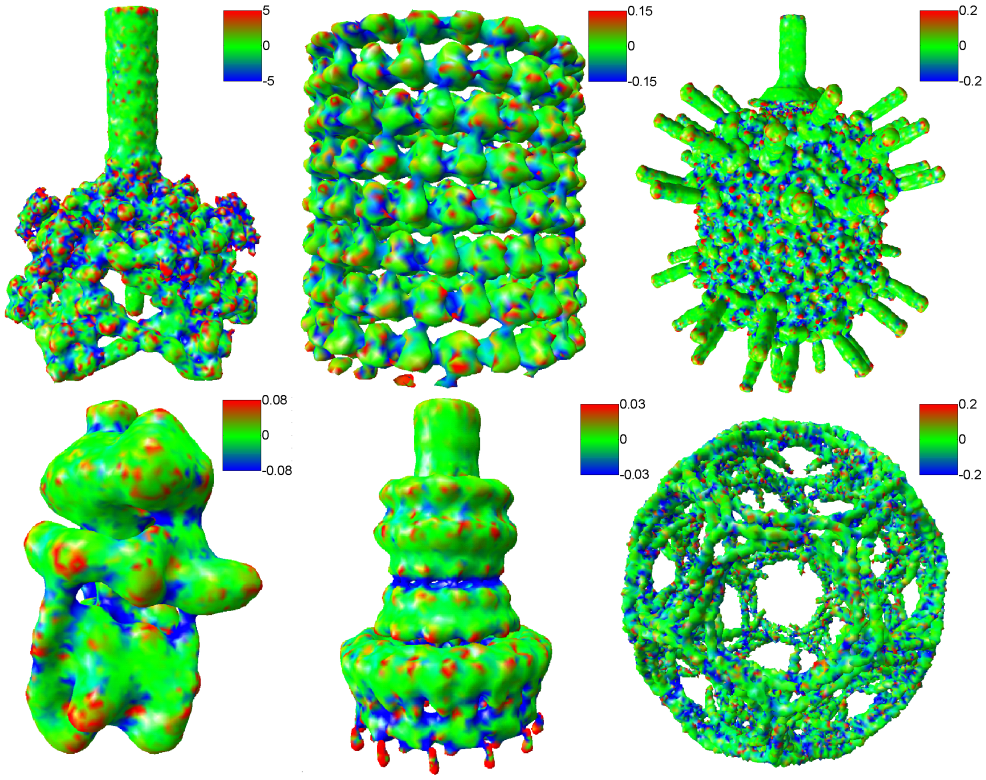

Figure 4.9: Gaussian curvature estimates for six cryo-EM map entries.

To further explore the accuracy and convergence of our curvature estimate, we design tests on different analytical models with different geometric types, including all cases of the intrinsically non-flat ones, namely, peak, pit, saddle ridge, minimal surface, and saddle valley. The pit type is identical to the peak case due to the symmetry. Specifically, the test results on a patch of a sphere are shown here. The convergence orders of our curvature method are measured by L_∞ and L_2 error norms. Tables 4.4 and 4.5 show the orders for Gaussian curvature and mean curvature, respectively. The average L_∞ order is about 1.4, while the second accuracy is achieved for the L_2

order. These results indicate the robustness and reliability of the proposed methods for curvature evaluation.

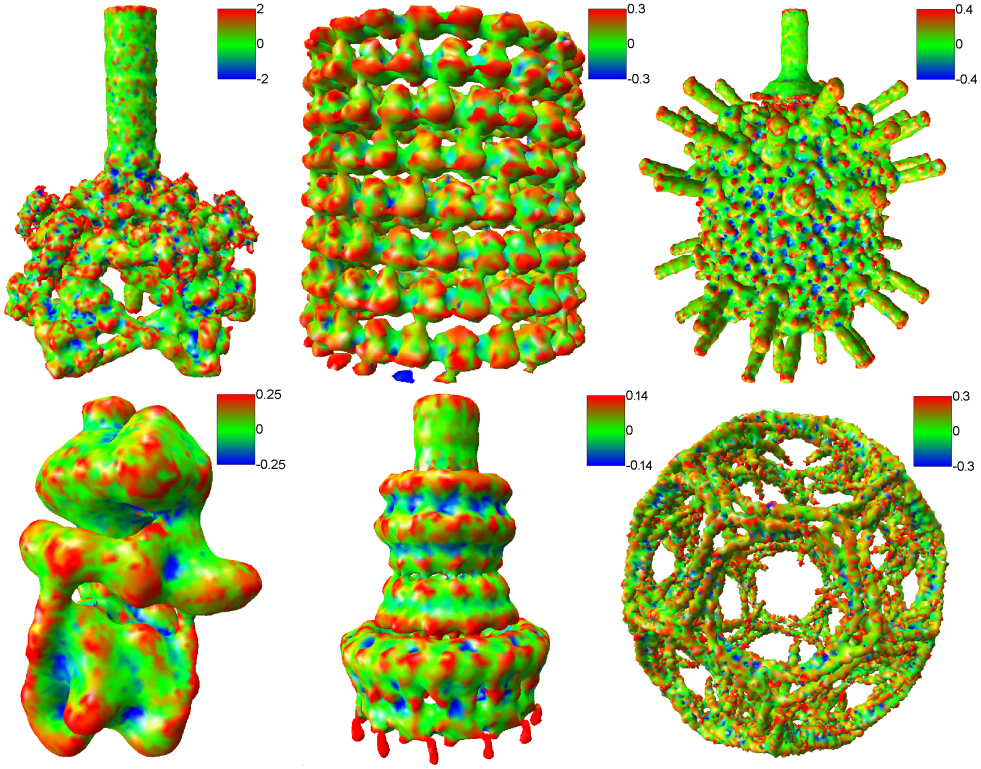

Figure 4.10: Mean curvature estimates for six cryo-EM map entries.

4.4.1.4 Applications of Curvature Estimates to Cryo-EM Maps

Having established the accuracy and convergence of proposed numerical methods for curvature estimation, we apply these methods for the curvature calculation of six cryo-EM map entries. Note that these complexes vary in dimensions. The absolute value of curvatures increases as the dimension decrease as shown in the analytically expressions given in the last section.

First, we evaluate Gaussian curvatures and illustrate the results in Fig. 4.9. Since the Gaussian curvature is an intrinsic measure of curvature and does not depend on the surface embedding, it is a convenient tool for identifying peak, pit, saddle ridge and saddle valley. These features are clearly demonstrated in Fig. 4.9. Taking the Shigella exneri T3SS needle complex as an example,

Gaussian curvatures are mostly negative along the ring regions, which can be identified as saddle valleys, while Gaussian curvatures are positive on peaks and noisy dots.

We next consider the mean curvatures of six cryo-EM map entries. In contrast to the Gaussian curvature, mean curvature is an extrinsic measure of curvature and it reflect the local characteristic of a surface. Figure 4.10 plots the mean curvature maps of six biomolecular complexes. Overall, mean curvatures are mostly positive for these complexes, indicating the main geometric features of peaks, ridges and noisy dots. However, regions with very negative mean curvature can be found for pits and valleys, which are clearly potential binding targets of other smaller compounds.

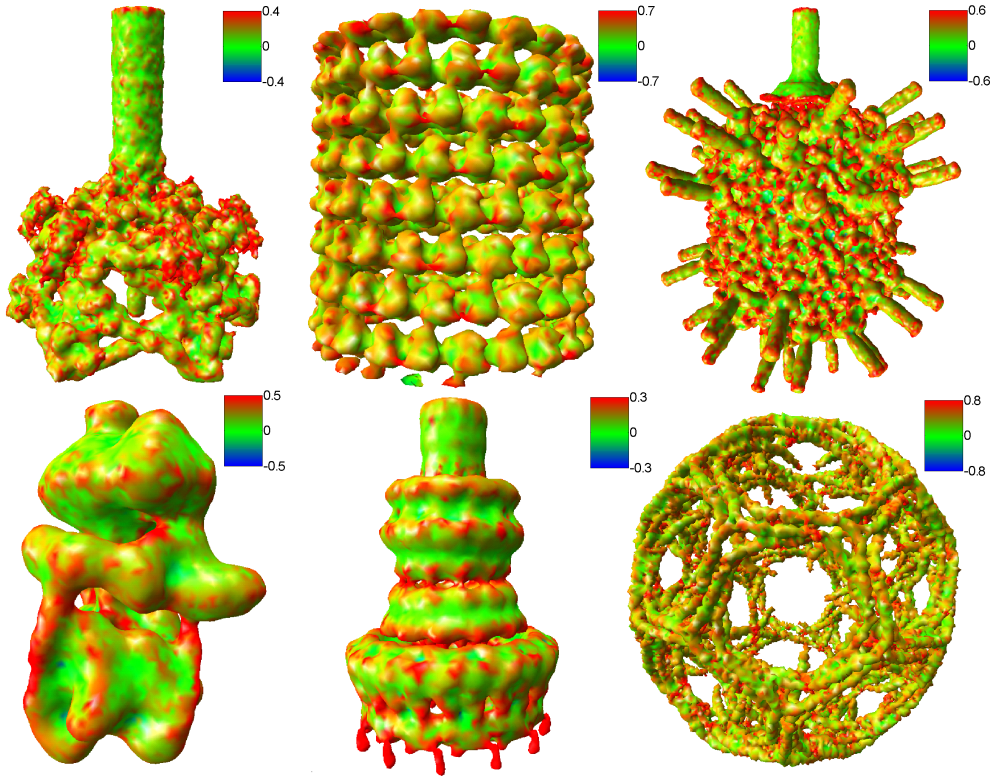

Figure 4.11: Maximum curvature (κ_1) estimates for six cryo-EM map entries.

To further utilize the power of the present curvature estimates, we investigate the behavior of the first and second principal curvatures. The accuracy and convergence of the present curvature estimates established in the last section enable us to accurately compute principal curvatures as well by Eqs. (2.6) and (2.7). The maximum curvatures, κ_1, are plotted in Fig. 4.11. It is interesting to note that the maximum curvature is a very good indicator for peaks and ridges of the biomolec-

ular complex, and possible noisy dots. Therefore, with a good confidence, one can exclude these regions with very large positive κ_1 values from being targets of small binding compounds.

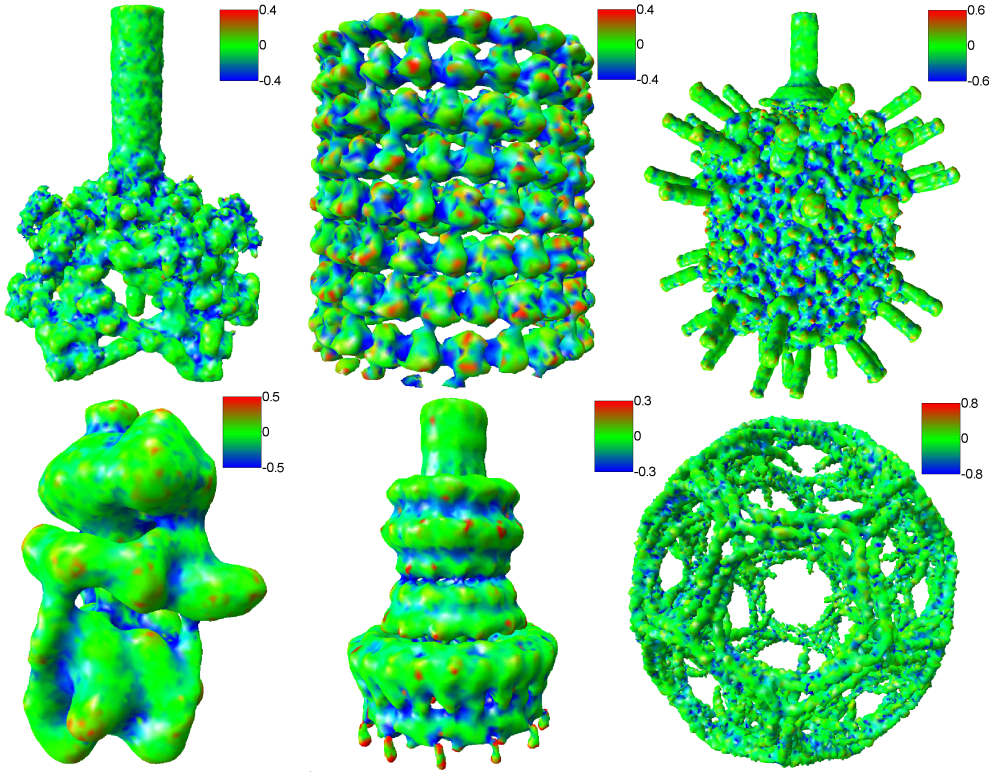

Figure 4.12: Minimum curvature (κ_2) estimates for six cryo-EM map entries.

Finally, we investigate the behavior of the minimum curvature, κ_2. Results are depicted in Fig. 4.12 for six cryo-EM entries. As expected, large negative curvatures indicate pits and valleys (pockets), which are potential binding sites of small compounds. We believe that the second principal curvature can be used as a promising binding indicator for practical docking, drug design and protein design analysis. This aspect, together with the electrostatic analysis, is further analyzed elsewhere for proteins.

4.4.1.5 Volumetric Meshing

If a Lagrangian surface representation readily exists, its volumetric meshing is a separate task. There are a number of strategies for volumetric meshing. First, we can tetrahedralize the surface mesh files by using CGAL library functions. To reduce the work load of the tetrahedralization

Figure 4.13: Comparison of volumetric meshing for an EMD1590 cut-open in the middle. Left: TetGen result; Right: CGAL result.

process, we use CGAL to extract the surface mesh with nearly same sized triangles. Then we use CGAL's tetrahedralization functionality to produce the tetrahedron mesh. The CGAL API function has five parameters to fine-tune the tetrahedralization process: angular bound for surface mesh triangles' minimal angle, triangle size bound for the maximum surface Delaunay ball radius, triangle distance bound for the maximum distance between a triangle's circumcenter and the surface Delaunay ball center, cell radius edge ratio bound for the maximum ratio of the circumradius of a cell to its shortest edge, and cell size bound for the maximum cell circumradius. Setting smaller values for the latter three parameters will lead to more sampling points in the tetrahedralization step, which will increase the number of vertices and tetrahedra in meshes. If the user needs smaller cells near the surface mesh and larger cells far from the surface mesh, a large cell size and small surface triangle size bound could be adopted. The CGAL library tetrahedralization process has

provable guarantees on the surface mesh quality, through tetrahedralization of the interior regions with constrained Delaunay triangulation.

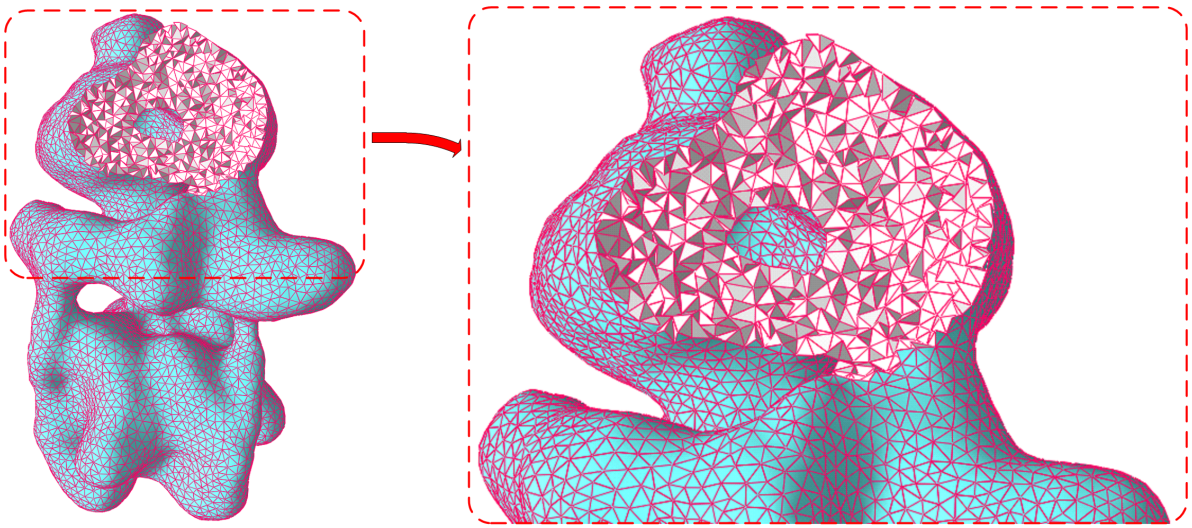

Figure 4.14: Cross-section view of CGAL result of EMD1590.

TetGen is another popular choice for tetrahedralization step with high performance. The TetGen library has a large number of parameters to easily meet various requirements by the users. The most commonly modified parameters for tetrahedralizing a surface mesh is the maximum volume constraint on tetrahedron and the cell radius edge ratio bound. The default value of the cell radius edge ratio bound is 2.0, which can be lowered by the user to remove most cases of low quality element shapes. A comparison of the results from TetGen and CGAL is given in Fig. 4.13. As the surface triangle meshes are of similar high quality, both produced desirable results.

To observe the quality of the tetrahedron meshes generated by the proposed methods, we can show the planar cross-sectional views of the tetrahedron meshes as in Fig. 4.13. Alternatively, we can also observe the internal structure by generating a cross section by removing a connected piece of volume as shown in Fig. 4.14. For this purpose, we first choose a surface face as a seed face, then use breadth-first search algorithm to find a number of tetrahedra connected to the seed face. If we set a constant number of tetrahedra to remove as the stop criterion, we can expose the internal elements in a curved cross-section at approximately the same distance from the seed face. This gives us a cutaway view to illustrate the interior meshing quality after tetrahedralization.

Both CGAL and TetGen share a parameter to tune the cell radius edge ratio bound of a generated tetrahedron. This parameter is highly effective in controlling the quality of the tetrahedra. All well shaped tetrahedra have small values (less than 3) for the ratio, and most of the badly shaped tetrahedra have large values. This does not mean that the limit can be set arbitrarily small, because the value has a lower bound of 0.612 (the value for an equilateral tetrahedron). The one case of a badly shaped tetrahedron with small (cell radius to shortest edge) ratio is called “sliver”, which has a flat and near-degenerate shape. Its cell radius edge ratio can go as low as 0.707. One effective way to prevent slivers from being created is to incorporate a minimum volume constraint, or to employ a procedure called sliver exudation as is done in CGAL.

4.4.2 PDB Datasets

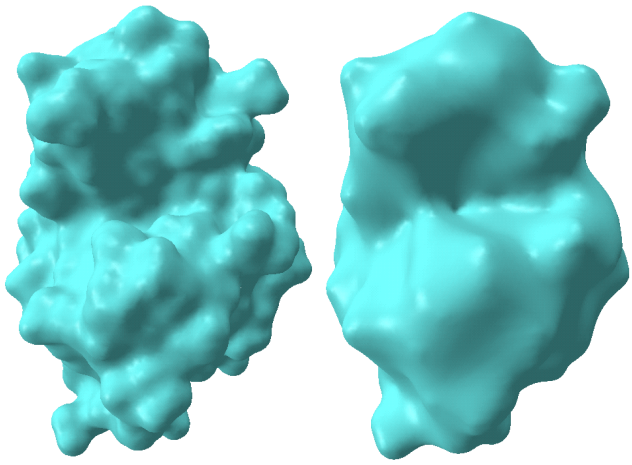

Figure 4.15: Multiresolution surfaces for protein 1HEW. The left chart is a protein surface with finer atomic details. The right chart is a “coarser" surface.

In this section, we demonstrate geometric modeling of biomolecules for PDB datasets. Surface meshes and volumetric meshes are constructed for these biomolecules. With this structural information, the geometric features, such as Gaussian curvature, mean curvature, minimum and maximum curvatures, and shape index are evaluated. The electrostatic potential distribution is also obtained from our models. The combination of electrostatics, curvature and multiresolution offers

a powerful tool for analyzing protein-protein interaction and protein-ligand binding. We also use the toonshading technique for the visualization and analysis. Six proteins from the PDB, namely 1HEW, 1ADS, 1BYH, 1EJN, 2WEB and 1MAG data, are used in our numerical experiments.

4.4.2.1 Multiscale Multiresolution Surfaces

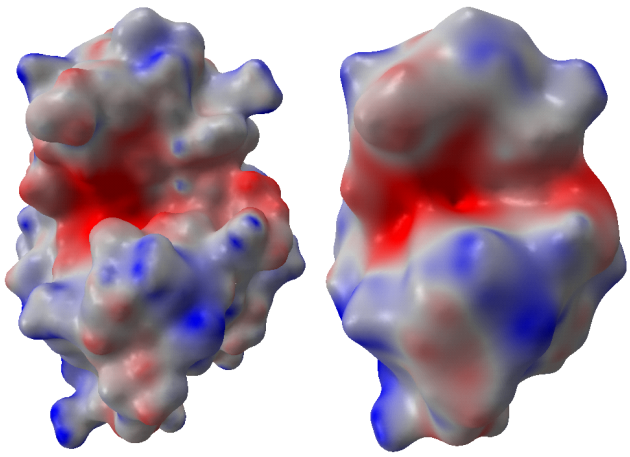

Figure 4.16: The electrostatic distributions on multiresolution surfaces for the protein 1HEW. The left chart is on a protein surface with finer atomic details.

In multiscale multiresolution model, we adjust the initial conditions by choosing different η. In our test, we choose η as 1.3 and 2.0 to deliver protein surfaces of different resolutions. With the small parameter, a surface with much atomic detail is generated. In contrast, when $\eta = 2.0$, the surface is much "thicker" with less atomic detail but with more salient global features. Note that the fine resolution surface can also be generated with a longer integration time, while the coarse resolution surface can be extracted at an earlier time of integration.

Different applications of biomolecular surfaces necessitate multiple resolutions of representation. For example, in ion channels, the radius of the pore is relatively small within the scale of few angstroms. The structure at atomic level contributes to the selectivity of the ion channel. A surface with more atomic detail is preferred. On the other hand, for protein-ligand binding and protein-protein interaction, it is not the detailed atomic shape that matters. Instead, properties like concave or convex regions are more important. Especially, in drug design, the drug molecule binds

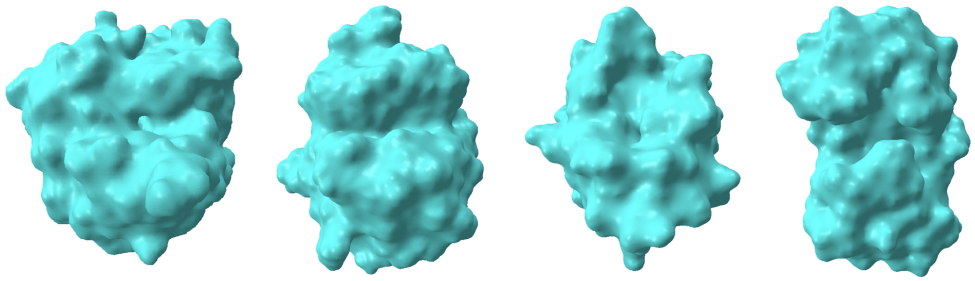

Figure 4.17: Mesh generation results. Left to right:1ADS, 1BYH, 1EJN, 2WEB.

to the protein just as a key to its lock. Detection and analyses of the concave surface area of a protein provides a way to screen the potential candidate drugs.

Except for the surface generation, the coupled system of LB-PB or LB-PNP also delivers information of electrostatic potential distribution. The results are rendered on the surfaces of proteins as shown in Figure 4.16.

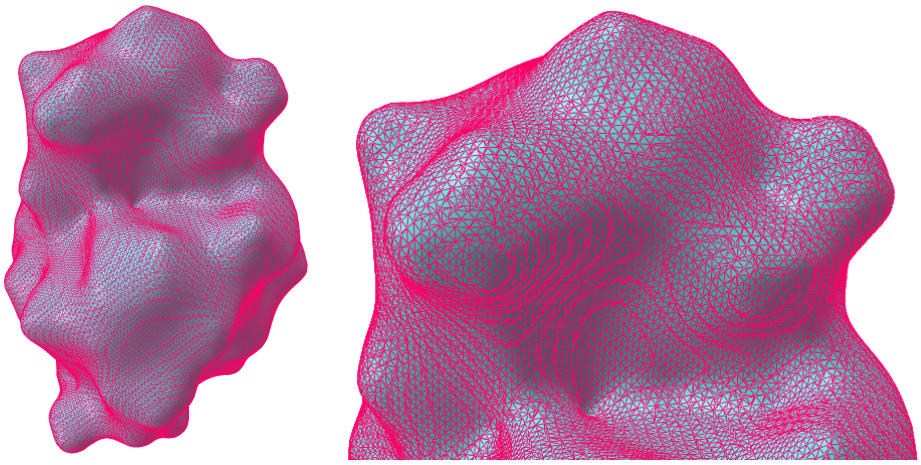

Figure 4.18: The mesh generated from the marching cubes method for protein 1HEW. The left is the entire surface mesh structure. The right is a close-up for the upper region showing the detailed mesh structure.

4.4.2.2 Surface Mesh Generation

In our multiscale multiresolution model, the structural information of a protein is stored in volumetric data, and the surface of a protein can be extracted with a certain isovalue. Basically, we use two methods for surface generation from the volumetric data. One is the marching cubes method. The other is the Delaunay-refinement-based method.

In the marching cubes method, we visit each cell once to extract the connectivity information of triangle meshes within the cell. A pre-computed lookup table is used and the algorithm is of linear complexity in terms of the grid size. In our tests, even for Cartesian grid with dimensions up to 200*200*200, it takes only up to a few seconds to generate the surface mesh on a regular PC. However, the marching cubes algorithm generally suffers from an excessive number of skinny triangles, which cannot be avoided due to the lack of element quality control. Many triangles have acute angles less than 30°. The overall shapes contain terracing artifacts, which are unnecessary for the preservation of the object shape. Figure 4.18 illustrates the mesh for protein 1HEW generated by the marching cubes algorithm.

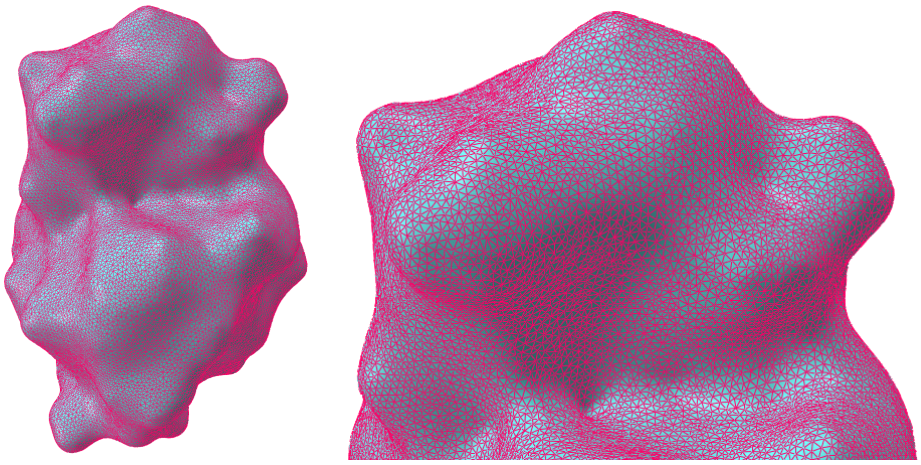

Figure 4.19: The mesh generated from Delaunay-based algorithm for protein 1HEW. The left chart is surface mesh structure. The right chart is a closed-up for the top part.

The Delaunay-based algorithm is available from the Computational Geometry Algorithms Library (CGAL) [43]. This method provides adjustable Delaunay triangulation parameters for an-

gular bound, radius bound and distance bound. Angular bound is for the minimum angle of mesh triangles. Radius bound is for the radius of the maximum surface Delaunay ball, which circumscribes a mesh triangle and is centered on the surface. Distance bound is for the maximum distance between the circumcenter of a surface triangle and the center of the surface Delaunay ball. The mesh quality and the computational time are directly associated with these parameters. In our tests, we set the angular bound to 30, the radius bound to 0.8 and the distance bound to 0.8 to extract the surface mesh. It also takes a few seconds to extract the surface with relatively good mesh qualities. An example is given in Figure 4.19.

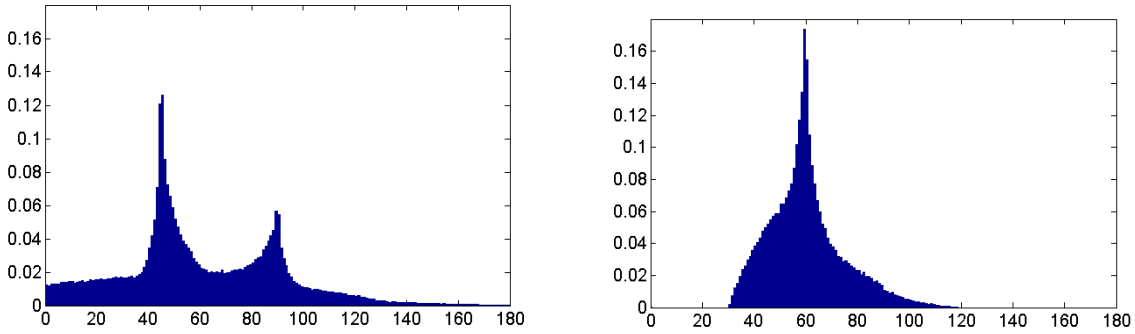

Figure 4.20: Comparison of the mesh quality for marching cubes method and Delaunay-based algorithm: The horizontal axis represents angle degree; The vertical axis represents ratio percentage of the vertices number. The left chart is the angle distribution from marching cubes method. The right chart is the angle distribution from Delaunay-based algorithm.

To quantitatively compare the performance of the above two methods, the angle distribution of the generated mesh is considered. Figure 4.20 presents the angle histogram calculated from the two meshes in Figs. 4.18 and 4.19. It can been seen that the marching cubes method produces many sharp angles, while Delaunay-based algorithm delivers a surface mesh with guaranteed lower bound of 30° for angles. We also count numbers of vertices and triangles for the two meshes. In Figure 4.18, 45,208 vertices and 90,412 triangles are used. In contrast, the Delaunay-based method result has only 32,755 vertices and 65,506 triangles at a similar accuracy.

In the CGAL library [43], remeshing algorithm is also available for improving the mesh quality. Figure 4.21 demonstrates the remeshed surface triangles based on the marching cube results.

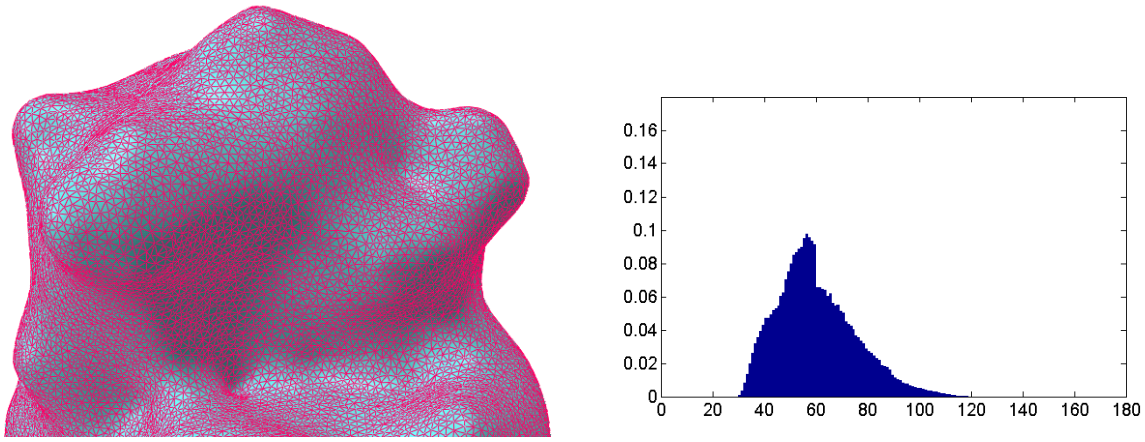

Figure 4.21: Remeshing results for protein 1HEW based on the structure from marching cubes method. The left chart is the surface structure after remeshing algorithm. The right is the angle distribution. The horizontal axis represents angle degree, and the vertical axis represents ratio percentage.

From the angle distribution, it is seen that the mesh quality is improved. The numbers for vertices and triangles are reduced to 31,603 and 63,202 respectively. This kind of high quality mesh is necessitated to guarantee the computational accuracy if the finite element methods are to be applied.

4.4.2.3 Volumetric Meshing

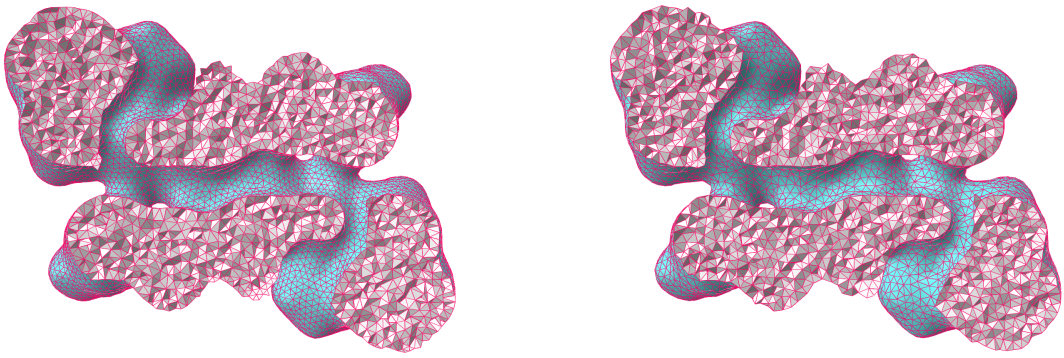

Figure 4.22: Volumetric meshing results on 1MAG. Left: Generated by TetGen; Right: Generated by CGAL.

In our tests, we set the aforementioned five parameters of Delaunay traingulation in CGAL as 30, 1.8, 1.8, 2, and 1.4 respectively. The right cutaway view in Figure 4.22 demonstrates the cross section mesh structure generated by constrained Delaunay triangulation algorithm for protein 1HEW.

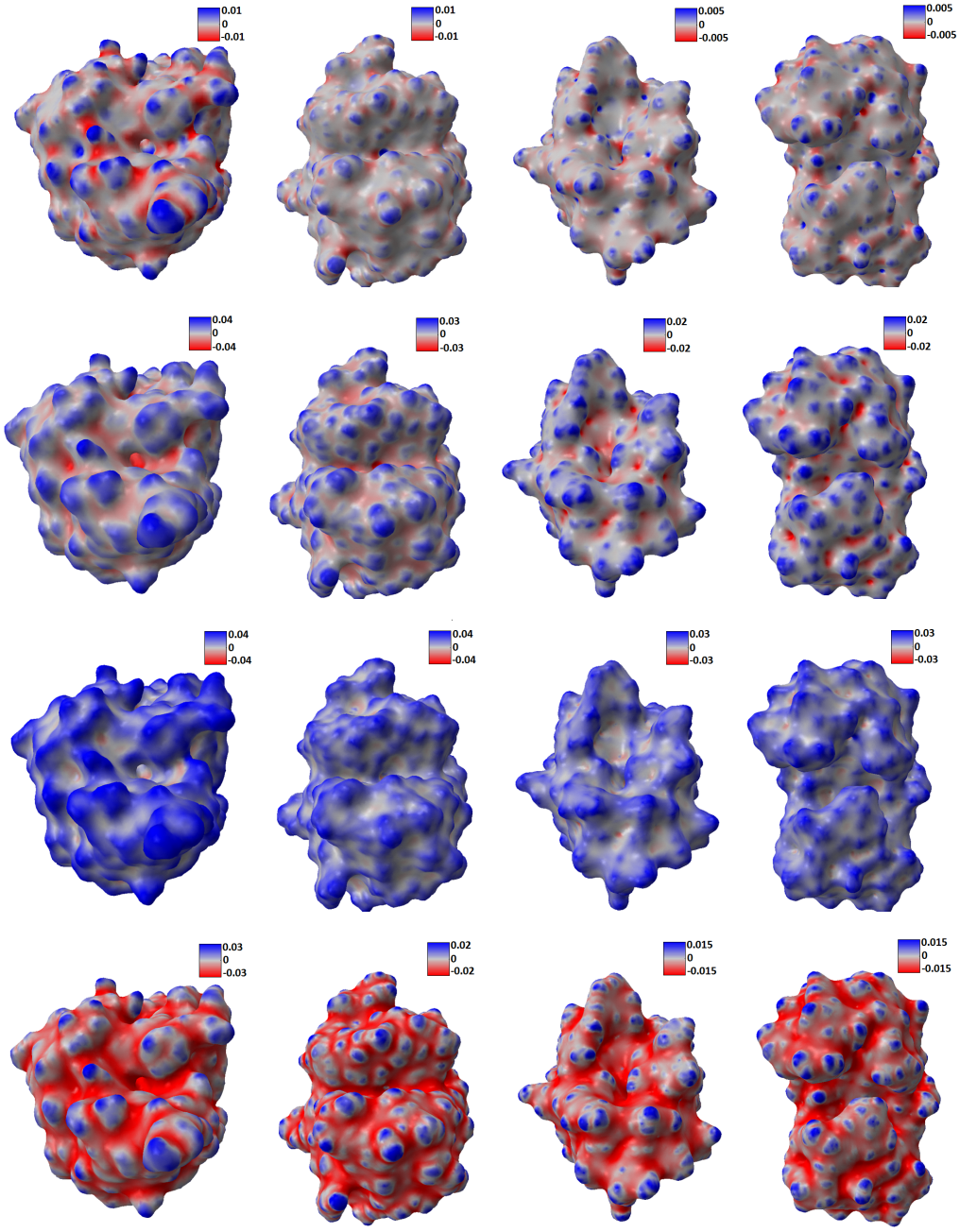

Figure 4.23: Curvature estimation results. From top to bottom: Gaussian, mean, maximum, and minimum curvatures. From left to right: 1ADS, 1BYH, 1EJN, and 2WEB.

The tetrahedralization algorithm provided by the TetGen library also has user-specified parameters to control the mesh quality, including the maximum volume bound on tetrahedra and the cell radius edge ratio bound. We set them to 1 and 1.4 in the test of 1HEW. The cross-section mesh structure generated by the TetGen library is illustrated in the left image of Figure 4.22.

4.4.2.4 Curvature Characterization

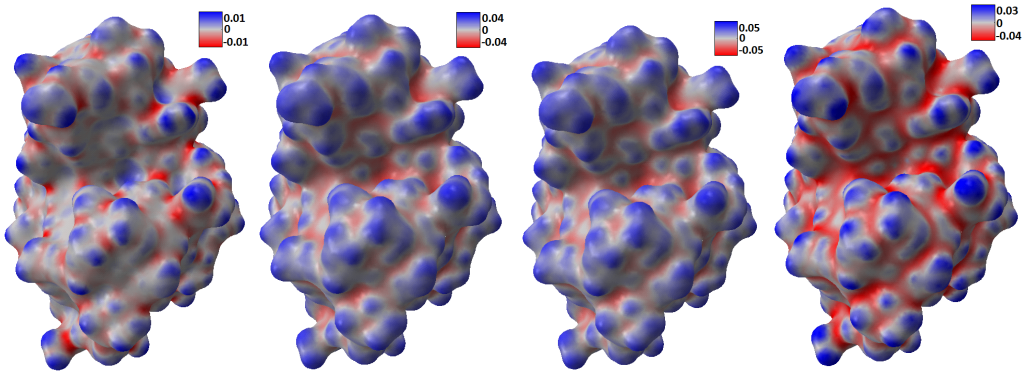

Figure 4.24: Curvature distributions on 1HEW surface with more atomic details. From left to right: Gaussian curvature, mean curvature, maximum curvature, and minimum curvature.

Curvatures describe the geometric features of a protein surface. Surface features can be usually characterized by the Gaussian curvature, mean curvature, maximum and minimum curvatures. The Gaussian curvature measures the intrinsic metric properties of a surface and can be used to distinguish the peak and pit region from the saddle ridge and saddle valley region. In contrast, mean curvature describes the extrinsic properties of a surface. Positive mean curvature is found in regions like peaks, ridges and noisy dots. For the pits and valleys area, the mean curvature assumes negative values. The maximum and minimum curvatures are of fundamental importance. They can be combined with each other to form different surface indices, which provide information about the geometric features. The Gaussian curvature and mean curvature are the product and average of the two parameters, respectively. Another set of shape descriptors, the shape index and curvedness, are also functions of the maximum and minimum curvatures.

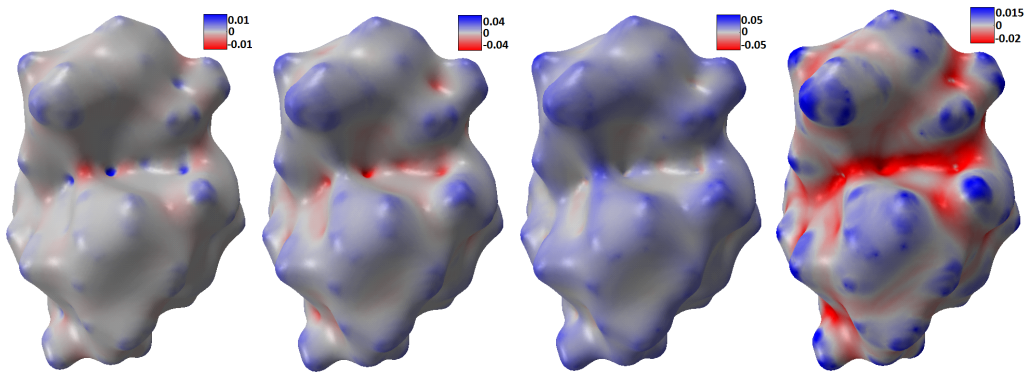

Figure 4.25: Curvature distributions on 1HEW surface with fewer atomic details. From left to right: Gaussian curvature, mean curvature, maximum curvature, and minimum curvature.

In Figure 4.23, we present the calculated estimates for Gaussian curvature, mean curvature, maximum and minimum curvature for four protein data. It can be seen that, these parameters capture the geometric features very well. For instance, the Gaussian curvature estimates with large positive value indicate the tips and pits areas very well. Here we make use of our potential driven molecular surface, which is free from geometric singularities. But even this kind of surface may still contain too much atomic detail. Global features such as the concave area with biological application in protein-ligand binding, cannot be derived straightforwardly from it. Therefore, the multiscale multiresolution model is employed. Using protein 1HEW as an example, we compare the surface generated from different initial conditions. In Figure 4.24, the smaller parameter ($\eta = 1.3$) is used, thus revealing more atomic structures. When we use larger parameter ($\eta = 2.0$), the resulting protein surface highlights global features, as shown in Figure 4.25. It is seen that the latter choice of parameter removes a lot of the surface fluctuations and produces much smoother curvature values. Thus, the global structures of the protein emerge as salient features.

With the consideration that the minimum curvature can be a potential candidate for the indication of concave area, the toon-shading technique, i.e. shading with fewer colors, is used on the protein surface with more visible global features. We set two thresholds κ_{2min} and κ_{2max} with κ_2 representing the minimum curvature. If a vertex's κ_2 value is smaller than κ_{2min}, the vertex is rendered as red. When the vertex's κ_2 value is larger than κ_{2max}, the blue color is assigned to

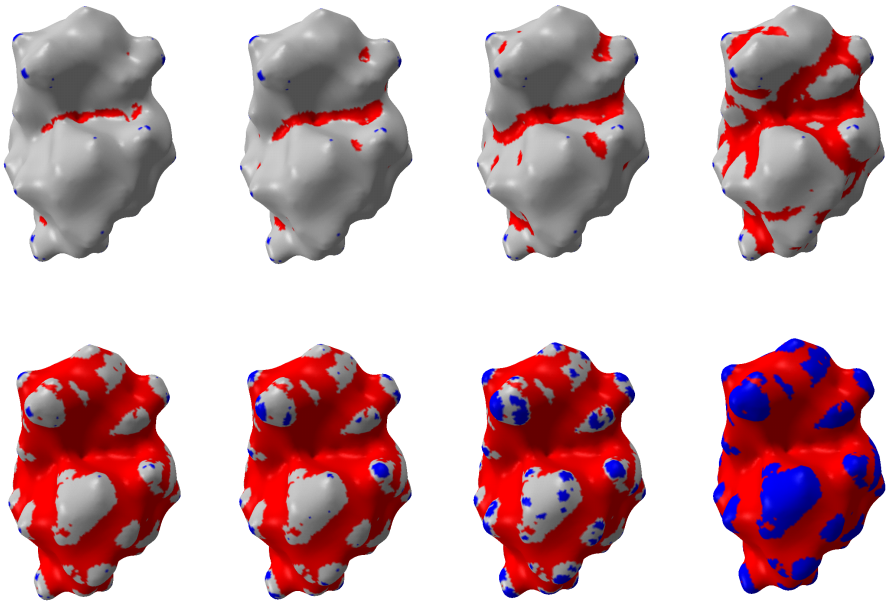

Figure 4.26: The toon-shading of minimum curvature on protein 1HEW.

it. All the vertices with a κ_2 value between κ_{2min} and κ_{2max} assume a grey color. In this way, one can easily tune these two parameters to set proper thresholds to distinguish the valley from other regions on surface. By changing κ_{2min} and κ_{2max} values, we can create a series of pictures by moving one parameter towards zero gradually while keeping the other parameter unchanged.

Table 4.6: Area distributions with the change of the two minimum curvature thresholds in Figure 4.26.

$(\kappa_{2min}, \kappa_{2max})$	(-0.2,0.15)	(-0.15,0.15)	(-0.1,0.15)	(-0.05,0.15)
$(-\infty, \kappa_{2min})$	29.2	101.8	364.2	1373.1
$[\kappa_{2min}, \kappa_{2max}]$	4822.0	4749.4	4487.1	3478.1
$[\kappa_{2max}, +\infty]$	6.7	6.7	6.7	6.7
$(\kappa_{2min}, \kappa_{2max})$	(0,0.15)	(0,0.1)	(0,0.05)	(0,0)
$(-\infty, \kappa_{2min})$	3267.7	3267.7	3267.7	3267.7
$[\kappa_{2min}, \kappa_{2max}]$	1583.5	1529.2	1357.5	0
$[\kappa_{2max}, +\infty]$	6.7	61.1	232.8	1590.2

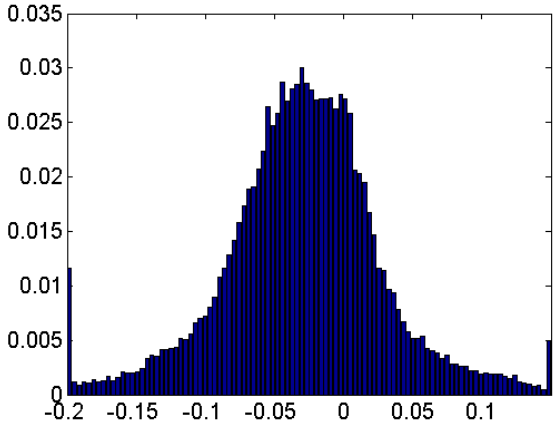

Figure 4.27: The histogram of the minimum curvature on protein 1HEW.

After setting one parameter to zero, we keep it unchanged and move the other parameter towards zero. Through this process, we can observe how the areas below threshold (κ_{2min} or above threshold κ_{2max}) expands. The results are demonstrated in Figure 4.26. Another advantage of using the toonshading technique is that we can quantify the change of the interested area when we adjust the threshold values. This is demonstrated in Table 4.6. The distribution of κ_2 values is demonstrated in Figure 4.27. Overall, the minimum curvature is distributed around 0.03.

4.4.2.5 Electrostatic Analysis

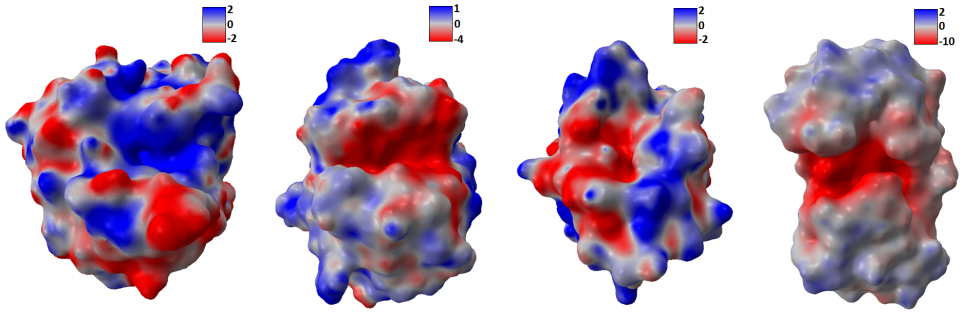

Figure 4.28: Electrostatic potential maps. Left to right: 1ADS, 1BYH, 1EJN, and 2WEB.

The electrostatic information can be obtained from the coupled systems of LB-PB or LB-PNP. In these models, the calculated electrostatic distribution is stored in the volumetric format. That is, the data is on the Cartesian grid with each node associated with an electrostatic value. For each vertex on the protein surface mesh, the tri-linear interpolation is used on the surrounding eight grid points to evaluate the electrostatic value. The results on four proteins, namely 1ADS, 1BYH, 1EJN and 2WEB, are demonstrated in Figure 4.28.

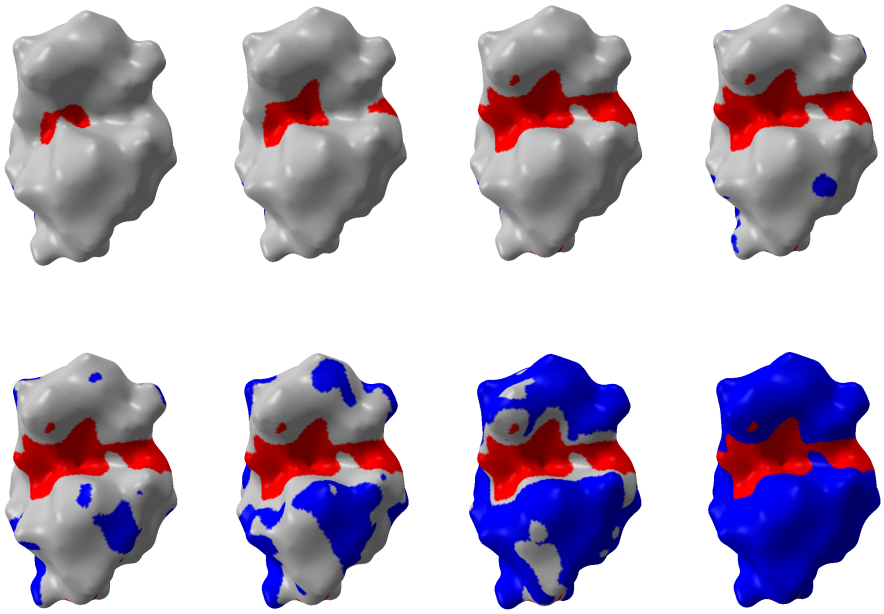

Figure 4.29: The electrostatic potential distribution of protein 1HEW.

Table 4.7: The toonshading results regarding the electrostatic potential distribution of protein 1HEW.

(Φ_{min}, Φ_{max})	(-2, 5)	(-1,5)	(0,5)	(0,4)	(0,3)	(0,2)	(0,1)	(0,0)
$(-\infty, \Phi_{min})$	37.2	107.10539	318.3	318.3	318.3	318.3	318.3	318.3
$[\Phi_{min}, \Phi_{max}]$	4803.1	4733.2	4522.0	4397.0	3942.5	2861.2	1034.6	0
$(\Phi_{max}, +\infty)$	17.7	17.7	17.7	142.7	597.1	1678.4	3505.0	4616.1

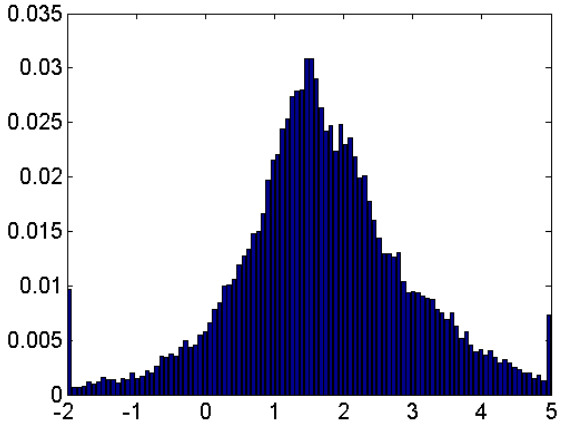

Figure 4.30: Histogram of electrostatic potential distribution on protein 1HEW.

The toonshading method again is used to identify the regions with positive, negative or neutral electrostatic potential values. The protein 1HEW is used here and the basic results is demonstrated in Figure 4.29 and Table 4.7. We also analyze the distribution of the electrostatic potential and present the results in the histogram Figure 4.30. Clearly, the overall surface of protein 1HEW is positively charged.

4.5 Summary

Molecular geometric modeling is fundamental for the conceptual understanding of biomolecular and subcellular structures and interactions. Molecular boundary or molecule shape is a crucial component in molecular geometric modeling. The traditional molecular surface definitions are ad hoc in origin and admit geometric singularities, which lead to computational difficulties in molec-

Table 4.8: The areas with both κ_2 and *pbe* parameter ranges for 1HEW model.

(Φ_{min}, Φ_{max})	$(-\infty, -1)$	$[-1, 2]$	$(2, +\infty)$
$(-\infty, -0.1)$	37.3	172.9	86.9
$[-0.1, 0.1]$	72.3	2887.8	1554.7
$(0.1, +\infty)$	0	21.6	24.4

ular dynamics, energy estimations and curvature calculations. Additionally, traditional geometric models are usually detached from physical modeling, which leads to extra parameterizations for the entire theoretical model. The work in this chapter presents a variational multiscale strategy for the unified geometric and physical modeling of aqueous biomolecular systems. We first discuss a variational model for the surface tension effect of a biomolecule in solvent. The Euler-Lagrange variation of the surface energy functional gives rise to the Laplace-Beltrami equation which determines the minimal molecular surface (MMS) of a biomolecule in solvent. Additionally, we take into consideration of cavitation and solvent-solute interactions to obtain a nonpolar solvation model. The addition of electrostatic energy in the energy functional gives us a full solvation model. At a non-equilibrium setting, we further employ Fick's laws to define a concentration flux and characterize flow motion. We use geometric measure theory to embed a two-dimensional (2D) surface in the 3D Euclidean space via a hypersurface function, which separate the microscopic region of the biomolecule from the macroscopic domain of the solvent. In all of our models, the generalized Laplace-Beltrami equation comes up with the geometric definition of the biomolecular surfaces. The Laplace-Beltrami equation is complemented by the generalized Poisson-Boltzmann and Nernst-Planck equations to describe respectively the electrostatic potential and solvent density in aqueous environment.

From the hypersurface function and its governing generalized Laplace-Beltrami equation, we introduce three approaches for multiresolution analysis of biomolecular surfaces. The first method is to generate multiresolution surfaces via appropriate initial conditions of the hypersurface function. The second approach is to create multiresolution analysis from different evolution durations of the generalized Laplace-Beltrami equation. Finally, proper selections of the isovalues in the isosurface extraction also lead to desirable surface resolutions. In general, fine resolution surfaces are suitable for the local analysis of solvent-solute interactions and ion channel gating, where the detail atomic features matters. In contrast, coarse-scale surfaces are appropriate for the characterization of global features, such as concave regions and convex regions, which are related to protein-DNA specification, protein-ligand binding and protein-protein interaction.

Based on the new multiresolution surface representations, two commonly used surface extraction methods, the marching cubes algorithm and the Delaunay based method in CGAL are discuss. The marching cubes method is relatively straightforward and fast. But its result meshes suffer from skinny triangles. The Delaunay based method incorporates adjustable parameters to control the mesh quality and the resulting high quality meshes are suitable for finite element modeling and curvature characterization. Alternatively, CGAL's remeshing functions can be used to improve the mesh quality of surfaces generated from the marching cubes algorithm. Once the surface mesh is obtained, volume mesh generation techniques are employed. The volume mesh structures provide the necessary information for finite element or finite volume analysis. In this chapter, a constrained Delaunay triangulation algorithm is implemented.

In protein-protein and protein ligand interactions, geometric features and electrostatic potential distributions play important roles. Especially in rational drug design, the drug binds to the regions of the protein with complementary electrostatic potential and matching (concave) curvatures. We compute electrostatic potentials associated with multiresolution surfaces. The resulting electrostatic maps are displayed in both continuous scales and discrete levels labeled with different pseudo-colors.

We carry out curvature characterization of various surface features, such as peak, pit, ridge, flat valley, saddle ridge, minimal surface, and saddle valley. These features are associated with appropriate signs of Gaussian curvature and mean curvature. We also develop minimum principle curvature descriptor and maximum principle curvature descriptor for identifying concave and convex regions, respectively. The utility of these curvature methods is amplified when they are performed hand-in-hand with our multiresolution surface representations. The further combination of curvature characterization, electrostatic map and multiresolution representation gives rise to a potential approach for the analysis on solvation, protein-ligand binding and protein-protein interaction.

We have also performed extensive tests on modeling and analyses on cryo-EM data. We demonstrated the efficiency of high order geometric PDEs for noise removal of cryo-EM data.

We investigated the performance of marching cubes and CGAL schemes for surface extraction in cryo-EM datasets. Specifically, we analyze the performance of four algorithms, the isosurface stuffing [105], TetGen [158], NetGen [148], Delaunay refinement and interleaved mesh optimization [171] for the volumetric meshing of these datasets. Informative results are found in curvature analysis. It is found that the maximum and minimum curvature maps of cryo-EM complexes can be used for binding site characterization. Specifically, the maximum curvature can also be used to exclude regions from the binding targets of small molecules, while the minimum curvature serves a promising indicator of binding targets.

Chapter 5

GEOMETRIC MODELING ON BIOMOLECULAR MODELS — CARTESIAN REPRESENTATION

5.1 Introduction

The objective of the present chapter is to provide an expository investigation and summary of tools, algorithms and methodologies for geometric modeling of biomolecules. We particularly focus on tools, algorithms and methodologies required for biophysical models in the Eulerian representation. Although Eulerian formulation [34] and Lagrangian formulation [35] of biomolecular surfaces can be formally equivalent, they depend on different tools, algorithms and methodologies. The starting points of our discussions are experimental data from either the PDB or the EMDB. For the latter, the high order PDEs are introduced to perform noise reduction. Geometric features, such as Gaussian curvatures, mean curvatures, and shape index, are employed to describe the geometric properties of biomolecular multiresolution surfaces generated by generalized geometric flows and from the EMDB for the first time.

The rest of this chapter is organized as follows. Section 5.2 is devoted to computational algorithms. We discuss in great detail the data sources, related software packages, and computational techniques for surface construction, quality improvement, and geometric characterization. We provide advanced interface methods for the evaluation of surface area and surface enclosed volume in the Cartesian representation. Efficient algorithms for calculating various curvature properties, such as Gaussian curvature, mean curvature, maximum and minimum principal curvatures, shape index, and curvedness are developed. The performance of these algorithms is compared. This chapter ends with concluding remarks in Section 5.3.

5.2 Computational Algorithms

5.2.1 PDB Data Processing and Surface Generation

The PDB (http://www.rcsb.org) is a repository for the 3D structural data of macromolecules, usually obtained by X-ray crystallography or NMR spectroscopy. Most data downloaded from the PDB need to be processed for preparing structures used in theoretical analysis and modeling [32]. Visualization is of great importance to our understanding and conceptualization of the biomolecular systems. Many software packages can be employed to generate triangular surface meshes for biomolecules. An example is the MSMS package. However, the MSMS surface cannot be directly used in Cartesian domain modeling and computation as discussed below.

5.2.1.1 Lagrangian to Eulerian Transformation

The molecular surface generated from the MSMS software is in the Lagrangian representation, i.e., triangle meshes are used to describe the surface. In order to generate the Cartesian representation for finite difference type of methods, one needs to carry out the transformation from Lagrangian to Eulerian representation, i.e., to immerse the 2D surface obtained from the Lagrangian representation into a bounded 3D domain with the Cartesian grid. In this process, one needs to extract interface information from the triangle mesh representation, including the coordinates of intersecting points between the surface and Cartesian mesh lines, and surface normal directions at these intersecting points.

For example, if we have a surface mesh in .vert and .face files, usually, the .vert file stores the point coordinates in the form of $\mathbf{v} = (v_x, v_y, v_z)$, and the .face file contains the connectivity information with each triangle represented by three vertex indices. The bounded box to encompass the protein can be constructed by expanding the tightest axis-aligned bounding box, i.e., by decreasing (increasing) the minimal (maximal) values of surface coordinates, by a certain value denoted as

d_c. The new Cartesian mesh domain is thus $[x_l, x_r] \times [y_l, y_r] \times [z_l, z_r]$, and can be obtained from,

$$\begin{aligned} x_l &= \min_{m=1,\ldots,N_t} (v_{m,x}) - d_c, \\ y_l &= \min_{m=1,\ldots,N_t} (v_{m,y}) - d_c, \\ z_l &= \min_{m=1,\ldots,N_t} (v_{m,z}) - d_c, \\ x_r &= \max_{m=1,\ldots,N_t} (v_{m,x}) + d_c, \\ y_r &= \max_{m=1,\ldots,N_t} (v_{m,y}) + d_c, \\ z_r &= \max_{m=1,\ldots,N_t} (v_{m,z}) + d_c, \end{aligned} \tag{5.1}$$

where N_t is the total number of the node points in the Lagrangian representation of the protein surface. One can specify the mesh spacing, i.e., the size of each grid, as h, and coordinates of Cartesian mesh nodes can be calculated and represented as $\{(x_i, y_j, z_k) | i = 1, \ldots, N_x; j = 1, \ldots, N_y; k = 1, \ldots, N_z\}$, with N_x, N_y and N_z standing for the total node numbers in each dimension. It can be seen that $x_l = x_1$ and $x_r = x_{N_x}$. Similar relations exist for y and z coordinates.

As the goal is to find the intersection points of each triangle with grid lines, we first find the plane equation. For each mesh triangle, one has the coordinates for its three vertices ($\mathbf{v}_1, \mathbf{v}_2$ and $\mathbf{v}_3$). The 2D plane that the triangle belongs to is

$$ax + by + cz + d = \begin{vmatrix} x - v_{1,x} & y - v_{1,y} & z - v_{1,z} \\ v_{2,x} - v_{1,x} & v_{2,y} - v_{1,y} & v_{2,z} - v_{1,z} \\ v_{3,x} - v_{1,x} & v_{3,y} - v_{1,y} & v_{3,z} - v_{1,z} \end{vmatrix} = 0. \tag{5.2}$$

The norm for the triangle is the same for the plane, represented as

$$\mathbf{n} = \left(\frac{a}{\sqrt{a^2 + b^2 + c^2}}, \frac{b}{\sqrt{a^2 + b^2 + c^2}}, \frac{c}{\sqrt{a^2 + b^2 + c^2}} \right). \tag{5.3}$$

We find the intersection points by testing grid edges within the bounding box of the triangle. It is easy to find the coordinate ranges for all the relevant grid edges, e.g. in x coordinate,

$$x_s = \min(v_{1,x}, v_{2,x}, v_{3,x}), \tag{5.4}$$

$$x_b = \max(v_{1,x}, v_{2,x}, v_{3,x}). \tag{5.5}$$

For all the points within the triangle, the values of x coordinate should fall in the range $[x_s, x_b]$. For any Cartesian grid line with x-coordinate x_i in $\{x_i | x_s \leq x_i \leq x_b\}$, the index i satisfies the restriction $i_0 \leq i \leq i_1$, where $i_0 = \lceil x_s/h \rceil$ and $i_1 = \lfloor x_b/h \rfloor$ with h being the grid spacing. Similarly, one can find similar lower and upper limit integers for the other two coordinates, $j_0 \leq j \leq j_1, k_0 \leq k \leq k_1$. Thus, to find an intersection between a surface triangle and a grid line along x direction, one can choose two arbitrary index (j,k) within the corresponded ranges, with their associated coordinates defined as (y_j, z_k), and calculate the related point in the triangle plane. The related coordinates are denoted as (x_o, y_j, z_k), and evaluated from

$$ax_o + by_j + cz_k + d = 0. \tag{5.6}$$

The intersecting points form a set, which is the collection of only three possible types of points:

$$\{(x_o, y_j, z_k) | j_0 \leq j \leq j_1; k_0 \leq i \leq k_1; ax_o + by_j + cz_k + d = 0\}, \tag{5.7}$$

$$\{(x_i, y_o, z_k) | i_0 \leq i \leq i_1; k_0 \leq k \leq k_1; ax_i + by_o + cz_k + d = 0\}, \tag{5.8}$$

$$\{(x_i, y_j, z_o) | i_0 \leq i \leq i_1; j_0 \leq j \leq j_1; ax_i + by_j + cz_o + d = 0\}. \tag{5.9}$$

The only task left to do is to determine whether the planar point we calculate falls within the triangle. The points located outside the triangle are discarded. If the point located on the boundary edges or the interior of the triangle, it is indeed a point where the interface intersects with the Cartesian grid lines. The normal vector for this interface intersecting point is defined to be the same as that of the triangle, or for efficiency, it can be computed as the linear interpolations of vertex normals. The normals and coordinates are then stored in the sequence of their related Cartesian nodes.

We test our method on a sphere with radius $r = 2$. Using the MSMS software, we generate the Lagrangian representation of the surface with 100 vertices for each 1×1 area. The Cartesian representation is set with a mesh spacing h and the interface-mesh intersecting points are calculated and the average error is evaluated by

$$\text{Error} = \sum \frac{|r - \sqrt{x_o^2 + y_o^2 + z_o^2}|}{N_o}, \tag{5.10}$$

Table 5.1: Test of the convergence of the proposed method for Lagrangian to Eulerian transformation.

h	Error	Order
2.500e-1	7.985e-4	
1.250e-1	2.651e-4	1.59
6.250e-2	7.265e-5	1.87
3.125e-2	1.979e-5	1.88

where (x_o, y_o, z_o) are the calculated interface-mesh intersecting points and N_o are the total number of such intersecting points. The errors of Lagrangian to Eulerian transformation are illustrated in Table 5.1. It can be seen from the table that second order accuracy is attained.

5.2.1.2 Surface Generation in Cartesian Representation

The basic idea for surface generation is to embed an enlarged van der Waals surface in a 3D domain and evolve this hypersurface under a geometric and potential driven flow under certain biological constraint. Note that directly evolving the geometric flow equation in the Lagrangian representation for a protein may be unstable due to the possible topological changes during the surface evolution. In the Cartesian setting, some basic information of the protein is needed, including atom positions $\mathbf{x}_i, i = 1, \cdots, n$, atom radii $r_i, i = 1, \cdots, n$ and also the atomic charges information for electrostatic analysis when a full solvation model is used. Here n is the total number of the atoms in the protein molecule. To set up the initial conditions, two domains are defined, one is D_χ representing the domain enclosed by the van der Waals surface; the other is an enlarged domain D:

$$D_\chi = \cup_{i=1}^{n} \{\mathbf{x} : |\mathbf{x} - \mathbf{x}_i| < r_i\}; \tag{5.11}$$

$$D = \cup_{i=1}^{n} \{\mathbf{x} : |\mathbf{x} - \mathbf{x}_i| < 1.3 r_i\}. \tag{5.12}$$

Here we choose a factor of 1.3 to guarantee the formation of properly connected surfaces. In fact, this special parameter can be adjusted to give different scales of the molecular surface, which may lead to dramatically different geometric features [64]. We denote S as the Cartesian representation

of the hypersurface. For the initial value of S, we consider two functions,

$$S(x,y,z,0) = \begin{cases} 1, (x,y,z) \in D \\ 0, \text{ otherwise.} \end{cases} \tag{5.13}$$

$$\chi(x,y,z) = \begin{cases} 1, (x,y,z) \in D_\chi \\ 0, \text{ otherwise.} \end{cases} \tag{5.14}$$

The characteristic function $\chi(x,y,z)$ is used to protect the van der Waals surface during the surface evolution. The Dirichlet boundary condition is used in the computation, the boundary value is $S = 0$. The formation of surface is driven only by the generalized Laplace-Beltrami equation (4.2) in Section 4.2. We spell out all the terms involved,

$$\frac{\partial S}{\partial t} = \gamma \left[\frac{(S_x^2+S_y^2)S_{zz} + (S_x^2+S_z^2)S_{yy} + (S_y^2+S_z^2)S_{xx}}{S_x^2+S_y^2+S_z^2} \right.$$
$$\left. - \frac{2S_xS_yS_{xy} + 2S_xS_zS_{xz} + 2S_zS_yS_{yz}}{S_x^2+S_y^2+S_z^2} + \frac{\sqrt{S_x^2+S_y^2+S_x^2}}{\gamma} V_1 \right],$$

where γ is the surface tension and V_1 is a general potential driven term due to other effects. We treat protein surface tension as a fitting parameter in the free energy calculation of a set of molecules [34, 35]. To take into consideration the biological constraints, we modify the evolution equation and incorporate the characteristic function $\chi(x,y,z)$,

$$\frac{\partial S}{\partial t} = (1-\chi(x,y,z))\gamma \left[\frac{(S_x^2+S_y^2)S_{zz} + (S_x^2+S_y^2)S_{yy} + (S_y^2+S_z^2)S_{xx}}{S_x^2+S_y^2+S_z^2} \right.$$
$$\left. - \frac{2S_xS_yS_{xy} + 2S_xS_zS_{xz} + 2S_zS_yS_{yz}}{S_x^2+S_y^2+S_z^2} + \frac{\sqrt{S_x^2+S_y^2+S_x^2}}{\gamma} V_1 \right].$$

The approximated steady state solution of $S(x,y,z,t)$ is obtained after certain large iteration time, $t = T_0$. It is a smooth function with relatively rapid changes near the protected atomic boundaries of D_χ. However, the hypersurface $S(x,y,z,T_0)$ gives rise to a family of isosurfaces. It is easy to extract an isosurface by setting $S(x,y,z,T_0) = C$, where C is a value between 0 and 1. The value of C can be adjusted to achieve the effect of multiresolution surfaces. However, by choosing $C = 0.5$,

one can attain a better accuracy in the calculations of the surface area and surface enclosed volume. For all the surfaces demonstrated in this paper, we choose $C = 0.5$. Figure 5.1 gives an example of the comparison between the molecular surface and the surface generated from the Laplace-Beltrami flow. It can be seen from the figure that the surface evolved from the Laplace-Beltrami is free from singularities.

When there are external potentials ($V_1 \neq 0$), the surface generation is usually coupled with the calculation of other physical variables governed by other equations. These coupled equations should be solved iteratively. For example, to take into consideration the electrostatic effect, the PB model is commonly employed. Due to the vastly different quantities of the dielectric constants in solute and solvent domains, the elliptic interface problems are frequently updated in the geometric and potential driven models.

5.2.2 EMDB Data Processing and Surface Generation

5.2.2.1 EMDB Data

As the data from the cryo-EM accumulated, EMDataBank.org(http://emdatabank.org/index.html) was established to create a global deposition and retrieval network for cryo-EM maps and associated metadata. It also serves as a portal of software tools for standardized map format conversion, segmentation, model assessment, visualization, and data integration. A list of EM software packages can be found in the website (http://emdatabank.org/emsoftware.html). MRC (Medical Research Council) is the file format used in cryo-EM data, in which the data are stored on a 3D grid of voxels (volumetric cells) with values corresponding to the density of electrons. It was developed by the MRC Laboratory of Molecular Biology, and is supported by almost every molecular graphics software package that supports volumetric data, such as, visual molecular dynamics (VMD), PyMOL, and UCSF Chimera. A detailed specification of the MRC file format can be found at the website. The Matlab code for extracting the voxel value information is also mentioned in the above webpage. Here we modify the code and incorporate a simple procedure to store the values in the

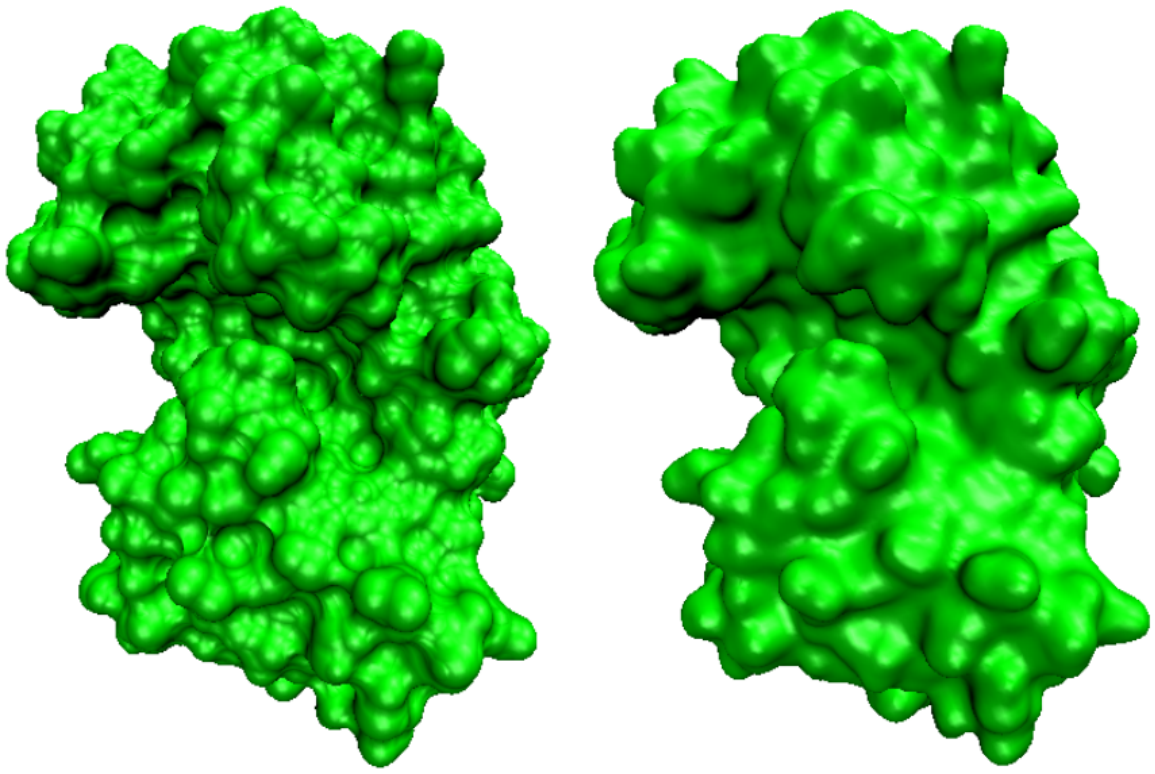

Figure 5.1: Comparison of the solvent excluded surface (Left) of protein 1PPL and surface generated by the Laplace-Beltrami flow with $V_1 = 0$ (Right). The latter is free from geometric singularity. In the generation of 1PPL MMS, an outer layer of 1.7Å is used to immerse the protein in solvent. The computational domain for protein 1PPL is [-14.7, 57.8]*[-16.7,41.3]*[-8.2,39.8]. We set the grid size to be 0.5 Å, and 100 iterations are carried out.

".dat" format for further use.

To avoid any confusion, here "emd_1048.map" contains the data directly downloaded from EMDataBank's website (http://www.ebi.ac.uk/pdbe-srv/emsearch/form) and is stored in the standard MRC format. With VMD, one can visualize the data directly. In the VMD, when loading the data, one needs to select the "CCP4,MRC density map" option for "Determine file type". In the "Graphical representation", the "Drawing method" is chosen to be "Isosurface". For the "isovalue" option, one needs to key in the recommended iso-values found in the webpage of the map data. One can select the "ColorID" in the "Coloring method" and adjust the value to render the surface with a specific color.

5.2.2.2 Noise Reduction of EMD

It is seen from the left chart in Fig. 5.2 that electron tomography sometimes produces extremely noisy and low contrast 3D density maps. The poor signal-to-noise ratio (SNR) hinders visualization and interpretation. Therefore, noise filtering techniques are indispensable when treating EM data. There are a number of effective methods and schemes for this task, like wavelet transform techniques, nonlinear anisotropic diffusions, Beltrami flow, bilateral filter, and iterative median filtering [164, 65, 66, 170, 95, 132, 173].

To further improve the noise reduction efficiency, high order geometric PDEs are employed, see Section 4.3.2. We can control the process by three parameters, the integration time t, the PDE order q, and the external term $P(u, |\nabla u|)$. For the protein surface generation from the PDB, a proper combination of the PDE order and integration time is required to deliver high-quality surface [212, 64]. If the solvation process is considered, the potential driven term should incorporate both the nonpolar and polar effects. In our noise reduction procedure for EMDB data, the external term is omitted. The integration time is adjusted to give different levels of noise amplitude and image construction. Figure 5.2 demonstrates an example of noise removal of EMD5119. To be specific, left chart in Fig. 5.2 is generated with the suggested contour level value 0.25 from the original noisy data. The right chart is produced with same contour level value but from the processed data. The noise is reduced efficiently, while the salient edges are preserved very well.

5.2.3 Surface Electrostatic Analysis

One of the most important problems in biological sciences is the understanding of electrostatic interactions in biomolecules. Electrostatic interactions are ubiquitous in any system of charged or polar molecules, such as proteins, nucleic acids, lipid bilayers, and sugars. The importance of electrostatics in biomolecular systems is due to the fact that electrostatic interactions frequently dominate other forces and determine the structure, function, stability, dynamics and transport of macromolecular systems. As shown in Section 4.2, electrostatic analysis is readily coupled with surface analysis. Electrostatic potential can be obtained by solving the Poisson-Boltzmann equa-

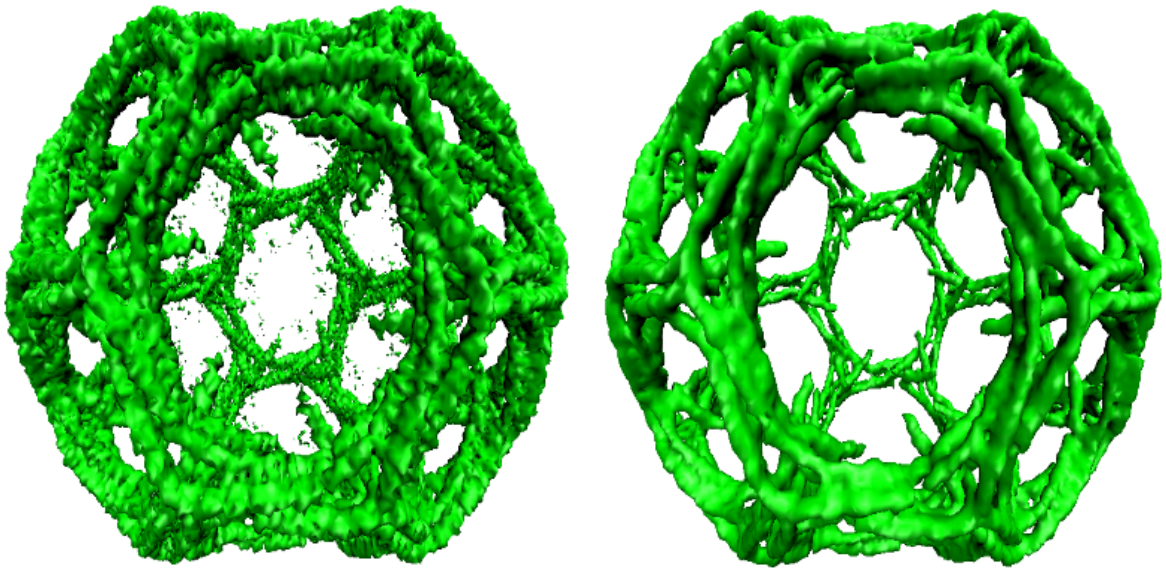

Figure 5.2: Noise reduction for emd5119. The left chart shows the noisy image from the original density maps; The right chart shows the image free from the noise.

tion. The surface electrostatic potential, obtained by the projection of electrostatic potential on a surface, is important for the understanding of protein-protein interactions, ligand binding, solvation, and drug design.

From the mathematical point of view, solvent-solute boundary can be treated as an interface. If we use the Poisson equation or the PB equation to describe the electrostatic potential with the different dielectric constants in the solvent and solute domains, an elliptic interface problem is constructed. The well-posedness of this equation relies on the interface information, which usually involves the jump conditions of the function values and the derivatives with respect to normal directions on the interface.

$$[u] = u^+ - u^- \qquad = \Psi_1, \ \forall \mathbf{x} \text{ on } \Gamma \tag{5.15}$$

$$[\beta u_{\mathbf{n}}] = \beta^+ u_{\mathbf{n}}^+ - \beta^- u_{\mathbf{n}}^- \qquad = \Psi_2, \ \forall \mathbf{x} \text{ on } \Gamma \tag{5.16}$$

where Γ denotes the interface, and vector $\mathbf{n}$ is the normal direction. Here $u^+, u_{\mathbf{n}}^+$ and β^+ denote the limiting values of from one subdomain Ω^+, and $u^-, u_{\mathbf{n}}^-$ and β^- from the other Ω^-. The total computational domain $\Omega = \Omega^+ \bigcup \Omega^-$, and interface $\Gamma = \Omega^+ \bigcap \Omega^-$. In the PB model, the variable

u is replaced by the electrostatic potential Φ. Due to the continuity of electrostatic potential and its flux density, the right terms in the jump condition equals zero, that is, $\Psi_1 = \Psi_2 = 0$.

5.2.3.1 Extraction of Interface Information from Volumetric Data

Interface information is required for both electrostatic analysis and geometric analysis. To extract interface information from volumetric data, one needs to know the isovalues (or level set values) at the Cartesian grid nodes. The volumetric data can be treated as a surface function on a grid with one value assigned to each grid node. When a new Cartesian mesh is employed in computation, the isovalues on the new grid nodes need to be evaluated for further applications in elliptic interface schemes and curvature analysis. For instance, if one has volumetric data $\{S_v\}_{320\times320\times320}$, the Cartesian mesh size is set to $N_x \times N_y \times N_z$ ($21 \times 21 \times 21$ in the example) , and $\{S_v\}$ should be sampled on the grid to produce $\{S\}_{N_x\times N_y\times N_z}$. Here $\{S\}_{N_x\times N_y\times N_z}$ can be seen as the discrete representation of the trilinearly interpolated surface function $S(x,y,z)$. We provide details for the trilinear interpolation below. First, we assume the domain for given volumetric data as $\Omega_v = [1,320]\times[1,320]\times[1,320]$, and denote the coordinates of node (i,j,k) on target Cartesian grid by (x_i,y_j,z_k), expressed as

$$\left(x_i,y_j,z_k\right) = \left(320\frac{i-1}{21}+1, 320\frac{j-1}{21}+1, 320\frac{k-1}{21}+1\right). \tag{5.17}$$

Then, we denote the integer part of (x_i,y_j,z_k) as (i_t,j_t,k_t), and the fractional part as (x_d,y_d,z_d). The Cartesian mesh node (i,j,k) can be viewed as a point in Ω_v, and encompassed by the cube formed by eight original grid nodes, with coordinates $(\mathbf{v}_m)_{m=1,\ldots,8}$. It is seen that the coordinates for diagonal two nodes are $\mathbf{v}_1 = (i_t,j_t,k_t)$ and $\mathbf{v}_8 = (i_t+1, j_t+1, k_t+1)$. If the isovalues on these grid nodes are denoted as $S_v(\mathbf{v}_m)_{m=1,\ldots,8}$, to calculate the isovalues $S_{i,j,k}$, eight weights $W(m)_{m=1,\ldots,8}$ are needed in the interpolation form,

$$S_{i,j,k} = \sum_{m=1}^{8} S_v(\mathbf{v}_m)W(m). \tag{5.18}$$

One can certainly choose more than 8 points to carry out the evaluation and also the weights are by no means unique. Here we just make use of the Lagrangian shape functions on cubes, and map

the original cube to a logical cube with the coordinate of the diagonal two nodes as $(-1,-1,-1)$ and $(1,1,1)$. The mesh point (i,j,k) is then projected to a new node with coordinate (ξ,η,ζ),

$$(\xi,\eta,\zeta) = (2\cdot x_d - 1, 2\cdot y_d - 1, 2\cdot z_d - 1)\,. \tag{5.19}$$

The weight functions corresponded to cubic nodes can be represented as

$$W(m) = \frac{1}{8}(1+\xi\xi_m)(1+\eta\eta_m)(1+\zeta\zeta_m), \quad m = 1,2,...,8, \tag{5.20}$$

where (ξ_m,η_m,ζ_m) are the nodal coordinates of the logical cube.

For volumetric data, a recommended isovalue is given to define the interface, denoted as Γ. Therefore, the region with isovalues bigger than the recommended one is specified as the biomolecular subdomain, which we usually denote as Ω^+ and with the opposite Ω^-. Each mesh node is assigned to a region. For a given node, when its surrounding six nodes are in the same subdomain, this node is defined as a regular node. Otherwise if any of its six surrounding nodes is located in the other subdomain, the node is an irregular node. Irregular nodes usually occur in pairs.

The real physical domain for the voxel data can be also found from the EMDB data description. The unit is usually in angstrom or nanometer. It is easy to interpolate the physical coordinates into Cartesian mesh nodes. To avoid heavy notation, we still use (x_i,y_j,z_k) to represent the coordinate of node (i,j,k). However, it is now assumed to be in the real physical domain.

In the matched interface boundary (MIB) algorithm, in order to implement the jump conditions, we need to know the interface information between the pair of irregular nodes. For example, if nodes (i,j,k) and $(i+1,j,k)$ are located in two different subdomains, the coordinate of the interface intersecting with the mesh is specified as $\mathbf{v}_o = (x_o,y_o,z_o)$,

$$\mathbf{v}_o = \left(\frac{S_0 - S_{i,j,k}}{S_{i+1,j,k} - S_{i,j,k}}(x_{i+1} - x_i) + x_i, \quad y_j, \quad z_k\right), \tag{5.21}$$

where S_0 stands for the recommended isovalue of the interface, and $S_{i,j,k}$ represents the isovalue at node (i,j,k). The normal direction is interpolated from the expression,

$$\mathbf{n}_o = \frac{S_0 - S_{i,j,k}}{S_{i+1,j,k} - S_{i,j,k}}(\mathbf{n}_{i+1,j,k} - \mathbf{n}_{i,j,k}) + \mathbf{n}_{i,j,k}, \tag{5.22}$$

where $\mathbf{n}_o$ is the normal direction at interface intersecting with mesh line point. The value of $\mathbf{n}_{i+1,j,k}$ can be evaluated from

$$\mathbf{n}_{i+1,j,k} = \left(\frac{S_{i+2,j,k} - S_{i,j,k}}{x_{i+2} - x_i}, \frac{S_{i+1,j+1,k} - S_{i+1,j-1,k}}{y_{j+1} - y_{j-1}}, \frac{S_{i+1,j,k+1} - S_{i+1,j,k-1}}{z_{k+1} - z_{k-1}} \right). \tag{5.23}$$

The value for $\mathbf{n}_{i,j,k}$ can be calculated in the same manner.

5.2.3.2 Solution of Poisson-Boltzmann Equation

In the MIB method, the Cartesian grid is employed. In its numerical schemes, the interface jump conditions are employed only at the intersecting points between the interface and the mesh lines. If the interface is analytically given, for instance, a sphere with certain radius, the coordinates of intersecting points can be easily determined when the mesh size is specified. However, for volumetric data from EMDataBank, an interpolation procedure is required. A detailed description is given below.

The MIB method has been developed to solve the elliptic interface problems with geometric singularities [213, 214, 200, 201, 76, 207]. It delivers the second order accuracy in solving the PB equation with complex protein interfaces, possible geometric singularities and charge singularities. The essential ideas of the MIB method are the following: The standard finite difference schemes are used on the simple Cartesian grids; the fictitious values are employed near the interface as a smooth extension of the non-smooth functions; interface jump conditions are incorporated into the calculation of the fictitious values; and to construct high-order schemes, the lowest order jump conditions are used repeatedly. For the PB equation, one more challenge is the charge singularities, which represent the partial charges of protein atoms assigned by the CHARMM or AMBER force field. In the PB equation, partial charges are represented by Dirac delta functions in the source term. Through the use of the Green's function formulation, the charge singularities are transformed into interface flux jump conditions, which are integrated into the geometric singularities framework [76].

When the Laplace-Beltrami equation is coupled with the PB equation, they should be solved

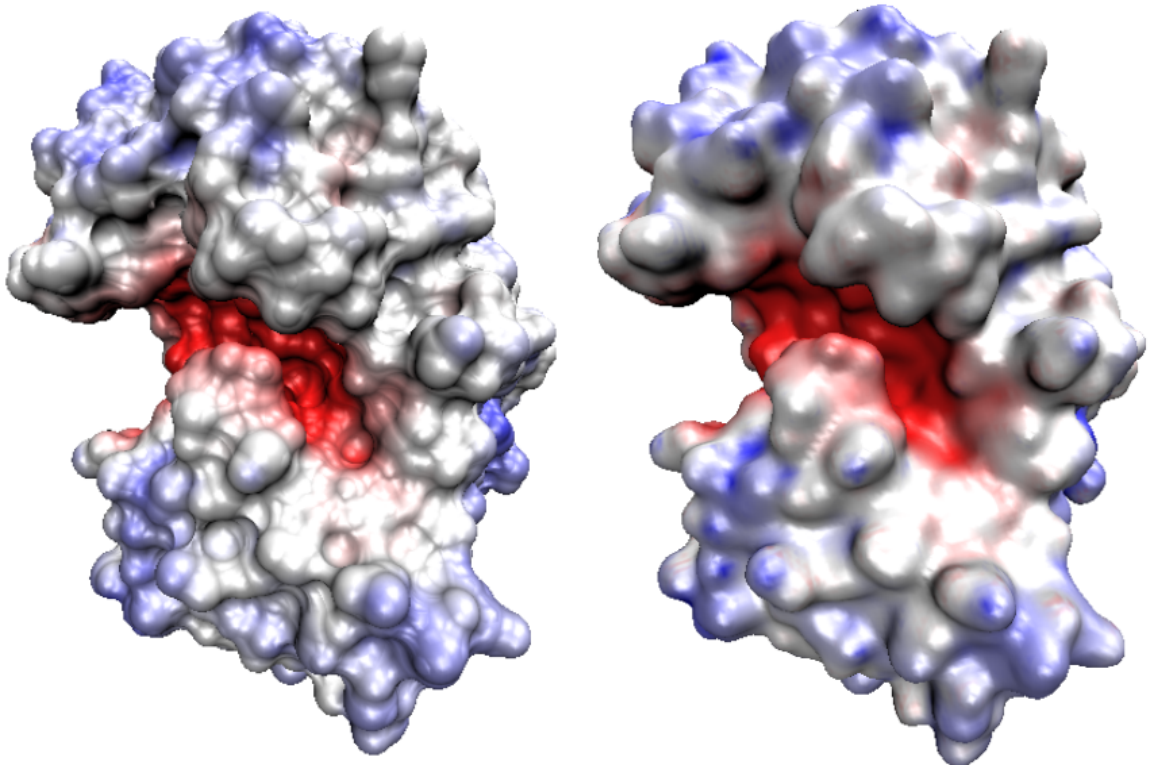

Figure 5.3: Comparison of electrostatic potential distributions on a molecular surface (Left) and a surface generated from a generalized Laplace Beltrami equation (Right) for protein 1PPL.

iteratively until self-consistency is reached [34, 35]. Two approaches have been employed. One approach is a simple relaxation algorithm: the characteristic function S and electrostatic potential ϕ is updated by a linear combination of the previous ones and the new ones. Basically, we start with the initial condition of hypersurface function S_0. This value is used in the PB equation to calculate a temporary electrostatic potential ϕ. The calculated ϕ value is then used in the Laplace-Beltrami equation to evaluate the new S. Instead of using this new S as the new input in the PB equation, we use a weighted average as described below,

$$S_{n+1,new} = \alpha S_{n+1,old} + (1-\alpha)S_n, \quad 0 \leq \alpha \leq 1, \tag{5.24}$$

where $S_{n+1,new}$ is the one applied to the evaluation of the electrostatic potential $\phi_{n+1,old}$. Once we have $\phi_{n+1,old}$, the electrostatic potential is updated with as,

$$\phi_{n+1,new} = \alpha' \phi_{n+1,old} + (1-\alpha')\phi_n, \quad 0 \leq \alpha' \leq 1. \tag{5.25}$$

Relaxation coefficients are denoted as α and α'. Again, the updated $\phi_{n+1,new}$ is used in Laplace-Beltrami equation to update the hypersurface function to $S_{n+2,old}$.

The other approach is to refresh the electrostatic potential at a lower frequency than that for updating the surface function. Basically, after a number of iterations (in our tests, 10 to 100 steps) of the generalized Laplace Beltrami equation, the electrostatic potential is then updated. By adaptively changing the number of iterations, one can increase the computational efficiency especially when the change in the surface function during each iteration step is very small.

The coupled system of Laplace-Beltrami equation and PB equation is highly nonlinear. To our best knowledge, there is no rigorous mathematical proof of the existence and uniqueness of the solution. In order to validate our model and algorithm, we evaluate the solvation free energy and compare it with the experimental results [34, 35]. We also check the volume and area of the protein structure calculated from the model during the iteration, and ensure that the convergence to the steady state is observed [35, 187]. In our algorithm, the solvation free energy is often used as an indication of the steady state. Its value can be obtained from minimizing the nonpolar part, polar part and homogeneous energy part as follows

$$\begin{aligned}\Delta G &= G_{nonpolar} + G_{polar} - G_{homogeneous} \\ &= \int_{\Omega} \left\{ \gamma|\nabla S| + pS + (1-S)\sum_{\alpha} \rho_{\alpha} U_{\alpha} \right\} d\mathbf{r} + \frac{1}{2}\sum_{i=1}^{N} Q(x_i)(\phi(x_i) - \phi_0(x_i)),\end{aligned}$$

where the nonpolar part is from Eq. (4.3), and $\phi_0(r_i)$ is the electrostatic potential for the homogeneous environment condition at ith atomic position. Figure 5.3 gives a comparison of electrostatic surface potentials on the molecular surface and the surface obtained from a generalized Laplace-Beltrami flow.

5.2.4 Computational Aspects of Geometric Features

We have introduced the procedure building the characteristic function representing the surface based on PDB data. For EMDB data, the surface is extracted from the volumetric data after noise reduction. Therefore, the protein surfaces from these two data banks are all in the Cartesian repre-

sentation. To evaluate the surface properties, Cartesian representation based algorithms are needed. In this section, the computational methods for the calculating basic geometric features are presented and their potential applications are discussed.

5.2.4.1 Surface Area and Volume Calculation

Based on PDB data and volumetric data from EMDB, the biomolecular surface can be represented by using characteristic functions in two ways: one is the sharp interface by extracting certain iso-values; and the other is the smeared interface that varies in a certain iso-value range. The smeared interface (smooth surface function) is more physical as the radius of the atom is indeed obtained by the probability measure of the electron cloud around the atomic nucleus. Mathematically, the sharp interface is simpler and straightforward.

In the Cartesian representation, the area of a sharp surface can be evaluated as [75, 35]

$$\text{Area} = \sum_{o \in R} \left(\frac{|n_{o,x}|}{h} + \frac{|n_{o,y}|}{h} + \frac{|n_{o,z}|}{h} \right) h^3, \tag{5.26}$$

where R is the set of intersection points located inside the protein domain. $\mathrm{n}_o = (n_{o,x}, n_{o,y}, n_{o,z})$ are the normal for the intersection point. As the surface information is in the Cartesian representation, interpolation is used to evaluate the interface coordinates and norms, see Section 5.2.3.1 for details.

In a smeared surface representation, the mean surface area and the related coarea formula are defined in Eq. (4.1)

$$\text{Area} = \int_0^1 \int_{S^{-1}(c) \bigcap \Omega} d\sigma dc = \int_\Omega |\nabla S(\mathbf{r})| d\mathbf{r}, \quad \mathbf{r} \in \mathbb{R}^3, \tag{5.27}$$

where the surface integration is converted to a volume integration, which is easier to evaluate in the Cartesian representation.

We can obtain a similar expression for the volume calculation. When the protein surface is defined as a sharp interface, a simple summation is used [35],

$$\text{Vol} = \sum_{(i,j,k) \in \Omega} \widetilde{\chi}(i,j,k) h^3, \tag{5.28}$$

Table 5.2: Test of the convergence of the proposed method for the surface area of a sharp interface in the Cartesian representation.

$N_x \times N_y$	Error	Order
21×21	3.391	
41×41	9.390e-1	1.85
81×81	2.026e-1	2.21
161×161	5.498e-2	1.88

where $\widetilde{\chi}(i,j,k)$ is a characteristic function with value 1 inside the protein domain and value 0 for the other. For a smooth surface function, the volume is computed as

$$\text{Vol} = \int_{\Omega} S(\mathbf{r})dr = \sum_{(i,j,k)\in\Omega} S(x_i,y_j,z_k)h^3, \tag{5.29}$$

where $S \in [0,1]$ is the surface characteristic function.

The scheme for computing sharp surface area in the Cartesian representation is tested with an analytical example. We use a sphere with the analytical expression of $x^2+y^2+z^2=4$ in the domain $[-5,5]\times[-5,5]\times[-5,5]$. The second order central finite difference scheme is used in our computation. The error is defined as the absolute value of the difference between the analytical surface area and the calculated surface area. The result is presented in Table 5.2. It is seen that second order accuracy is attained.

5.2.4.2 Curvature Evaluation

The evaluation of curvature properties from iso-surface embedded volumetric data has been thoroughly studied in geometric modeling. There are a variety of elegant methods in the literature. Essentially, the first and second fundamental forms in the differential geometry are involved in the definition and evaluation of the curvatures. We give a brief introduction of the mathematical background [162, 10].

The surface of interest can be extracted from a level set with iso-value S_0, i.e., $S(x,y,z)=S_0$. We assume S to be non-degenerate, i.e., the norm of its gradient is non-zero when it is equal to S_0. Without loss of generality, we further assume that its projection onto z is non-zero. According

to the implicit function theorem, locally, there exists a function $z = f(x,y)$, which parameterize the surface as $\mathbf{S}(x,y) = (x,y,f(x,y))$. One has the relation $S(x,y,f(x,y)) = S_0$. The differentiation with respect to x and y produces two more equations

$$S_x(x,y,f(x,y)) + S_z(x,y,f(x,y))f_x(x,y) = 0, \tag{5.30}$$

$$S_y(x,y,f(x,y)) + S_z(x,y,f(x,y))f_y(x,y) = 0. \tag{5.31}$$

Thus $f_x(x,y)$ and $f_y(x,y)$ can be expressed as:

$$f_x(x,y) = -\frac{S_x(x,y,z)}{S_z(x,y,z)}, \tag{5.32}$$

$$f_y(x,y) = -\frac{S_y(x,y,z)}{S_z(x,y,z)}. \tag{5.33}$$

We define $E(x,y,z), F(x,y,z), G(x,y,z), L(x,y,z), M(x,y,z)$ and $N(x,y,z)$ below to be the coefficients in the first and second fundamental forms. For simplicity, we omit parameter parts. Their values for surface $\mathbf{S} = (x,y,f)$ can be given as

$$E = \langle \mathbf{S}_x, \mathbf{S}_x \rangle = 1 + f_x^2 = 1 + \frac{S_x^2}{S_z^2}; \tag{5.34}$$

$$F = \langle \mathbf{S}_x, \mathbf{S}_y \rangle = f_x f_y = \frac{S_x S_y}{S_z^2}; \tag{5.35}$$

$$G = \langle \mathbf{S}_y, \mathbf{S}_y \rangle = 1 + f_y^2 = 1 + \frac{S_y^2}{S_z^2}; \tag{5.36}$$

$$L = \langle \mathbf{S}_{xx}, \mathbf{n} \rangle = \frac{2S_x S_z S_{xz} - S_x^2 S_{zz} - S_z^2 S_{xx}}{g^{\frac{1}{2}} S_z^2}; \tag{5.37}$$

$$M = \langle \mathbf{S}_{xy}, \mathbf{n} \rangle = \frac{S_x S_z S_{yz} + S_y S_z S_{xz} - S_x S_y S_{zz} - S_z^2 S_{xy}}{g^{\frac{1}{2}} S_z^2}; \tag{5.38}$$

$$N = \langle \mathbf{S}_{yy}, \mathbf{n} \rangle = \frac{2S_y S_z S_{yz} - S_y^2 S_{zz} - S_z^2 S_{yy}}{g^{\frac{1}{2}} S_z^2}, \tag{5.39}$$

where $g = S_x^2 + S_y^2 + S_z^2$ and the normal direction $\mathbf{n} = \frac{(S_x, S_y, S_z)}{g^{\frac{1}{2}}}$. As the Gaussian curvature can be represented as the ratio of the determinants of the second and first fundamental forms, it can be

given by

$$K = \frac{2S_xS_yS_{xz}S_{yz} + 2S_xS_zS_{xy}S_{yz} + 2S_yS_zS_{xy}S_{xz}}{g^2} - \frac{2S_xS_zS_{xz}S_{yy} + 2S_yS_zS_{xx}S_{yz} + 2S_xS_yS_{xy}S_{zz}}{g^2} + \frac{S_z^2S_{xx}S_{yy} + S_x^2S_{yy}S_{zz} + S_yS_{xx}S_{zz}}{g^2} - \frac{S_x^2S_{yz}^2 + S_y^2S_{xz}^2 + S_z^2S_{xy}^2}{g^2}. \tag{5.40}$$

The mean curvature is the average second derivative with respect to the normal direction,

$$H = \frac{2S_xS_yS_{xz} + 2S_xS_zS_{xz} + 2S_yS_zS_{yz} - (S_y^2 + S_z^2)S_{xx} - (S_x^2 + S_z^2)S_{yy} - (S_x^2 + S_y^2)S_{zz}}{2g^{\frac{3}{2}}}. \tag{5.41}$$

An alternative algorithm for the curvature extraction from volumetric data is the Hessian matrix method [99]. For volumetric data $S(x,y,z)$, we define the surface gradient $\mathbf{g}$ and surface norm $\mathbf{n}$.

$$\mathbf{g} = \nabla S = (S_x, S_y, S_z); \tag{5.42}$$

$$\mathbf{n} = -\frac{\mathbf{g}}{|\mathbf{g}|}. \tag{5.43}$$

The Hessian matrix, $\mathbf{H}$, is given by

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 S}{\partial^2 x} & \frac{\partial^2 S}{\partial x \partial y} & \frac{\partial^2 S}{\partial x \partial y} \\ \frac{\partial^2 S}{\partial x \partial y} & \frac{\partial^2 S}{\partial^2 y} & \frac{\partial^2 S}{\partial y \partial z} \\ \frac{\partial^2 S}{\partial x \partial z} & \frac{\partial^2 S}{\partial y \partial z} & \frac{\partial^2 S}{\partial^2 z} \end{bmatrix}. \tag{5.44}$$

The two principal curvatures can be evaluated by the following procedure.

1. Calculate matrix $\mathbf{P} = \mathbf{I} - \mathbf{nn^T}$, here $\mathbf{I}$ is the identity matrix and $\mathbf{T}$ denotes the transpose;

2. Evaluate matrix $\mathbf{G} = \mathbf{I} - \frac{\mathbf{PHP}}{|\mathbf{g}|}$,

$$\mathbf{G} = (g_{ij})_{(i,j=1,3)} \tag{5.45}$$

3. Calculate the trace t and Frobenius norm f of matrix $\mathbf{G}$;

$$t = g_{11} + g_{22} + g_{33}; \tag{5.46}$$

$$f = \|\mathbf{G}\| = \sqrt{\sum_i \sum_j g_{ij}^2}; \tag{5.47}$$

$$\kappa_1 = \frac{t + \sqrt{2f^2 - t^2}}{2}; \tag{5.48}$$

$$\kappa_2 = \frac{t - \sqrt{2f^2 - t^2}}{2}. \tag{5.49}$$

When the two principal curvatures are available, the Gaussian curvature K and mean curvature H can be easily calculated,

$$K = \kappa_1 \kappa_2; \tag{5.50}$$

$$H = \frac{\kappa_1 + \kappa_2}{2}. \tag{5.51}$$

Essentially, the Hessian matrix method generates the same results as the above algorithm derived from the first and second fundamental form.

5.2.4.2.1 Numerical Test for Analytical Cases

We use the second order central difference scheme to do the discretization. Two analytical examples are considered. We denote L_∞ and L_2 the L_∞ error and L_2 error.

Case 1. We set the domain as $[-10,10] \times [-10,10] \times [-10,10]$, and define a surface as

$$Z(x,y) = \frac{(x^2 - y^2)(x^3 - y^3)}{2000}. \tag{5.52}$$

Basically, the volumetric data $f(x,y,z)$ are defined as $f(x,y,z) = z - Z(x,y)$. Therefore, the analytical surface $Z(x,y) = \frac{(x^2-y^2)(x^3-y^3)}{2000}$ can be extracted by setting $f(x,y,z) = 0$. The analytical expressions for Gaussian curvature and mean curvature can be calculated,

$$K = \frac{z_{xx} z_{yy} - z_{xy}^2}{(1 + z_x^2 + z_y^2)^2}, \tag{5.53}$$

$$H = \frac{(1 + z_x^2) z_{yy} - 2 z_x z_y z_{xy} + (1 + z_y^2) z_{xx}}{2(1 + z_x^2 + z_y^2)^{\frac{3}{2}}}. \tag{5.54}$$

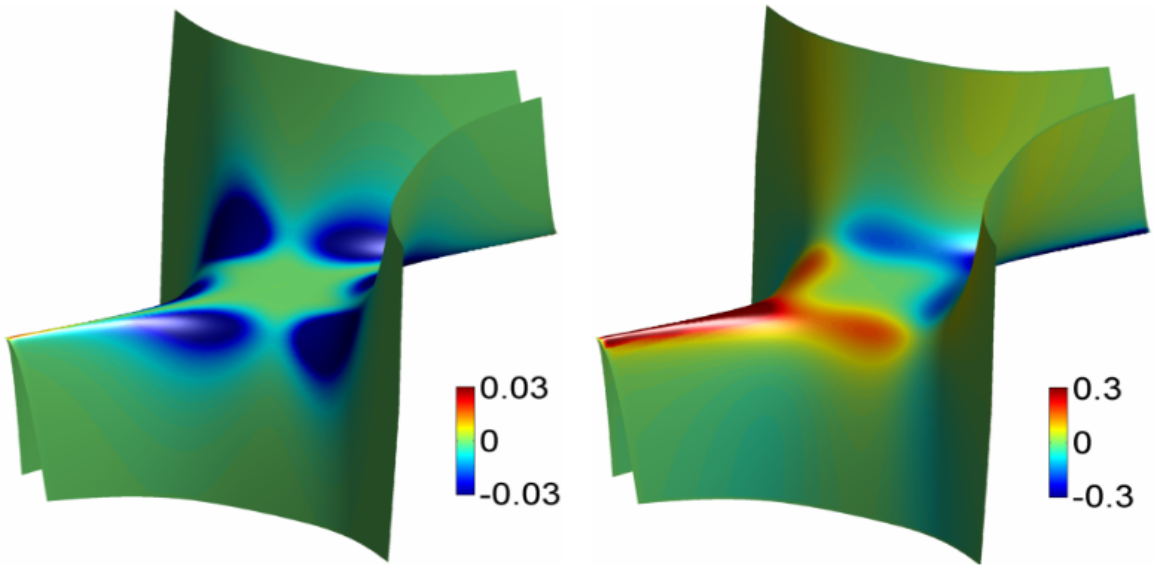

Figure 5.4: Computational results of Gaussian curvature and mean curvature for test Case 1.

Table 5.3: Numerical errors and convergence orders for calculating Gaussian curvature (Case 1).

$n_x \times n_y$	L_∞	Order	L_2	Order
21 × 21	1.483e-1		1.543e-2	
41 × 41	7.049e-2	1.07	4.280e-3	1.85
81 × 81	2.028e-2	1.80	8.494e-4	2.33
161 × 161	5.348e-3	1.92	1.816e-4	2.23

Table 5.4: Numerical errors and convergence orders for calculating mean curvature (Case 1) .

$n_x \times n_y$	L_∞	Order	L_2	Order
21 × 21	4.498e-1		3.735e-2	
41 × 41	4.057e-2	3.47	2.667e-3	3.81
81 × 81	2.231e-3	4.18	5.262e-4	2.34
161 × 161	6.893e-4	1.69	1.343e-4	1.97

The numerical results are demonstrated in Fig. 5.4. Tables 5.3 and 5.4 give the error and associated convergence order. As we use the second order finite difference scheme to evaluate the derivatives, the second order accuracy is obtained. We also tested the Hessian matrix method, it generates the same results.

Case 2. In this case the domain is set as $[-10,10] \times [-10,10] \times [-10,10]$, and a surface is

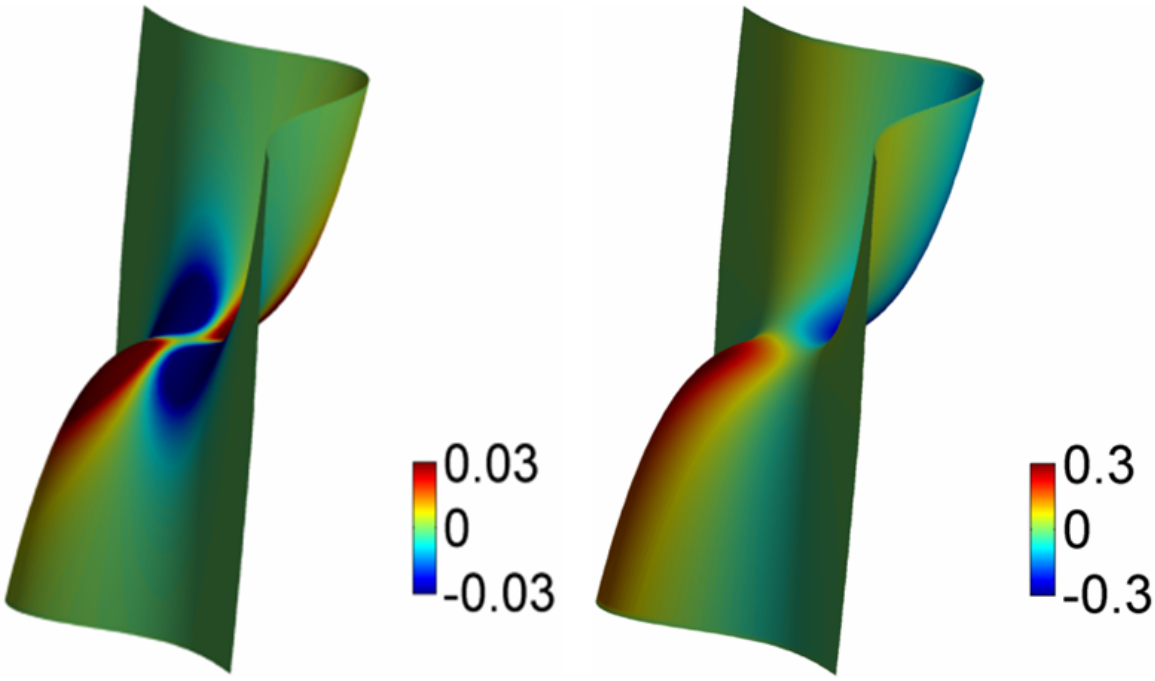

Figure 5.5: Computational results of Gaussian curvature and mean curvature for test Case 2.

Table 5.5: Numerical errors and convergence orders for calculating Gaussian curvature (Case 2).

$n_x \times n_y$	L_∞	Order	L_2	Order
11 × 11	2.682e-2		6.295e-3	
21 × 21	7.268e-3	1.88	1.420e-3	2.15
41 × 41	1.846e-3	1.98	3.528e-4	2.01
81 × 81	4.927e-4	1.91	8.751e-5	2.01

defined as

$$Z(x,y) = \frac{(x^3 - y^3)}{30}. \tag{5.55}$$

The volumetric data $f(x,y,z)$ are defined as $f(x,y,z) = z - Z(x,y)$. Therefore, the analytical surface can be extracted by setting $f(x,y,z) = 0$. We can calculate the analytical solution for the Gaussian curvature and the mean curvature using Eqs. 5.53 and 5.54. Fig. 5.5 demonstrates the numerical results. The error and associated convergence order are listed in Tables 5.5 and 5.6. The Hessian matrix method gives the same results and both methods achieve the second order accuracy.

Table 5.6: Numerical errors and convergence orders for calculating mean curvature (Case 2).

$n_x \times n_y$	L_∞	Order	L_2	Order
11×11	4.451e-2		1.009e-2	
21×21	1.146e-2	1.96	2.415e-3	2.06
41×41	2.923e-3	1.97	5.942e-4	2.02
81×81	7.604e-4	1.94	1.470e-4	2.02

5.2.4.2.2 Numerical Test for Protein Data

Having validated the two methods for curvature evaluation, we apply them to calculation of the structural features of proteins. Three protein structures are considered, two are from EMDB, i.e., EMD5273 and EMD5020, and the other one is from PDB with ID:1PPL. Figures 5.6, 5.7 and 5.8 demonstrate our results. All the protein surfaces generated in this section are extracted with the isovalue C=0.5. The data size for 1PPL is $146 \times 117 \times 97$. EMD5273 and EMD5020 have the same data size of $100 \times 100 \times 100$. As curvature evaluation algorithms involve only simple interpolation, the computation cost is very small. On a PC with Pentium 4 CPU 3.60GHz and 1.00 GB RAM, the computation times are about 4.2, 2.1 and 2.2 second for proteins EMD5273, EMD5020 and 1PPL, respectively.

These curvatures describe the concave and convex properties of the protein surface, see Section 2.1.2. It is well known that in drug design and protein-protein interaction, the surface geometry is of significant importance [38]. Usually, the geometry of the drug should match to a concave region of the protein just like the key and lock relation. The same applies when two proteins interact with each other, or when a substrate binds to the active site of an enzyme. The quantitative measurement of curvatures has a great potential for further modeling and analysis of the geometric impact on biomolecular interactions.

Gaussian curvature characterizes the topological property of a surface. When integrated over the surface, Gaussian curvature gives rise to the information of the genus number, which is, loosely speaking, the number of "holes" in the biomolecule. The genus number can be applied to systems like ion channel proteins, whose open state and close state have different genus numbers. From

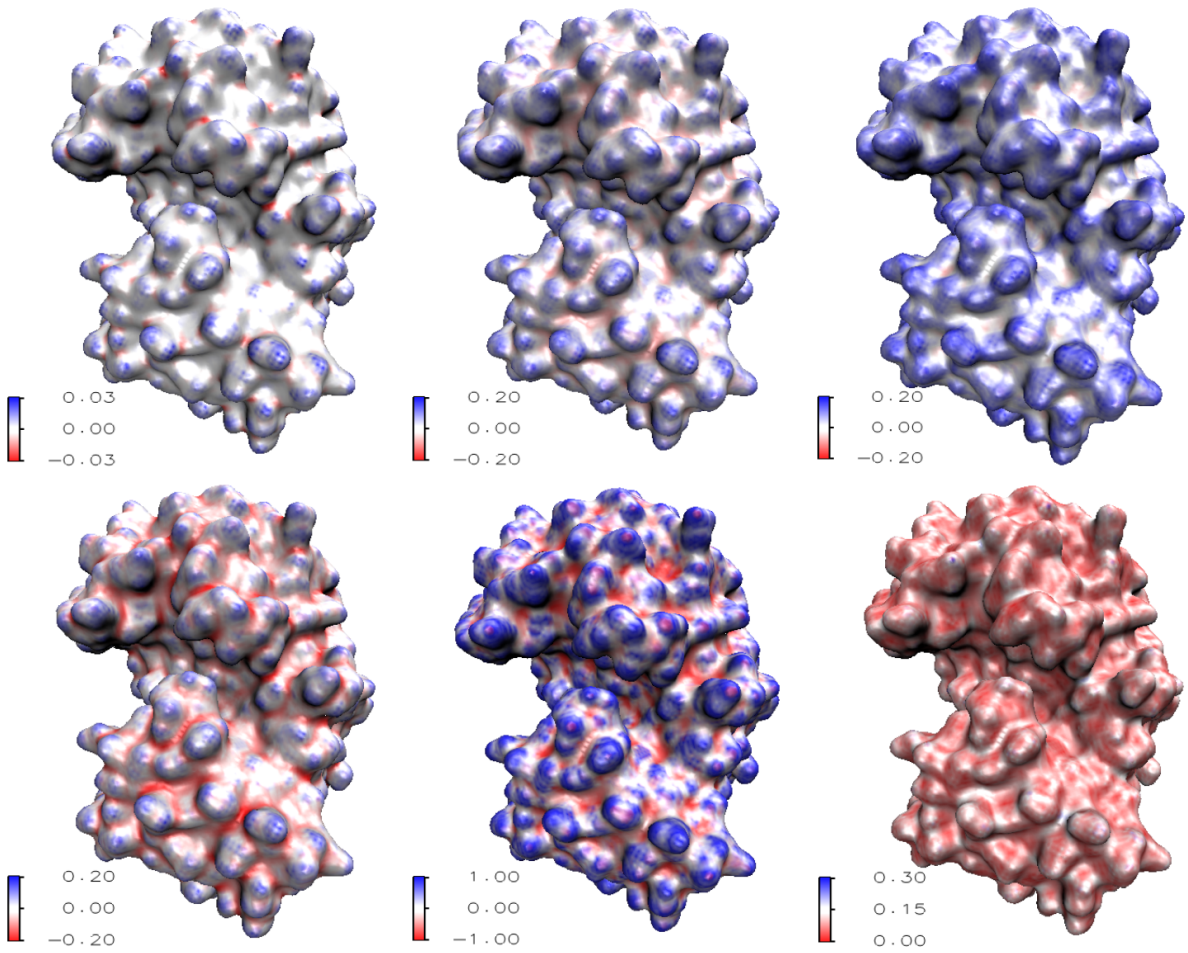

Figure 5.6: Curvature analysis of Protein 1PPL. From top left to bottom right: Gaussian curvature, mean curvature, maximum curvature, minimum curvature, shape index, and curvedness.

the minimum curvature and shape index, we can obtain a rather clear picture of concave regions. Actually, the concaveness can be quantitatively characterized by the values of minimum curvature and shape index. Similarly, the convexity can be parameterized by the maximum curvature and shape index. The curvedness provides the information about the amplitude of the curvature, e.g., a large value usually indicates a sharp edge and/or corner.

The traditional MS suffers from geometric singularities, for which curvatures are undefined. Computationally, near geometric singularities, curvatures tend to have much dramatic local variations and the accuracy of computational results is reduced. In our MMS and surface generated from geometric and potential driven geometric flows, the geometric singularities are removed, and

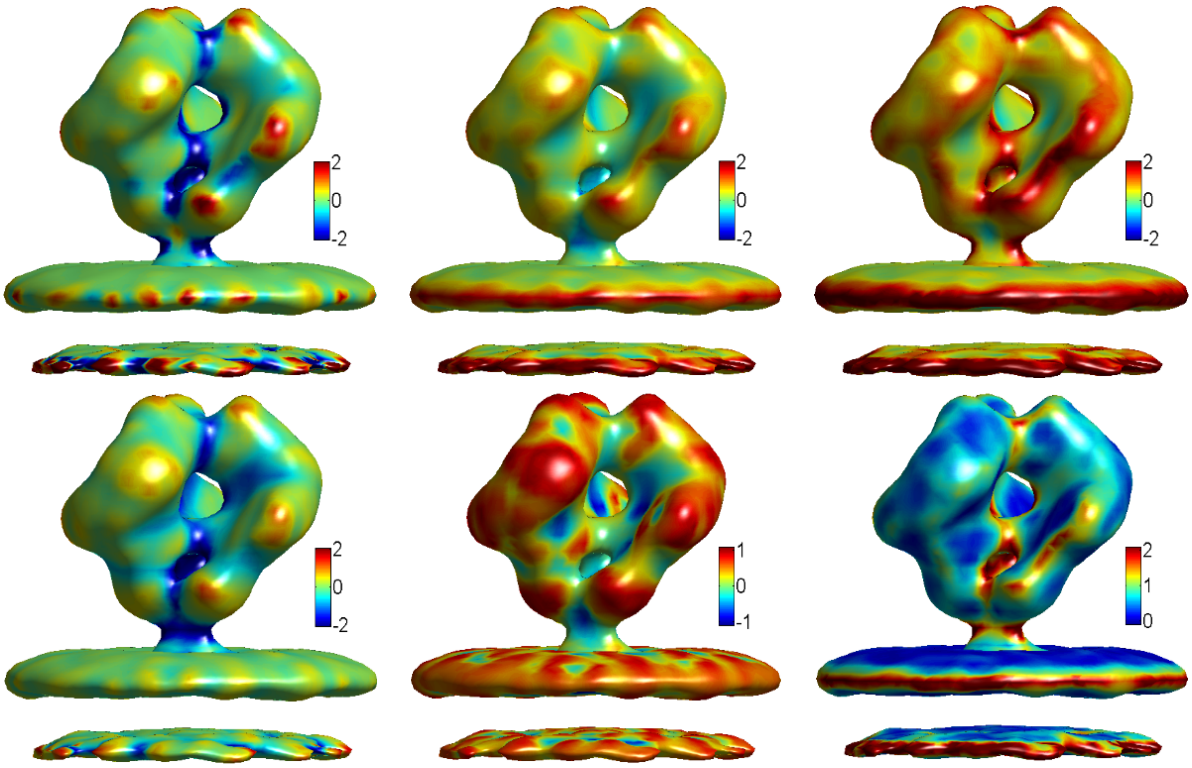

Figure 5.7: EMD5273 curvature properties. From top left to bottom right: Gaussian curvature, mean curvature, maximum curvature, minimum curvature, shape index, and curvedness.

the surface is smooth with less local curvature fluctuations. Further, a multiresolution model is proposed in our recent work [64]. Obtained with an adjustable parameter, a family of multiresolution surfaces can be designed to reduce local curvature variations. Consequently, concave regions and convex regions reflect the molecular morphology, instead of local atomic characteristics. In differential geometry based multiscale multiresolution models, the electrostatic potential is also coupled with the molecular surface generation. With polar and nonpolar areas defined by electrostatic potential, and concave and convex regions evaluated by the above curvature schemes, our approaches have a great potential for the prediction of protein active sites and/or binding sites.

5.2.5 Polarized Curvature and Binding Site Prediction

Based on the above curvature analysis and electrostatic characterization, it is clear that a potential protein binding site should be both electrostatically favorable and geometrically favorable. To

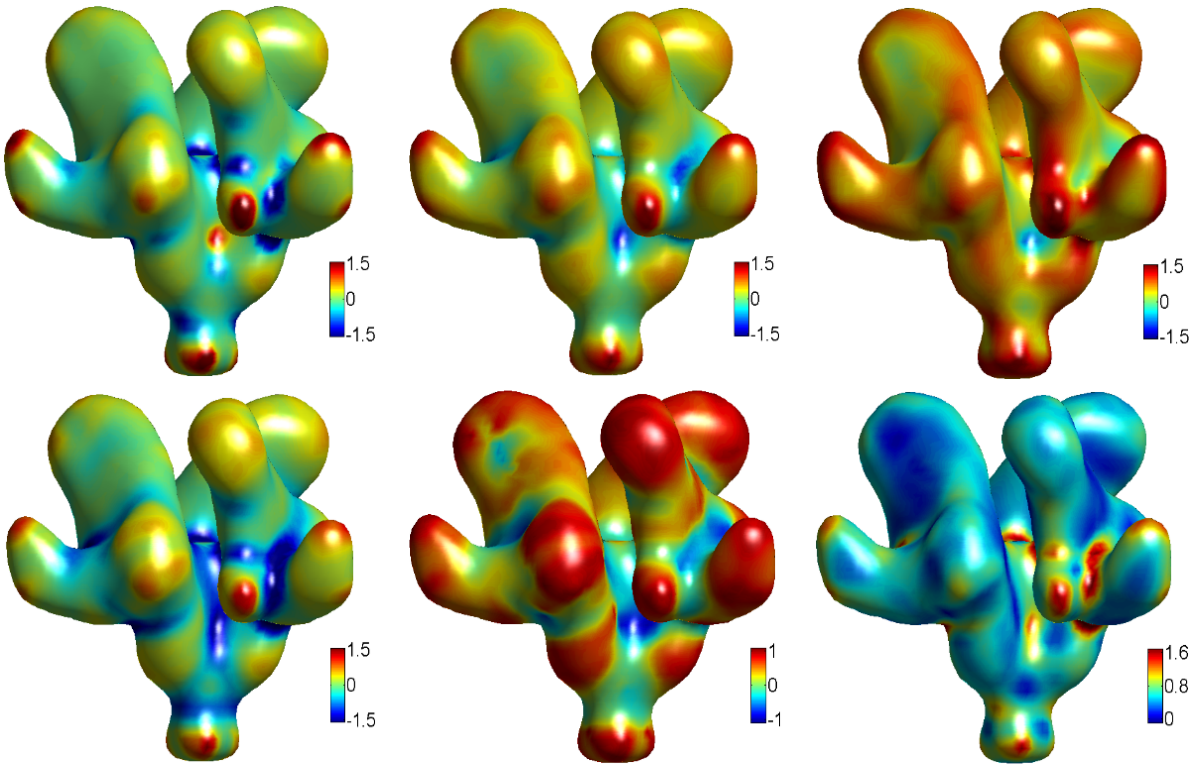

Figure 5.8: EMD5020 curvature properties. From top left to bottom right: Gaussian curvature, mean curvature, maximum curvature, minimum curvature, shape index, and curvedness.

combine these compatibilities, we propose polarized curvatures as the products of electrostatic potentials and curvatures. Specifically, the maximal curvature κ_1 and minimal curvature κ_2 are employed to construct their products with positive electrostatic surface potential Φ^+ and negative electrostatic surface potential Φ^-. Large amplitudes of these products indicate four different potential binding sites as summarized in Table 5.7. For example, a large amplitude of $\Phi^+ \times \kappa_1$ on a certain region of a protein surface indicates a potential binding site for a negatively charged protein or virus, while a large amplitude of $\Phi^+ \times \kappa_2$ indicates a potential binding site for a negatively charged small ligand. Similar behavior can be stated for the products of $\Phi^- \times \kappa_2$ and $\Phi^- \times \kappa_1$.

Figure 5.9 demonstrates the effectiveness of our proposed polarized curvature analysis. The top row illustrates the electrostatic surface maps and (small ligand) binding sites of four proteins. The bottom row displays the predictions of polarized curvatures ($\Phi \times \kappa_2$). In these cases, the minimal curvature (κ_2) is used to predict the concave regions of protein surfaces for potential binding sites

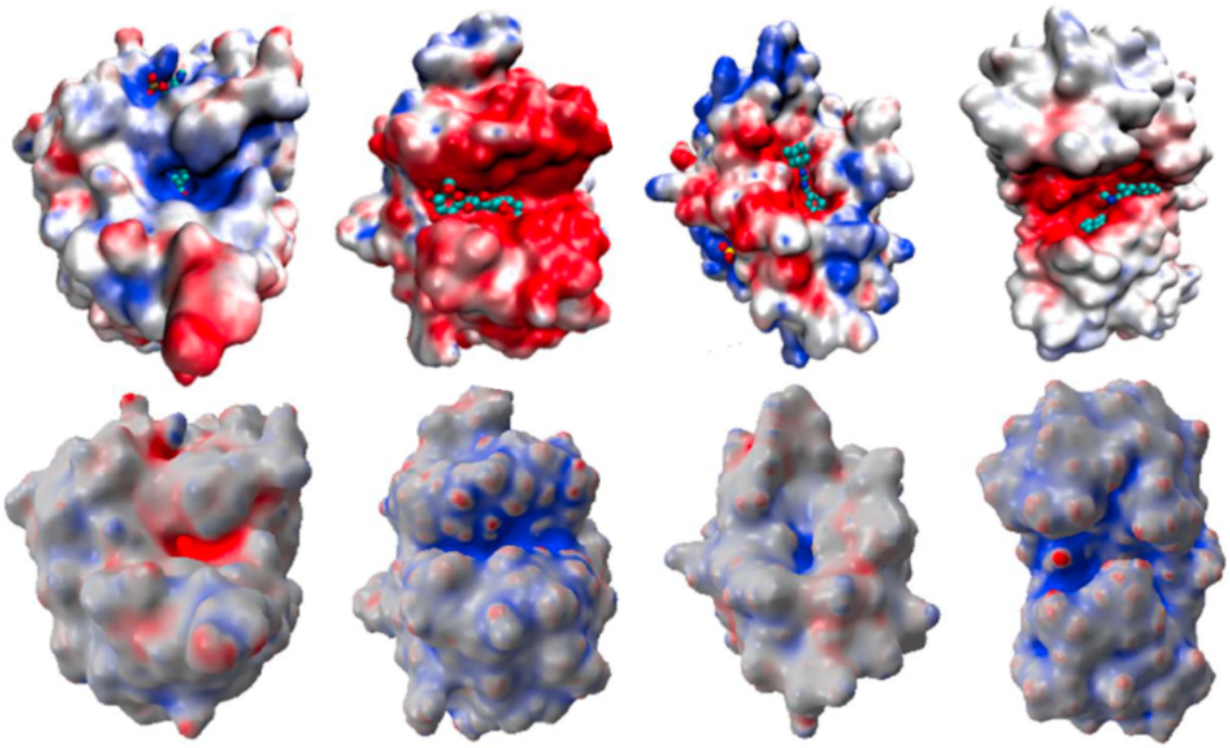

Figure 5.9: Polarized curvature based binding site prediction for four proteins (Left to right:1ADS, 1BYH, 1EJN, 2WEB). Top row: Protein-ligand complexes displayed with electrostatic surface potential; Bottom row: Polarized curvature maps ($\Phi \times \kappa_2$) indicating the binding sites.

Table 5.7: Polarized curvatures as binding indicators of protein surfaces. Maximum curvature (κ_1), minimum curvature (κ_2), positive electrostatic surface potential (Φ^+) and negative electrostatic surface potential (Φ^-) are combined to indicate potential binding sites.

	$\kappa_1 > 0$	$\kappa_2 < 0$
$\Phi^+ > 0$	site for negatively charged protein	site for negatively charged small ligand
$\Phi^- < 0$	site for positively charged protein	site for positively charged small ligand

of small ligands. The protein on the left chart is positively charged at its binding site and the rest of proteins are all negatively charged at their binding sites. The polarized curvatures shown in the bottom row give correct predictions for all binding sites.

In our future work, we will combine the polarized curvature analysis and the binding affinity analysis readily available in our multiscale solvation model [35] for more accurate prediction of protein-ligand binding, protein-DNA specificity and protein-protein interactions. We will make

this approach automatic and robust.

5.3 Summary

Geometric modeling has widespread applications in the visualization, analysis and characterization of macromolecules. For proteins, their structural features are intrinsically associated with their functions and molecular mechanisms. The exploration of the geometric features of a protein molecular surface enhances our understanding of molecular morphology and molecular mechanism, and allows significant applications to drug design and protein-protein interaction. This is particularly true when the geometric modeling is associated with the electrostatic analysis. The work in this chapter offers expository investigation and comprehensive summary of tools, algorithms and methodologies for geometric modeling of macromolecules in the Eulerian formulation, which is advantageous in handling potential topological changes.

Our study is based on two major biomolecular structure sources collected from experiments: the Protein Data bank (PDB) and the Electron Microscopy Data Bank (EMDB). The PDB contains information about structures obtained mainly by using X-ray crystallography and NMR spectroscopy at the atomic level resolution. Whereas, EMDB provides information mainly about multiproteins, organelles, viruses, and subcellular complexes obtained mostly from cryo-Electron Microscopy (cryo-EM) at the molecular level resolution. In this chapter, based on data from the PDB and the EMDB, related geometric modeling methods, software packages and visualization tools are provided and discussed in great detail.

The protein data from the PDB are in atomic resolution, so that crucial information like atom positions, van der Waals radius and partial charges can be obtained either directly or indirectly. Different definitions of the macromolecular surface have been proposed and constructed based on experimental data. However, the resulting surfaces usually suffer from geometric singularities (i.e., tips, cusps and self-intersecting facets) and violate the energy minimization principle, due to the fact that they are just ad hoc divisions of the protein and its surroundings. The minimal molecular surface (MMS) is proposed as a surface that minimizes the surface free energy. This variational

formulation based biomolecular surface fulfills the principle of energy minimization, while producing a smooth surface through Laplace-Beltrami flows obtained from the Euler-Lagrange equation. As the solvation process is of fundamental importance to biomolecular systems, it should be considered in the surface modeling. By adding the solvation energy, which is composed of nonpolar and polar parts, into the total free energy functional and by using the Euler-Lagrange equation, the geometric and potential driven Laplace-Beltrami flow is formulated. Essentially, the external potential term incorporates various solvation effects, except the surface tension. Further, in different types of biomolecular systems, other related effects, such as chemical potential and fluid flow, are accounted in external potential terms as well. In this paper, we explore all the surface generation related geometric aspects, including surface modeling, computational methods, algorithms and techniques.

The data from the EMDB, in contrast, is in a volumetric format and usually without detailed atomic information. These data often have a poor signal to noise ratio (SNR) and a noise reduction process is required. High order geometric PDEs can suppress the high-frequency components efficiently. In this paper, for the first time, the high order geometric PDEs are applied to the EMD noise removal. With the suitable PDE order and iteration time, the noise is drastically reduced, while image features are preserved.

Curvature properties indicate the concave or convex regions, which are likely to be the potential binding sites or active sites. Within the framework of the Cartesian representation, we tested second order computational algorithms for curvature evaluation. Six different curvature descriptors, including Gaussian curvature, mean curvature, maximum curvature, minimum curvature, shape index, and curvedness, are employed for the first time to the two types of protein surfaces, variational surfaces generated from PDB data and surfaces extracted from denoised EMDB data. An interesting feature of our work is that the curvature analysis for surfaces generated from our variational model is paired with the electrostatic analysis resulted from the same model. Such a feature enables us to introduce polarized curvatures for the screen of protein-ligand binding and protein-protein interaction sites. We demonstrate that the proposed polarized curvatures give rise

to reasonable predictions of protein-ligand binding sites.

Chapter 6

TOPOLOGICAL FEATURE DETECTION

6.1 Introduction

Topological features reveal the global structures of the shapes. On closed surfaces in 3D, the topological features are represented through certain equivalent classes of loops. In this chapter, we propose a practical definition of topologically and geometrically useful loops using the theory of persistent homology on volumes to address the issues discussed in Section 1.4. We also provide an efficient algorithm that produces all of these loops fully automatically. Some of them are indeed topologically trivial in surface topology, but we will make their 3D topological relevance precise through the definitions in Sec. 6.2.

The remainder of the chapter is organized as follows. In Sec. 6.2, we provide the necessary mathematical definitions used in persistent homology before giving our definition of choking loops. We describe the procedure of detecting such topological structures in Sec. 6.3.1, and provide a method to identify, within an equivalent homotopy class, a discrete approximation of the choking loop on the surface in Sec. 6.3.2. We then discuss the application in molecule stability analysis in Sec. 6.4. We show results in Sec. 6.5, and conclude in Sec. 6.6.

6.2 Mathematical Background

In the first half of this section, we briefly introduce the concept of homology in topology, and the concept of persistent homology, which provides a way to geometrically measure the topological features. These concepts are crucial to the definitions that we give in the second half of this section, which can indeed be seen as a specifically designed, yet straightforward special case of persistent homology.

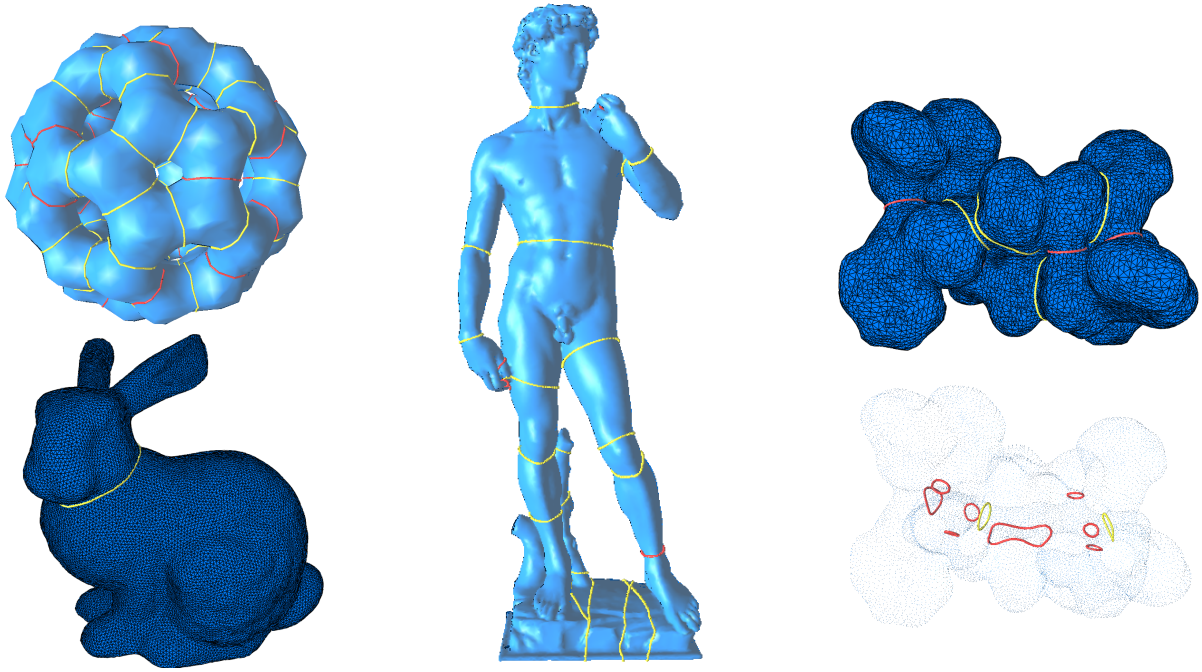

Figure 6.1: Upper left: C_{60} "buckyball" is of genus-31, but our algorithm finds 59 more "choking" loops (yellow) than the usual 31 homology generators (red). Lower left: Although the bunny model has a trivial topology, we still find topological features corresponding to the narrowing of the neck. Middle: This David statue model is of genus-5, with 3 handles near the right hand, 1 formed by the legs and pedestal, and the last one by the left arm; our approach extracts other topologically-relevant handles, e.g., around the waist or the neck. Right: Focusing only on the shortest 1-homology generators of a 1mag protein (top) fails to identify important ion channel loops (bottom, yellow) that our algorithm easily extracts from the surface description.

6.2.1 Preliminaries

For a more formal and detailed treatment of persistent homology theory, please refer to [56, 55]. The basic concepts and theory in persistent homology has been discussed in details in Section 2.3.

As in [49], we define two types of loops, given a closed surface M separating the 3D space into an inside I (with finite volume) and an outside O (in practice, the volume between the surface and a bounding box).

Definition 1 *A loop in $H_1(M)$ but not $H_1(M \cup I)$ is a handle.*

Definition 2 *A loop in $H_1(M)$ but not $H_1(M \cup O)$ is a tunnel.*

Intuitively speaking, a handle loop can shrink through the inside of the object into a point and a tunnel loop can shrink through the outside of the object into a point.

Homology groups are topological invariants, and as such, they are not influenced by the lengths defined on the surface or the embedding in 3D space. To enable measurements, persistent homology can be used to give a notion of persistence for each homology generator by measuring its life span in a filtration, which is a nested sequence of subcomplexes of K.

$$\emptyset = K_{-1} \subset K_0 \subset ... \subset K_n = K \tag{6.1}$$

The inclusion map from K_i to K_j ($i \leq j$) induces a mapping from homology groups of earlier subcomplexes to those of later subcomplexes. If we assume that we build the filtration by adding one simplex at a time, each p-simplex will either create a nontrivial p-cycle in the homology of the new subcomplex, or eliminate a nontrivial $p-1$-cycle in the homology of the previous subcomplex. This can be seen as a consequence of the fact that it increases the Euler characteristic ($\chi = \#V - \#E + \#F - \#T$, the alternating sum of numbers of simplices of different dimensions) by $(-1)^p$, and the fact that $\chi = dim(H_0) - dim(H_1) + dim(H_2) - dim(H_3)$, the alternating sum of dimensions of homology groups of different orders. Using positive simplices to represent the homology generators (nontrivial cycles), we can mark the birth time of each homology generator by the order i of the subcomplex K_i. We can pair each negative simplex with the positive simplex representing the nontrivial cycle that it kills, and mark the death time of that nontrivial cycle, j of the subcomplex K_j. The difference between the two times $j-i$ is the persistence of that nontrivial cycle.

In our algorithm, we use *lower-star filtration*, defined by the nested sequence of complexes with simplices added in ascending order of the Morse function d (a function without degenerate critical points, discretely, it can be a function that takes different values at different vertices after symbolic perturbation). One important fact of the pairing algorithm in [56] is that we always kill the *youngest* cycle among all the cycles that could be killed by a negative simplex (the elder's rule); we thus avoid converting an important persistent topological structure into a sequence of short-lived nontrivial cycles. In the case of lower-star filtration, this rule can also be interpreted

as a way to measure the smallest amount of perturbation to the Morse function that is necessary to cancel out a topological structure by its paired simplex. Thus, the persistence of a topological feature is measured by the difference in d, to reduce the dependence on the discretization of the objects.

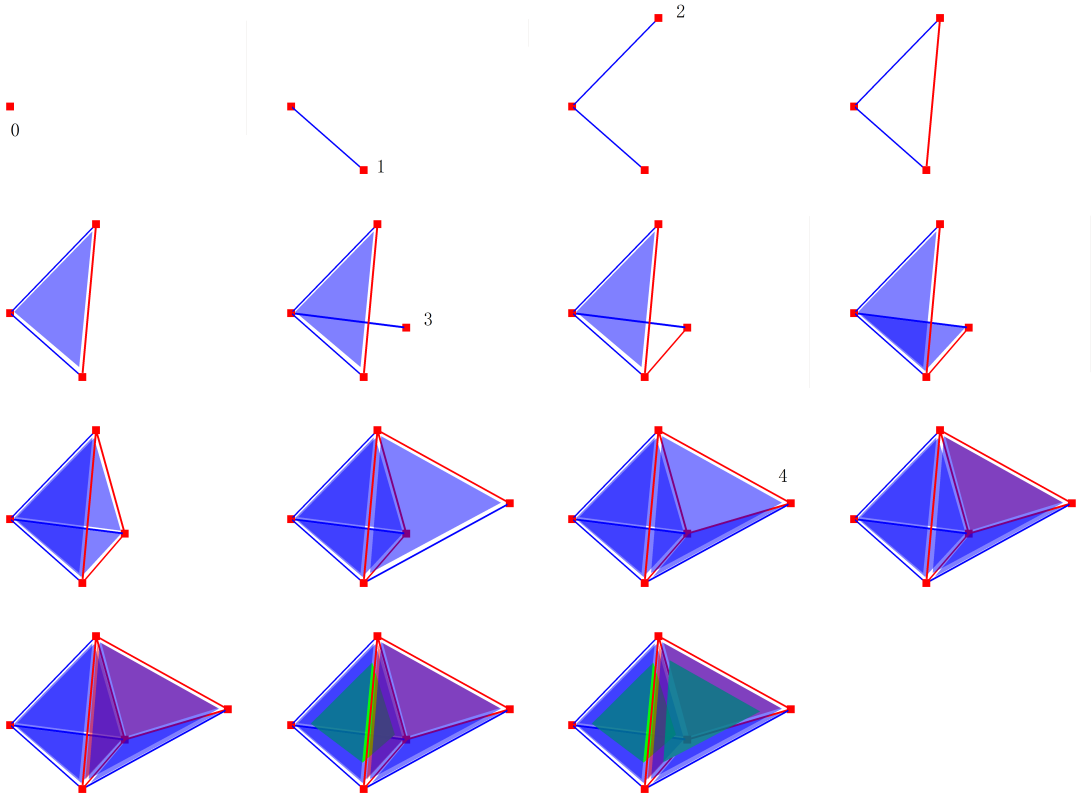

Figure 6.2: Here we show a 3D simplicial complex (tet mesh) containing two tets. We show the process of building persistent homology using the pairing algorithm. The red simplices are positive, and blue ones negative. We use transparent rendering. Thus, dark blue will show when a negative face is covered by another negative face, and purple faces are positive faces covered by negative faces. To make the negative tets visible, we render both in green.

Here we show a complex with two tetrahedra as an example for the pairing algorithm in persistent homology. We have five vertices in this case. The simplicial complex K consists of all the simplices contained in $\{0,1,2,3\}$ and $\{4,1,2,3\}$. The filtration is constructed as follows:

1. positive $\{0\}$;

2. positive $\{1\}$ (two connected components now);

3. negative $\{0,1\}$ killing the younger connected component $\{1\}$;

4. positive $\{2\}$, and negative $\{0,2\}$;

5. positive edge $\{1,2\}$ creating a nontrivial loop, subsequently killed by negative face $\{0,1,2\}$;

6. a few more pairs: $(\{3\},\{0,3\})$, $(\{1,3\},\{0,1,3\})$, $(\{2,3\},\{0,2,3\})$, $(\{4\},\{1,4\})$, $(\{2,4\},\{1,2,4\})$, $(\{3,4\},\{1,3,4\})$;

7. positive face $\{2,3,4\}$ on the surface creating one piece of void inside;

8. positive face $\{1,2,3\}$ cutting the void into two pieces;

9. negative tetrahedron $\{0,1,2,3\}$ killing the younger membrane $\{1,2,3\}$;

10. negative tetrahedron $\{1,2,3,4\}$ killing $\{2,3,4\}$ and filling the inside space.

In this example, the only homology structures with persistence greater than 1 (not immediately killed after birth) are the connected component represented by $\{0\}$, and the closed membrane represented by $\{2,3,4\}$. See [56, 49] if the reader wishes to see example pseudo-code of the pairing algorithm.

6.2.2 Definition of Choking Loops

We observe that a handle can be detected as a narrow passage inside the material side of the surface. As we offset the surface towards the interior, the swollen surface will "choke" off the air passage inside the handle. These would naturally include the g first homology generators of the surface, as well as other similar candidates. In fact these additional choking locations are, formally, second homology generators of the swollen surface, as these membranes now divide the volume enclosed by the surface into more than one connected components. The locations for handle-type 1-homology generators can actually also be seen as where the membranes cut the topologically nontrivial inside volume into a topologically trivial ball-like volume.

More practically, we create a filtration of the tetrahedral mesh of the inside of the surface $M \cup I (= K)$ as follows. First, we assume that a parameter d denoting the distance of each vertex to the surface is stored for each interior vertex. We can use symbolic perturbation to determine the order of vertices at a same distance [55]. We then build a filtration of the surface mesh. Next, we add one interior vertex at a time in ascending order of distance, and add any simplices containing the vertex that can be added without violating the condition of forming a subcomplex. When all vertices within distance d are added, the current subcomplex is the solid object between M and its offset by d toward the interior, which we denote by K_d. In other words, we build the aforementioned lower-star filtration using the distance field to the surface for the inside volume. We now define choke face and its associated choking loop.

Definition 3 *A choke face (*3D choke point*) at a distance d from the surface M with persistence* δ *is a negative face killing a 1-cycle nontrivial in* $H_1(M)$ *but trivial in* K_d*, or a positive face representing a 2-cycle non-trivial in both* K_d *and* $K_{d+\delta}$.

These are locations where an object with a diameter greater than $2d$ will get stuck. If f is a positive face, it separates $K \backslash K_d$ into more connected components. If f is a negative face, $K \backslash K_d$ is cut into a topologically simpler volume, (e.g., cutting a torus into a topological ball). The red triangles and the yellow triangles in Figure 6.3are example positive and negative choke faces, respectively.

We intentionally left out the requirement for persistence δ on the negative faces (corresponding to the original $2g$ nontrivial loops on the surface) : depending on the application they may still be important to the surface topological structure, no matter how non-persistent they are. However, if required, a condition that would place these negative faces on the same footing as the positive faces is easy to formulate: we measure the persistence of the negative faces by the difference between the distance at which they are found and the distance at which the volume inside that piece of the original surface is filled. This persistence means that these handle loops are cutting the inside volume into a topologically simpler volume long before the void is completely filled up, as the

inside passage would not mean much if the inside volume linked by it was about to disappear as a whole. A similar requirement applies to the original tunnels. In this case, we use the difference between the distance to fill up the bounding box representing the outside space within which we put the object and the distance at which the negative face is found. It means that if the persistence is small, i.e. the whole room is about to be filled up before we kill the tunnel of the object in that room, the tunnel would have been almost as wide open as the room to begin with.

Definition 4 *A choking loop associated with a choke face f (first added to the filtration in K_d) is the loop* on the surface *M, formed as the boundary of the smallest membrane B containing f, such that $B \backslash f$ is homotopic to the boundary of f in $K_d \backslash f$.*

Intuitively speaking, we deform f to the membrane B by growing it inside K_d, turning it from what locally separates $K \backslash K_d$ into what locally separates K, the entire volume inside. Boundary of the smallest membrane going through the choke point is not necessarily a geodesic loop, although very close to one in practice. However, this can be a more reasonable requirement in finding the narrowing in the volume, e.g. in medical applications for detecting constrictions of airway or blood vessels, as the area of the membrane limits the capacity of fluid flow roughly speaking.

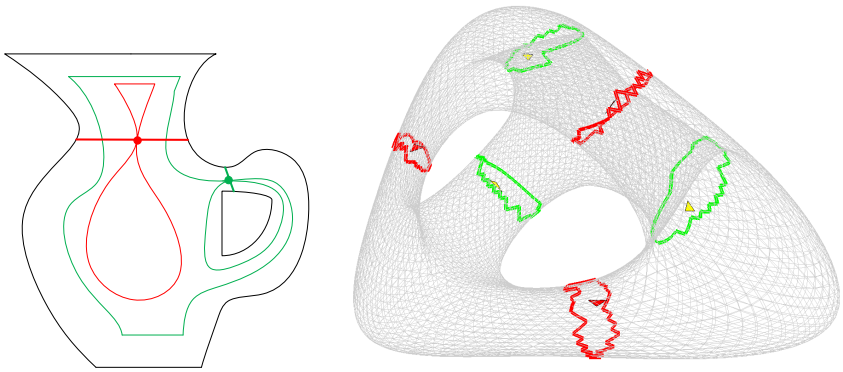

Figure 6.3: Left: 2D illustration, when green offset curve is reached, a handle is detected, and when red offset curve is reached, an additional choke point is detected). Right: Display of 6 3D choke points (3 handles corresponding to the genus-3 in red, and 3 additional handles in yellow) with their associated choking loops. On this model, we show the loops before the postprocessing shortening.

We can also define all tunnel-like nontrivial loops similarly, which can be named "external" choking loops. In this case, we offset the surface along positive normal direction to build the filtration, and again examine its persistent homology. In practice, we create a large bounding box of the surface, and treat the space between the surface and the bounding box as the volume in the above definition. The external choking loops include the g tunnels (which cut the volume outside into a topological ball if we see the 3D space as part of the 3D ball S^3) as well as other membranes, cutting the space outside into pieces, enclosing most of the pieces and leaving only one outside.

In the above definition we used two parameters d and δ, both of which are rather intuitive. d denotes how far we have to offset the surface to create the choking loop. The use of δ avoids creating many duplicate loops that would have been merged when the offset is changed by δ, providing resilience to geometric noise. Thus, our definition can create useful loops even for genus-0 objects, but it will not create cluttered clusters of loops. For high genus models, one might occasionally find a choking loop associated with a positive choke face before finding any handles, as there may be a very narrow passage connecting the bulk of the inside volume to another large piece of volume (roughly speaking, with radius greater than δ) enclosed by a topologically trivial patch of the surface.

6.3 Choking Loop Calculation

Built on persistent homology with the filtration ordered by a distance function, our algorithm requires a volume mesh with sufficient internal vertices to discern the distances at which the choking loops are detected. In contrast to the HanTun algorithm [49], also based on persistent homology but with a volume mesh containing only surface vertices, we employs 2-homology as well as 1-homology to detect the additional choking loops. Furthermore, our distance-based homology gives the loops a geometrically relevant ordering and associated persistence.

6.3.1 Detection of Nontrivial Topology

We now present the procedure used to compute choke faces. Without loss of generality, we only describe handle-like choking loops here. Before we start building the filtration, we first preprocess the surface representation. If we are given a surface triangle mesh, we create a tetrahedralization of the boundary surface using Tetgen [158]. The distance field to the surface can be estimated by a number of different ways. For example, we may run Closest Point Transform [120] to create a distance field on a regular grid of the bounding box of the object, followed by trilinear interpolation of the distance field on the grid for the internal vertices of the tetrahedral mesh. Alternatively, we can run a multi-source Dijkstra's algorithm computing the shortest distance through edges from the surface for each vertex. For implicit surfaces, fast marching can be performed to create the distance field if the level set function is not already a signed distance field, and we perform marching cubes to create a surface mesh, and proceed with the tetrahedralization and trilinear interpolation.

The filtration parameter/time is the distance to the surface. Thus, at time 0, we start the filtration by adding all cells of the boundary surface, e.g. in a breadth-first traversal from a seed vertex. When the whole closed surface is in the filtration, we will have $2g$ positive edges left representing the $2g$ homology generators. Next we add all interior edges with both vertices on the surface into the filtration, followed by interior faces with all three vertices on the boundary in the ascending order of the number of edges inside the volume, and the tets with all vertices on the boundary. We then add the vertex with the smallest d, followed by the simplices in the tetrahedral mesh containing the new vertex, i.e., the edges connecting it to vertices already in the current subcomplex, the faces formed by the vertex and two previous vertices, and the tets formed by the vertex and three previous vertices. We repeat the process, until we reach the distance d_{max}. Any positive faces with persistence greater than δ_{min} and any negative faces paired with positive edges on the surface will be identified as choke faces.

This is to our knowledge, the first application of persistent 2-homology for extracting surface loops. Unlike [49] (using only persistent 1-homology), we need to handle the pairing between tets and faces as well as the pairing between faces and edges. With our particular order of adding

simplices to the filtration, most of the positive simplices will only have a small persistence.

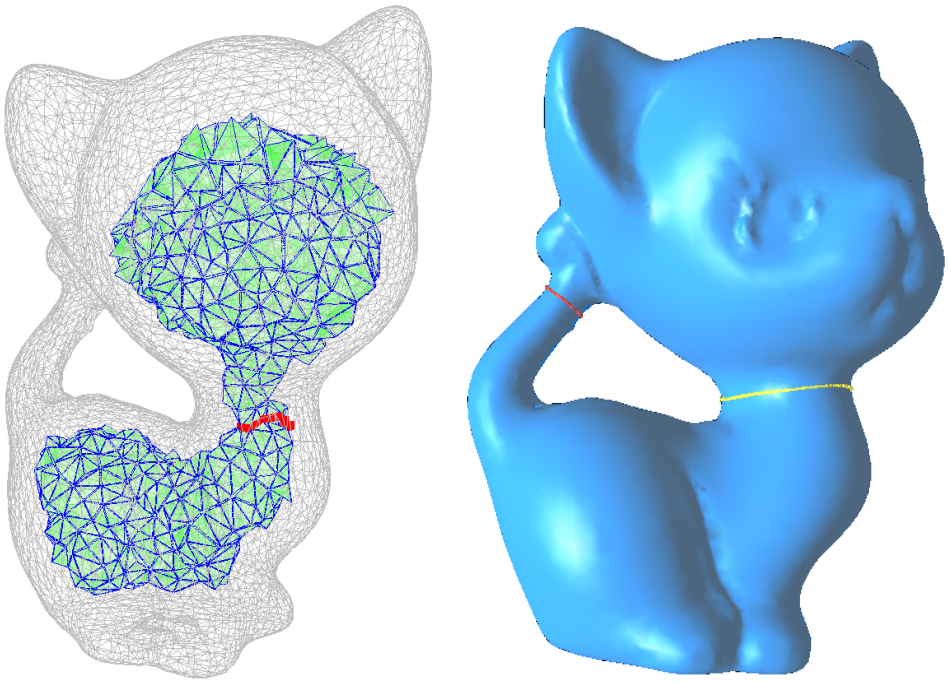

Figure 6.4: Our filtration is built in the order of distance from the surface. The filtration when the choking loop shown in yellow to the right is detected is the volume between the offset surface shown in green and the original surface. The 1-homology handle shown in red to the right (around the tail) has already been killed when we reach this offset distance.

6.3.2 Computation of the Associated Surface Loops

We now present a robust method to compute an approximation of the membrane B starting from a choke face. Intuitively speaking, we gradually deform the boundary loop of the membrane approximately along the gradient of the distance field to reach the surface in a fastest (greedy) way, so that the membrane swept will be a minimal one touching the boundary. We can see the procedure as partitioning the nearby tets in K into one connected cluster to the left of the membrane, and another to the right of the membrane, as in min-cut of the dual graph. We use the following fast procedure:

1. Add the choke face f to the membrane, and add the two tets adjacent to f to the two clusters.

2. Pick the vertex v_k in the boundary loop $(v_0v_1...v_nv_0)$ with the largest distance d from the surface.

3. Find the local optimal membrane patch within the one-ring neighborhood of v_k (within the filtration before the seed face is included), such that it morphs $v_{k-1}v_kv_{k+1}$ to a path connecting v_{k-1} to v_{k+1} on the boundary of the one-ring. This membrane patch partitions tets in the one-ring into left and right, consistent with those already classified as left or right.

4. Merge the local membrane patch separating the two classes of the one-ring to the membrane.

5. Repeat Steps 2, 3 and 4 until all boundary edges of the membrane are on the surface.

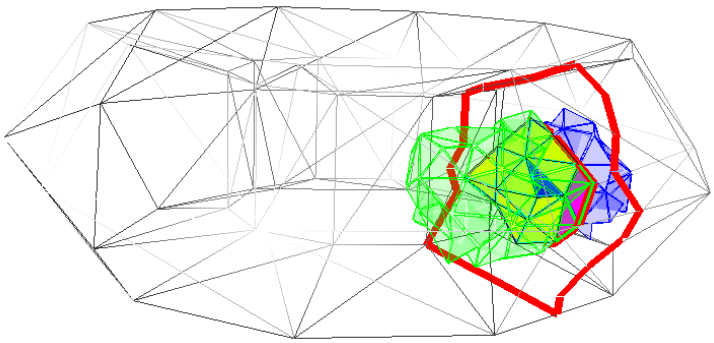

Figure 6.5: Starting from the blue triangle representing the choke face, we gradually expand the membrane separating the left (blue tets) and the right (green tets) internal space towards the surface, until the loop is entirely on the boundary. In the step shown here, the purple faces will be added to the yellow membrane. The final result is the red loop on the surface of the torus.

In Step 3, the membrane is optimal in terms of an average of the length of its edges on the boundary of the one-ring (to reduce area) and the distance of the corresponding vertices along the path to the boundary (to reach the surface fast).

Finally, to improve the geometric shape of the result, we employ the method in [198] to allow the path to go through the surface triangles instead of being restricted on edges, thus creating a smoother loop. We can allow it to reach a locally minimal length in terms of geodesic distance, or stop after few iterations to smooth the loop without deviating far. In practice, the edge loops are very close to geodesic loops to begin with.

6.4 Persistent Homology for Molecule Stability Analysis

In some applications, the exact membranes or choking loops are irrelevant. One such application is the analysis of the protein folding process, where we use Betti-1 number ($\beta_1 = dim(H_1)$) in persistent homology to measure the stability of a given biomolecule. However, in this case, the persistent homology theory is extended to cell complex to accommodate for regular grids.

6.4.1 Rationale

The shape of protein plays an important role in its functions. In order to perform their biological function, proteins usually fold into one specific spatial conformation. This final structure can be treated as a stable equilibrium state when all the interactions and forces, such as hydrogen bonding, ionic interaction, van der Waals force, and hydrophobic interaction, are balanced. In most cases, short-range forces dominate these interactions. Thus, the proximity between atoms, are important in determining the flexibility of the protein.

In fact, we propose to use the accumulated β_1 for a certain filtration to estimate the stability of the given molecule. A biomolecule is typically composed of a large number of atoms. Each type of atoms are often regarded as balls with a specific radius, which is the mean distance between the nucleus and the approximate boundary of the surrounding cloud of electrons. If we define the surface of an atom based on the radius, the surface of the molecule is the boundary of the union of all such small balls. Such surfaces are continuous surfaces with specific Betti numbers. As discussed before, β_1 represents the number of independent nontrivial loops on the surface. We observe that if each atom radius rescales to a smaller or larger value, β_1 changes accordingly. Since the nontrivial loops provide constraints on the spatial formation, we postulate that the energy level is correlated to $\beta_1(s)$ measure the number of loops at a given rescaling factor s of the radius. Thus, we propose to use β_1 as an indicator to study the stability property of the molecule. Starting from $s = 0$, we accumulate β_1, i.e. $\int_{s=0}^{\infty} \beta_1(s)$, as this indicator.

6.4.2 Algorithms

Instead of using triangular mesh to compute the persistent homology, we use cell complex to simulate the increase of radius of atoms. For such structured volumetric meshes, the adjacently information and incidence relations can be directly computed based on the coordinates without resorting to additional storage. Cell complex also provide us with a handy way to control and compare how the results are affected by different grid sizing.

The pairing and persistent homology algorithms are given as follows.

Algorithm 4 Pairing($\sigma, \beta_p, \beta_{p-1}$)

```
1: init b as boundary of σ
2: init c as the youngest positive (p − 1)-cell in b
3: while true do
4:    if c is unpaired or b is empty then
5:       break
6:    end if
7:    set b′ as the cycle killed by the cell paired with c
8:    add b′ to b
9:    set c to be the youngest positive (p − 1)-cell in b
10: end while
11: if b is empty then
12:    set σ as positive
13:    β_p = β_p + 1
14: else
15:    set σ as negative
16:    paired σ with c
17:    β_{p−1} = β_{p−1} − 1
18: end if
```

However, the above algorithm suffers from grid resolution dependency. The true energy estimate should not depend on the resolution of the discretization heavily. Thus, we propose a filtered version of β_1, which exclude the homology generators with persistence below a threshold from the integral. Note that the persistent homology algorithm remains the same.

Algorithm 5 Persistent homology algorithm

```
init boolean tables for vertices, edges, faces of complex to record their existence in filtration
compute distance field d(x,y,z) on vertices by fast marching algorithm
set marching distance threshold θ
do a partial sort (threshold θ) on vertices based on distance field
get sorted vertices as new list V (keep order)
init β0 as size of V
init list L to store result
for all vertex v in V do
    get E, F as one-ring edges, one-ring faces of v
    for all edge e in E not in the filtration yet do
        if boundary vertices of e have been added into filtration then
            Pairing(e, β1, β0)
            add e to filtration
        end if
    end for
    for all face f in F not in the filtration yet do
        if boundary edges of f have been added into filtration then
            Pairing(f,β2, β1)
            add f to filtration
        end if
    end for
    add v to filtration, append tuple (distance(v), β1) to list L.
end for
output list L
```

6.5 Results and Discussion

We ran our algorithm on a few genus-0 models. These surface loops, e.g. shown on the bunny model, have well-defined topological meaning of local separating membranes as given in the mathematical definition of choking loops.

For high genus models, there can be a lot of legitimate candidates for the shortest loops that can form a basis of the 1-homology group, as well as additional surface loops that do not belong to combinations of these bases, as in the genus-0 case. As shown on the C_{60} model (the Buckyball, genus 31), our algorithm produces all 90 useful handle loops, as opposed to only 31 of them produced by other methods. We also find the complete set of 32 tunnels.

We observe in our tests that there are often many more handle-type choking loops than ho-

Algorithm 6 modified Pairing($\sigma, \beta_2, \beta_1, L, \delta$) with filtering

```
1: init b as boundary of σ
2: init c as the youngest positive 1-cell in b
3: while true do
4:    if s is unpaired or b is empty then
5:       break
6:    end if
7:    set b' as the cycle killed by the cell paired with c
8:    add b' to b
9:    set c to be the youngest positive 1-cell in b
10: end while
11: if b is not empty then
12:    set σ as negative
13:    paired σ with c
14:    β1 = β1 − 1
15:    if distance(σ) − distance(c) < δ then
16:       for all tuple distance(v, β1) in L do
17:          if distance(c) < distance(v) < distance(σ) then
18:             decrease associated β1 value by 1
19:          end if
20:       end for
21:    end if
22: end if
```

mology generators, since there are often more narrowing passages for the inside of the object. However, for protein models, there can be more tunnels than those obtained by 1-homology of surfaces. Those tunnels are important in automatic detection of ion channels crucial in analysis of the biomolecular surfaces. Those tunnels allow ions to flow past membranes of cells, and they play important biological roles, e.g., in nerve impulse of the nervous system. For models with knots (boundary of Seifert's surface [174]), we did not find additional choking tunnels.

The maximum offset distance d_{max} specifies how far we want to go inside the volume, and the minimum persistence δ_{min} determines which structures are topological noises to be filtered out. In most tests, we found the default setting of $d_{max} = 50\%$ and $\delta_{min} = 10\%$ to produce satisfying results, with all the important loops included without introducing a high density of loops. Alternatively, we can set $d_{max} = 100\%$ with a low δ_{min} to extract all useful chokepoints, and allow the user to adjust them interactively after the first run. The most costly step in our implementation

Table 6.1: Statistics of the results (all time measurements in milliseconds, taken on a Windows 7 system with Intel Core i7@2.8GHz and 12GB RAM). From left to right: surface vertex number, total vertex number, tetrahedralization time by TetGen, preprocessing time (distance field construction), time running persistent homology for surface mesh, time running persistent 1-homology and 2-homology for inside volume, time to find and postprocess all the homology generators in the basis and all additional choking loops, genus g, number of additional choking loops k, TetGen parameters used, maximum distance in building the filtration, and persistence threshold for the loops shown. Only timing for handles is reported, as that for tunnels is similar.

model	surf#v	total#v #f #t	TetGen	prep	surf	inside	basis	choke	g	k	param	$d_{max}(\%)$	$\delta_{min}(\%)$
armadillo	40K	70K,656K,307K	13,011	12,948	3,292	79,763	0	531	0	4	pfq1.2	50	6
1mag	17K	33K,328K,155K	10,437	4,773	1,358	19,094	390	671	8	6	pfq1	75	15
bunny	26K	98K,1088K,531K	17,846	11,060	2,387	91,026	0	640	0	1	pfq1	75	2.5
david	26K	46K,430K,202K	8,955	5,725	2,761	25,709	1,154	234	5	3	pfq1.2	50	10
fertility	47K	99K,985K,469K	29,235	17,924	3,245	79,935	811	344	4	2	pfq1.2	50	10
kitten	8K	17K,166K,79K	3,728	1,482	546	10,187	47	140	1	1	pf1.2	50	15
neptune	32K	52K,471K,219K	9,890	8,549	9,516	130,963	94	4,805	3	7	pfq1.2	50	10
tangle	7K	11K,97K,45K	2,809	1,107	546	5,335	47	94	5	7	pfq1.2	50	10

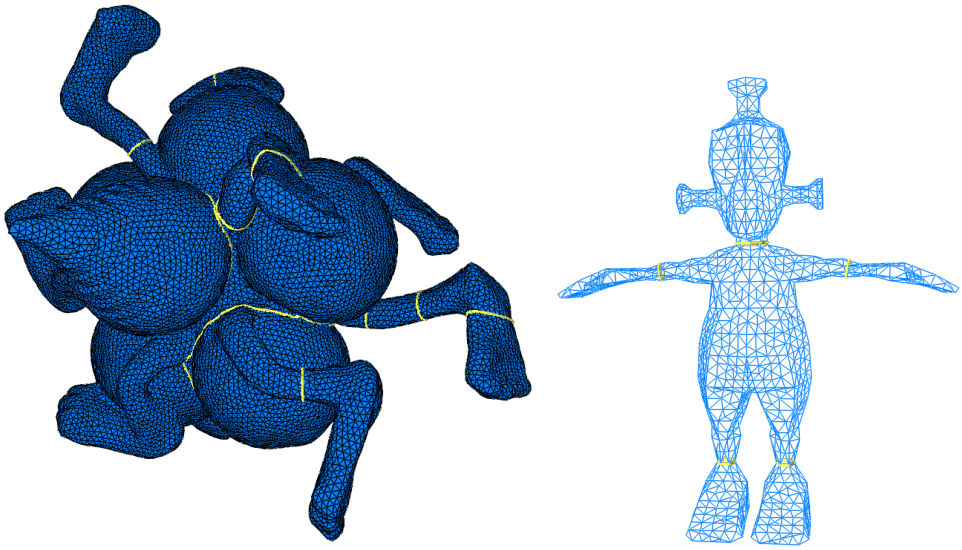

Figure 6.6: Additional genus-0 models and their choking loops.

is the persistent homology pairing algorithm (see Table 6.1).While the worst case of computing persistent homology has upper bound of $O(m^3)$,where m is the size of the simplicial complex, the practical running time appears to be nearly linear [56]. . All the other steps (computing persistence to detect choke faces, tracing back to find choking loops, and postprocessing) take mostly less than a second. If we are interested only in smaller features for a particular application such as filtering, we can set d_{max} to a small number, which greatly improves the efficiency.

The tessellation of the inside and outside space generates numbers of vertices roughly propor-

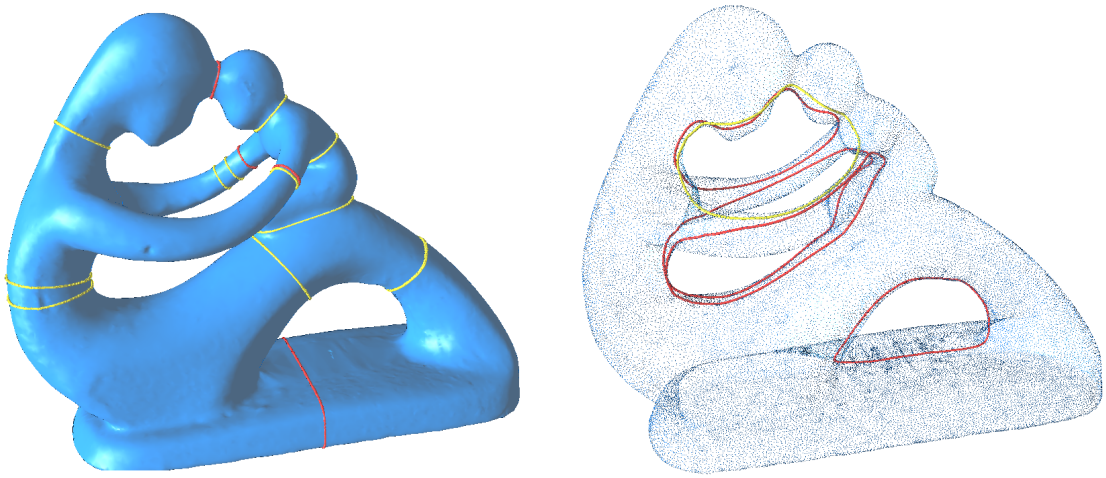

Figure 6.7: The handles and tunnels on fertility model. We render only the vertices when showing the tunnels to make them more visible. The red loops can be obtained by homology generators, and the yellow loops are the additional choking loops.

tional to the numbers of surface vertices, which was enough to compute the choking loops, since the efficient postprocessing can improve the geometric shape. We have tested on both interpolated signed distance field and Dijkstra distance from the edge graph, and found similar results even at low resolution. As long as the distance field is accurate enough to discern the life cycles of persistent choking loops, the results are insensitive to tessellation differences in our tests, especially after the proposed post-processing. See Figure 6.11 for a typical example. In case the application requires better precision for feature size and persistence, a high resolution tetrahedral mesh can be used.

6.5.1 Homology-based Analysis on Fullerenes

For fullerenes, the exact locations of all atoms are known (http://www.ccl.net/cca/data/fullerenes/). The datasets record the exact location of each carbon atom in 3D. To compute β_1 numbers during the increase of the radii, a distance filed is needed to guide how the cells are added into the filtration. We first compute the bounding box of the model, and then generate the regular grid mesh based on the bounding box. A distance field can be computed by fast marching method starting from each atom's center [151]. If the molecule contains multiple types of atoms, fast marching is performed

Figure 6.8: A number of additional topologically interesting loops can be found on the Neptune model.

on each type, and the minimal value at each grid point among all the fast marching result is used for the final distance field.

For each specific distance value, an iso-surface can be extracted by marching cubes [115, 121] to study the topological features, e.g. the Betti numbers. However, as the grid resolution increases, the topological features will rely heavily on the grid size [138]. Thus, to evaluate the topological features on the filtration created by the distance field, the persistent homology method can be used to approximate the Betti number computation for each different distance value. The cells are gradually incorporated according to their marching distance and the topological features are captured during the traversal of the filtration.

Our results show that integral of β_1 is highly dependent on the grid spacing without filtering out small scale nontrivial loops (see Fig. 6.13). Thus we use the filtered results with a minimum

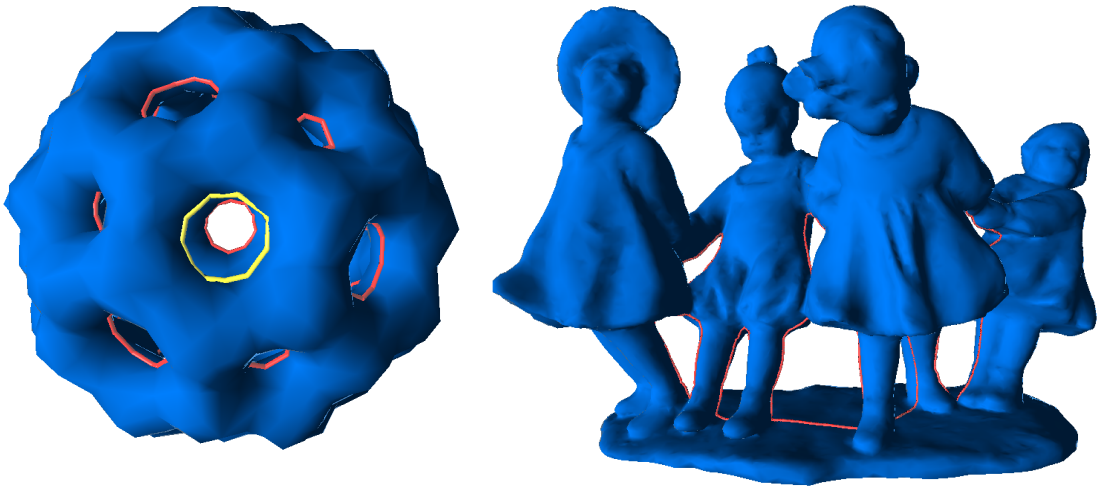

Figure 6.9: More models showing the tunnels detected by our algorithm.

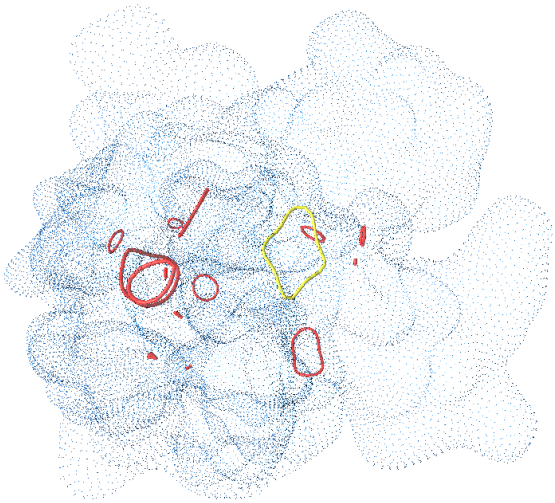

Figure 6.10: We find one additional tunnel loop aside from the 1-homology generators for 2kix protein. Some of the red loops here can be seen as topological noise.

persistence (see Fig. 6.14). It can be seen that our simple topology-based estimates produces decent prediction on the stability of fullerenes when compared with physical estimates, which would require inefficient molecular dynamics simulation, when applied to irregularly shaped molecules. In Fig. 6.15, the estimate is monotonically decreasing as the number of carbon atoms increasing in the fullerene series. This follows closely to the cohesive energies computed by second-order

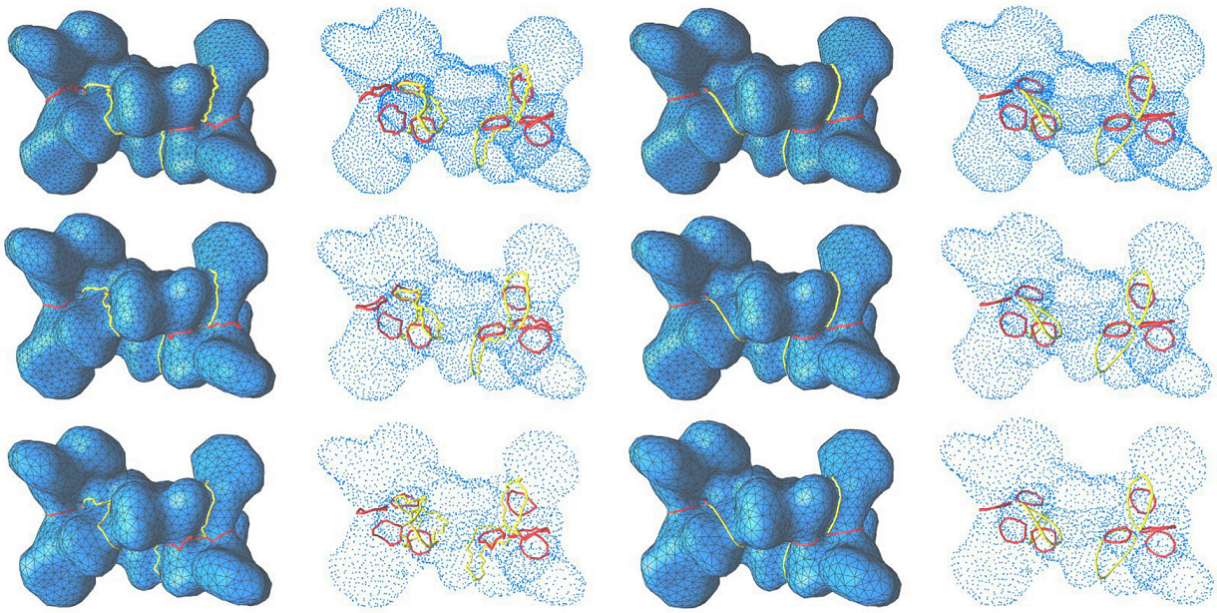

Figure 6.11: Comparison of the handle loop results on 1mag at different resolutions. All the loops (8 homology generators in red and 4 additional choking loops in yellow) are captured by setting $d_{max} = 70\%$ and $\delta_{min} = 25\%$. Top to bottom: meshes with surface vertex counts 7.7k, 4.8k and 3.4k (TetGen parameters pfq1.2a0.5, pfq1.2a1 and pfq1.2a1.5, resp.); left to right: choking loops, their unoccluded view, post-processed loops, and their unoccluded view.

Møller-Plesset perturbation theory [68].

6.6 Summary

We present a method to compute nontrivial loops that are not possible to produce using conventional topological methods for surfaces. Our contribution is threefold: we first provide a mathematical definition for such loops using persistent homology theory; we then provide an efficient algorithm based on our definition; last, we examine the applicability of the loop count in molecule stability prediction. Theoretically, our definitions can be seen as related to topological structures in an extension of proximity complexes. While proximity complexes are built from unions of balls with radius d in a finite point set, we use all the points on a surface.

A potential limitation of the algorithm is that if there are two nearby short loops, the shorter one might push the other slightly away from the geometric optimal location, depending on the

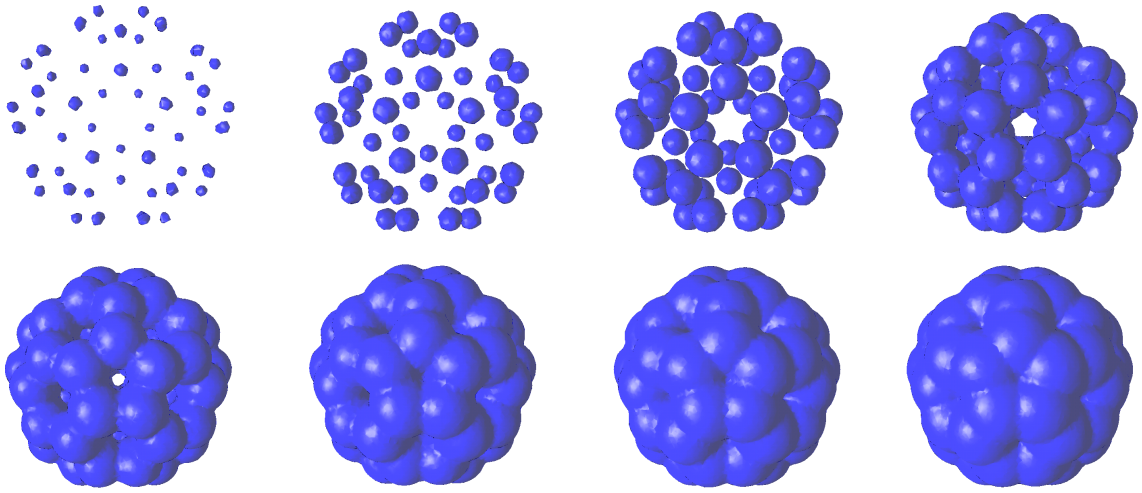

Figure 6.12: Illustration of Fullerene C_{60} surface grows from the atom center location.

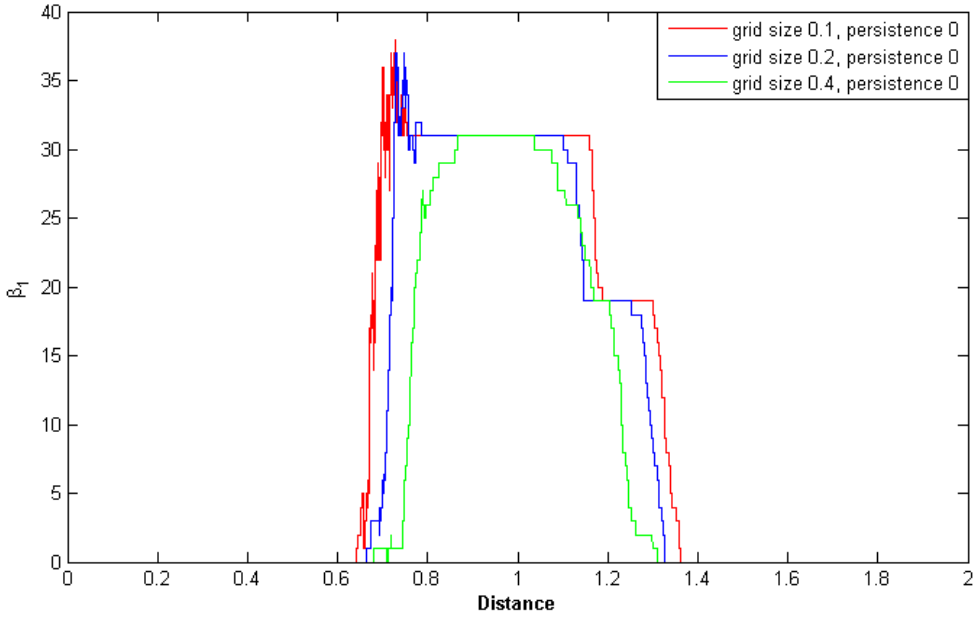

Figure 6.13: Comparison among different grid sizes without persistence filtering for β_1 curve on C_{60} data.

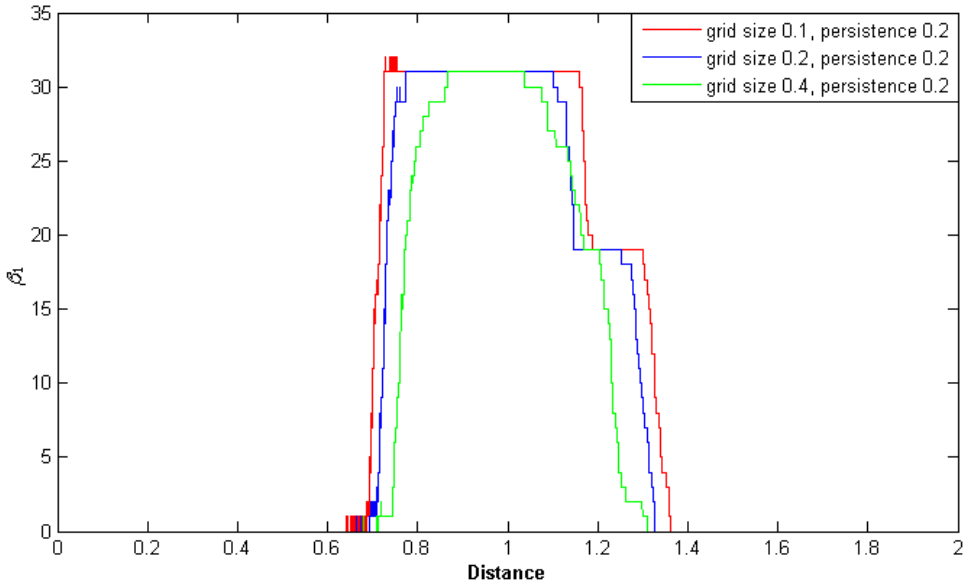

Figure 6.14: Comparison among different grid sizes with persistence filtering size 0.2 for β_1 curve on C_{60} data.

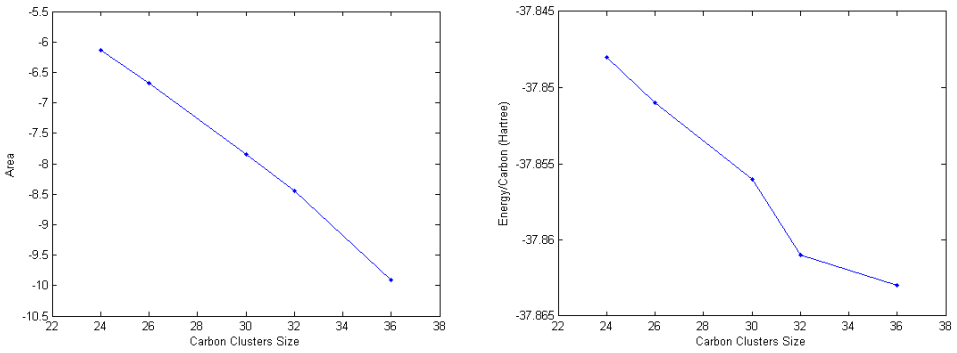

Figure 6.15: Comparison between Relative MP2 energies (left) from [68] and area integral value of β_1 (right) for different carbon clusters.

persistence δ one chooses—although we found in practice that the postprocessing loop shortening step can usually move them to decent locations (e.g. loops near the left knee of David statue in Figure 6.1). Another potential problem is that the loops discovered first can be around a cross section in the shape of a long thin rectangle instead of a cross section in the shape of a disk, (e.g., those on the basis of David statue). However, we can eventually detect all these loops, and simply sort them again based on lengths. We can also easily discard such loops if required by the application, by allowing the loop around the choke face to deform only within a certain distance, and eliminating those failed to reach the surface. On the other hand, for motion planning or constriction detection, this long narrow passage should be detected earlier than shorter geodesic loops, as a ball with radius d will get stuck.

For future work, we plan to explore the possibilities of improving computational time for level sets (used by many biomedical or biomolecular applications), leveraging the regular grid structure of the inside and outside domains, or using the implicit representation directly. We also wish to explore other 3D Morse functions (e.g., distance from the medial axes, diffusion times, or diffusion distances) to guide the construction of the filtration used in persistent homology. An obvious application that derives from our method is topological filtering, where we only need to set a short distance for the surface offset to kill tiny handles (offset inward) and to kill tiny tunnels (offset outward).

Chapter 7

CONCLUSION

In this thesis, we present an assortment of computational tools aimed at geometric and topological modeling of *large and complex* surfaces and volumes. They cover an entire pipeline from the data structure to global topological feature detection. We demonstrate the efficacy and efficiency of the proposed data structure and algorithms through sample applications to biomolecular surface analysis, since those surfaces are constructed from possibly millions of atoms and are inherently more complicated than man-made objects, which, on the contrary, can often be assembled through regular primitive shapes.

By employing the topological combinatorial maps and the fact that practical meshes have limited types of cells, we proposed a compact data structure, with around *10% memory cost* of what is offered by popular geometric modeling libraries. Such a data structure can greatly reduce the memory footprint of geometric and topological algorithms requiring constant-time incidence and adjacency queries. Furthermore, our constructions can be easily extended to objects embedded in higher dimensions.

With the compact representation, we then developed a *comprehensive framework* for geometric analysis, including the commonly used measurements such as area, volume, and curvature. In addition, we also incorporate these procedures based on the implicit surface representations defined on Cartesian grids. While most of the techniques included in our framework are existing methods, they have not been tested on real datasets in the biomolecular context, or applied to models with such complexities. We not only run thorough tests on both analytical models and real microscopy datasets to verify and select the proper algorithms and parameters for each stage, but also provide conversion tools between possibly different representations used in consecutive steps. We also demonstrate the utility of our system in combined analysis of curvatures and electrostatics, both of which play important roles in the research on protein docking and drug design.

Unlike the local geometric descriptors such as curvatures, topological features are intrinsically global structures. Even with persistent homology theory, which provides continuous measurements for topological invariants, it may still be hard to identify which persistence is pertinent to the specific application. We present *the first definition on 3D bottlenecks* by using the signed distance function to the surface as the Morse function of the filtration. The resulting systems of loops on the boundary surface provide a more intuitive notion of tunnels and handles than what is offered by the regular homology generators, which is limited to twice the genus number. In addition to the direct use of our algorithm in segmentation and detection of features such as ion channels in biomolecular membranes, we also explored the application of the topological concept in protein stability, using the integral of the number of nontrivial loops when we increase the threshold distance from zero to infinity.

In summary, our work combines novel and existing geometric and topological approaches, and offer simple to use, mathematically sound, efficient algorithms for common geometric and topological analysis of large and complex shapes such as biomolecular surfaces.

7.1 Future work

Our data structure can be modified to accommodate dynamical changes in the connectivity through replacing the indices by pointers, and be generalized to arbitrary dimensions. Higher compression rates may also be achieved through employing entropy encoding techniques for the tables we generated to represent the connectivity. Space filling curves provide another possibility for differential encoding of the indices.

Our geometric analysis toolkits can be made more efficient by hierarchical data structures and adaptive refinement. The accuracy and smoothness may benefit from using subdivision surfaces or other high-level descriptions of the underlying surfaces. A mixture of the Lagrangian and Eulerian representations may also benefit temporal sequences of deforming biomolecules or other geometric objects.

Our topological analysis relies on volumetric meshes even when only the surface is analyzed,

which can be inefficient when the resolution on the persistence measure is high. It is possible to explore efficient algorithms when only scarce samples are needed for the persistent homology instead of the full filtration. Alternative definitions of bottleneck may also be relevant when the application requires separating membranes of a small area instead of a small diameter.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Burcu Akinci, Frank Boukamp, Chris Gordon, Daniel Huber, Catherine Lyons, and Kuhn Park, *A formalism for utilization of sensor systems and integrated project models for active construction quality control*, Automation in Construction **15** (2006), no. 2, 124–138.

[2] Pierre Alliez, Giuliana Ucelli, Craig Gotsman, and Marco Attene, *Recent advances in remeshing of surfaces*, Shape analysis and structuring, Springer, 2008, pp. 53–82.

[3] Tyler J Alumbaugh and Xiangmin Jiao, *Compact array-based mesh data structures*, Proceedings of the 14th International Meshing Roundtable, Springer, 2005, pp. 485–503.

[4] Christian B Anfinsen, *Studies on the principles that govern the folding of protein chains*, 1972.

[5] F Betul Atalay and David M Mount, *Pointerless implementation of hierarchical simplicial meshes and efficient neighbor finding in arbitrary dimensions*, International Journal of Computational Geometry & Applications **17** (2007), no. 06, 595–631.

[6] Nathan A Baker, *Biomolecular applications of poisson-boltzmann methods*, Reviews in computational chemistry **21** (2005), 349.

[7] Nathan A Baker, Donald Bashford, and David A Case, *Implicit solvent electrostatics in biomolecular simulation*, New algorithms for macromolecular simulation, Springer, 2006, pp. 263–295.

[8] Nathan A Baker, David Sept, Simpson Joseph, Michael J Holst, and J Andrew McCammon, *Electrostatics of nanosystems: application to microtubules and the ribosome*, Proceedings of the National Academy of Sciences **98** (2001), no. 18, 10037–10041.

[9] PW Bates, Zhan Chen, Yuhui Sun, Guo-Wei Wei, and Shan Zhao, *Geometric and potential driving formation and evolution of biomolecular surfaces*, Journal of Mathematical Biology **59** (2009), no. 2, 193–231.

[10] PW Bates, Guo-Wei Wei, and Shan Zhao, *Minimal molecular surfaces and their applications*, Journal of Computational Chemistry **29** (2008), no. 3, 380–391.

[11] PW Bates, GW Wei, and Shan Zhao, *The minimal molecular surface*, arXiv preprint q-bio/0610038 (2006), 1–9.

[12] Martin Z Bazant, Brian D Storey, and Alexei A Kornyshev, *Double layer in ionic liquids: Overscreening versus crowding*, Physical Review Letters **106** (2011), no. 4, 046102.

[13] Mark W Beall and Mark S Shephard, *A general topology-based mesh data structure*, International Journal for Numerical Methods in Engineering **40** (1997), no. 9, 1573–1596.

[14] Marshall Wayne Bern and Paul E Plassmann, *Mesh generation*, Pennsylvania State University, Department of Computer Science and Engineering, College of Engineering, 1997.

[15] Claudia Bertonati, Barry Honig, and Emil Alexov, *Poisson-boltzmann calculations of non-specific salt effects on protein-protein binding free energies*, Biophysical journal **92** (2007), no. 6, 1891–1899.

[16] Stephan Bischoff, Darko Pavic, and Leif Kobbelt, *Automatic restoration of polygon models*, ACM Transactions on Graphics (TOG) **24** (2005), no. 4, 1332–1352.

[17] Daniel K Blandford, Guy E Blelloch, David E Cardoze, and Clemens Kadow, *Compact representations of simplicial meshes in two and three dimensions*, International journal of computational geometry & applications **15** (2005), no. 01, 3–24.

[18] Peter Blomgren and Tony F Chan, *Color tv: total variation methods for restoration of vector-valued images*, Image Processing, IEEE Transactions on **7** (1998), no. 3, 304–309.

[19] Jean-Daniel Boissonnat and Steve Oudot, *Provably good sampling and meshing of surfaces*, Graphical Models **67** (2005), no. 5, 405–451.

[20] Dobrina Boltcheva, David Canino, Sara Merino Aceituno, Jean-Claude Léon, Leila De Floriani, and Franck Hétroy, *An iterative algorithm for homology computation on simplicial shapes*, Computer-Aided Design **43** (2011), no. 11, 1457–1467.

[21] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy, *Polygon mesh processing*, CRC press, 2010.

[22] Mario Botsch, Stephan Steinberg, Stephan Bischoff, and Leif Kobbelt, *Openmesh-a generic and efficient polygon mesh data structure*, 2002.

[23] PT Bremer, EM Bringa, MA Duchaineau, AG Gyulassy, D Laney, A Mascarenhas, and V Pascucci, *Topological feature extraction and tracking*, Journal of Physics: Conference Series, vol. 78, IOP Publishing, 2007, p. 012007.

[24] Sergio Cabello, Éric Colin de Verdière, and Francis Lazarus, *Finding shortest non-trivial cycles in directed graphs on surfaces*, Proceedings of the 2010 annual symposium on Computational geometry, ACM, 2010, pp. 156–165.

[25] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro, *Geodesic active contours*, International journal of computer vision **22** (1997), no. 1, 61–79.

[26] Waldemar Celes, Glaucio H Paulino, and Rodrigo Espinha, *A compact adjacency-based topological data structure for finite element mesh representation*, International Journal for Numerical Methods in Engineering **64** (2005), no. 11, 1529–1556.

[27] Erin W Chambers, Jeff Erickson, and Amir Nayyeri, *Minimum cuts and shortest homologous cycles*, Proceedings of the 25th annual symposium on Computational geometry, ACM, 2009, pp. 377–385.

[28] Antonin Chambolle and Pierre-Louis Lions, *Image recovery via total variation minimization and related problems*, Numerische Mathematik **76** (1997), no. 2, 167–188.

[29] Tony Chan, Antonio Marquina, and Pep Mulet, *High-order total variation-based image restoration*, SIAM Journal on Scientific Computing **22** (2000), no. 2, 503–516.

[30] Frédéric Chazal, Leonidas J Guibas, Steve Y Oudot, and Primoz Skraba, *Analysis of scalar fields over point cloud data*, Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2009, pp. 1021–1030.

[31] Chao Chen, Daniel Freedman, et al., *Quantifying homology classes*, Proceedings of the 25th Annual Symposium on the Theoretical Aspects of Computer Science, 2008, pp. 169–180.

[32] Duan Chen, Zhan Chen, Changjun Chen, Weihua Geng, and Guo-Wei Wei, *Mibpb: A software package for electrostatic analysis*, Journal of computational chemistry **32** (2011), no. 4, 756–770.

[33] Duan Chen, Zhan Chen, and Guo-Wei Wei, *Quantum dynamics in continuum for proton transport ii: Variational solvent–solute interface*, International journal for numerical methods in biomedical engineering **28** (2012), no. 1, 25–51.

[34] Zhan Chen, Nathan A Baker, and Guo-Wei Wei, *Differential geometry based solvation model i: Eulerian formulation*, Journal of computational physics **229** (2010), no. 22, 8231–8258.

[35] Zhan Chen, Nathan A Baker, and GW Wei, *Differential geometry based solvation model ii: Lagrangian formulation*, Journal of mathematical biology **63** (2011), no. 6, 1139–1200.

[36] Zhan Chen and Guo-Wei Wei, *Differential geometry based solvation model. iii. quantum formulation*, The Journal of chemical physics **135** (2011), no. 19, 194108.

[37] Zhan Chen, Shan Zhao, Jaehun Chun, Dennis G Thomas, Nathan A Baker, Peter W Bates, and GW Wei, *Variational approach for nonpolar solvation analysis*, The Journal of chemical physics **137** (2012), no. 8, 084101.

[38] Gregory Cipriano, George N Phillips, and Michael Gleicher, *Multi-scale surface descriptors*, Visualization and Computer Graphics, IEEE Transactions on **15** (2009), no. 6, 1201–1208.

[39] David Cohen-Steiner, Jean-Marie Morvan, et al., *Second fundamental measure of geometric sets and local approximation of curvatures*, Journal of Differential Geometry **74** (2006), no. 3, 363–394.

[40] Michael L Connolly, *Depth-buffer algorithms for molecular modelling*, Journal of Molecular Graphics **3** (1985), no. 1, 19–24.

[41] Robert B Corey and Linus Pauling, *Molecular models of amino acids, peptides, and proteins*, Review of Scientific Instruments **24** (1953), 621–627.

[42] Martin D Crossley, *Essential topology*, Springer, 2006.

[43] Guillaume Damiand, *Combinatorial maps*, CGAL User and Reference Manual, CGAL Editorial Board, 4.0 ed., 2012.

[44] Éric Colin De Verdière and Francis Lazarus, *Optimal system of loops on an orientable surface*, Discrete & Computational Geometry **33** (2005), no. 3, 507–534.

[45] Mathieu Desbrun, Eva Kanso, and Yiying Tong, *Discrete differential forms for computational modeling*, Discrete differential geometry, Springer, 2008, pp. 287–324.

[46] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr, *Implicit fairing of irregular meshes using diffusion and curvature flow*, Proceedings of the 26th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 1999, pp. 317–324.

[47] Tamal K Dey, Anil N Hirani, and Bala Krishnamoorthy, *Optimal homologous cycles, total unimodularity, and linear programming*, SIAM Journal on Computing **40** (2011), no. 4, 1026–1044.

[48] Tamal K Dey, Kuiyu Li, and Jian Sun, *Computing handle and tunnel loops with knot linking*, Computer-Aided Design **41** (2009), no. 10, 730–738.

[49] Tamal K Dey, Kuiyu Li, Jian Sun, and David Cohen-Steiner, *Computing geometry-aware handle and tunnel loops in 3d models*, ACM Transactions on Graphics (TOG), vol. 27, ACM, 2008, p. 45.

[50] David P Dobkin and Michael J Laszlo, *Primitives for the manipulation of three-dimensional subdivisions*, Proceedings of the third annual symposium on Computational geometry, ACM, 1987, pp. 86–99.

[51] Vinciane d'Otreppe, Romain Boman, and Jean-Philippe Ponthot, *Generating smooth surface meshes from multi-region medical images*, International Journal for Numerical Methods in Biomedical Engineering **28** (2012), no. 6-7, 642–660.

[52] Qiang Du, Vance Faber, and Max Gunzburger, *Centroidal voronoi tessellations: applications and algorithms*, SIAM review **41** (1999), no. 4, 637–676.

[53] J Dzubiella, JMJ Swanson, and JA McCammon, *Coupling nonpolar and polar solvation free energies in implicit solvent models*, The Journal of chemical physics **124** (2006), no. 8, 084905.

[54] Paul H Edelman and Michael E Saks, *Combinatorial representation and convex dimension of convex geometries*, Order **5** (1988), no. 1, 23–32.

[55] Herbert Edelsbrunner and John Harer, *Computational topology: an introduction*, American Mathematical Soc., 2010.

[56] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian, *Topological persistence and simplification*, Discrete and Computational Geometry **28** (2002), no. 4, 511–533.

[57] Jack Edmonds, *A combinatorial representation of polyhedral surfaces*, Notices of the American Mathematical Society **7** (1960), 646.

[58] Frank Eisenhaber and Patrick Argos, *Improved strategy in analytic surface calculation for molecular systems: handling of singularities and computational efficiency*, Journal of Computational Chemistry **14** (1993), no. 11, 1272–1280.

[59] Jeff Erickson and Kim Whittlesey, *Greedy optimal homotopy and homology generators*, Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2005, pp. 1038–1046.

[60] Herbert Federer, *Curvature measures*, Transactions of the American Mathematical Society (1959), 418–491.

[61] Xin Feng, Yuanzhen Wang, Yanlin Weng, and Yiying Tong, *Compact combinatorial maps in 3d*, Computational Visual Media, Springer, 2012, pp. 194–201.

[62] _____, *Compact combinatorial maps: A volume mesh data structure*, Graphical Models **75** (2013), no. 3, 149–156.

[63] Xin Feng, Kelin Xia, Zhan Chen, Yiying Tong, and Guo-Wei Wei, *Multiscale geometric modeling of macromolecules ii: Lagrangian representation*, Journal of computational chemistry **34** (2013), no. 24, 2100–2120.

[64] Xin Feng, Kelin Xia, Yiying Tong, and Guo-Wei Wei, *Geometric modeling of subcellular structures, organelles, and multiprotein complexes*, International journal for numerical methods in biomedical engineering **28** (2012), no. 12, 1198–1223.

[65] JJ Fernandez, S Li, and V Lucic, *Three-dimensional anisotropic noise reduction with automated parameter tuning: application to electron cryotomography*, Current Topics in Artificial Intelligence, Springer, 2007, pp. 60–69.

[66] Jose-Jesus Fernandez, *Tomobflow: feature-preserving noise filtering for electron tomography*, BMC bioinformatics **10** (2009), no. 1, 178.

[67] José-Jesús Fernández and Sam Li, *An improved algorithm for anisotropic nonlinear diffusion for denoising cryo-tomograms*, Journal of structural biology **144** (2003), no. 1, 152–161.

[68] Martin Feyereisen, Maciej Gutowski, Jack Simons, and Jan Almlöf, *Relative stabilities of fullerene, cumulene, and polyacetylene structures for cn: n= 18–60*, The Journal of chemical physics **96** (1992), no. 4, 2926–2932.

[69] Federico Fogolari and James M Briggs, *On the variational approach to poisson–boltzmann free energies*, Chemical Physics Letters **281** (1997), no. 1, 135–139.

[70] Alexander Fotin, Yifan Cheng, Piotr Sliz, Nikolaus Grigorieff, Stephen C Harrison, Tomas Kirchhausen, and Thomas Walz, *Molecular model for a complete clathrin lattice from electron cryomicroscopy*, Nature **432** (2004), no. 7017, 573–579.

[71] Achilleas S Frangakis and Reiner Hegerl, *Noise reduction in electron tomographic reconstructions using nonlinear anisotropic diffusion*, Journal of structural biology **135** (2001), no. 3, 239–250.

[72] Emilio Gallicchio, MM Kubo, and Ronald M Levy, *Enthalpy-entropy and cavity decomposition of alkane hydration free energies: Numerical results and implications for theories of hydrophobic solvation*, The Journal of Physical Chemistry B **104** (2000), no. 26, 6271–6285.

[73] Emilio Gallicchio and Ronald M Levy, *Agbnp: An analytic implicit solvent model suitable for molecular dynamics simulations and high-resolution modeling*, Journal of computational chemistry **25** (2004), no. 4, 479–499.

[74] Emilio Gallicchio, Linda Yu Zhang, and Ronald M Levy, *The sgb/np hydration free energy model based on the surface generalized born solvent reaction field and novel nonpolar hydration free energy estimators*, Journal of computational chemistry **23** (2002), no. 5, 517–529.

[75] Weihua Geng and Guo-Wei Wei, *Multiscale molecular dynamics using the matched interface and boundary method*, Journal of computational physics **230** (2011), no. 2, 435–457.

[76] Weihua Geng, Sining Yu, and Guowei Wei, *Treatment of charge singularities in implicit solvent models*, The Journal of chemical physics **127** (2007), no. 11, 114106.

[77] Paul Louis George, Frédéric Hecht, and E Saltel, *Automatic mesh generator with specified boundary*, Computer methods in applied mechanics and engineering **92** (1991), no. 3, 269–288.

[78] Guy Gilboa, Nir Sochen, and Yehoshua Y Zeevi, *Forward-and-backward diffusion processes for adaptive image enhancement and denoising*, Image Processing, IEEE Transactions on **11** (2002), no. 7, 689–703.

[79] ———, *Image sharpening by flows based on triple well potentials*, Journal of Mathematical Imaging and Vision **20** (2004), no. 1-2, 121–131.

[80] Dirk Gillespie, Wolfgang Nonner, and Robert S Eisenberg, *Density functional theory of charged, hard-sphere fluids*, Physical Review E **68** (2003), no. 3, 031503.

[81] Michael K Gilson, Malcolm E Davis, Brock A Luty, and J Andrew McCammon, *Computation of electrostatic forces on solvated molecules using the poisson-boltzmann equation*, The Journal of Physical Chemistry **97** (1993), no. 14, 3591–3600.

[82] Valentin Gogonea and Eiji Ōsawa, *Implementation of the solvent effect in molecular mechanics. 1. model development and analytical algorithm for the solvent-accessible surface area*, Supramolecular Chemistry **3** (1994), no. 4, 303–317.

[83] Aleksey Golovinskiy and Thomas Funkhouser, *Consistent segmentation of 3d models*, Computers & Graphics **33** (2009), no. 3, 262–269.

[84] John B Greer and Andrea L Bertozzi, *H^ 1 solutions of a class of fourth order nonlinear equations for image processing*, Discrete and continuous dynamical systems **10** (2004), no. 1/2, 349–366.

[85] _____, *Traveling wave solutions of fourth order pdes for image processing*, SIAM Journal on Mathematical Analysis **36** (2004), no. 1, 38–68.

[86] Eitan Grinspun, P Schröder, and Mathieu Desbrun, *Discrete differential geometry: an applied introduction*, ACM SIGGRAPH Course **7** (2006), 1–83.

[87] André Guéziec and Robert Hummel, *Exploiting triangulated surface extraction using tetrahedral decomposition*, Visualization and Computer Graphics, IEEE Transactions on **1** (1995), no. 4, 328–342.

[88] Leonidas Guibas and Jorge Stolfi, *Primitives for the manipulation of general subdivisions and the computation of voronoi*, ACM Transactions on Graphics (TOG) **4** (1985), no. 2, 74–123.

[89] Attila G Gyulassy, Mark A Duchaineau, Vijay Natarajan, Valerio Pascucci, Eduardo M Bringa, Andrew Higginbotham, and Bernd Hamann, *Topologically clean distance fields*, Visualization and Computer Graphics, IEEE Transactions on **13** (2007), no. 6, 1432–1439.

[90] Franck Hétroy, *Constriction computation using surface curvature*, Eurographics (short paper), 2005, pp. 1–4.

[91] Franck Hétroy and Dominique Attali, *Detection of constrictions on closed polyhedral surfaces*, Proceedings of the symposium on Data visualisation 2003, Eurographics Association, 2003, pp. 67–74.

[92] Julie L Hodgkinson, Ashley Horsley, David Stabat, Martha Simon, Steven Johnson, Paula CA da Fonseca, Edward P Morris, Joseph S Wall, Susan M Lea, and Ariel J Blocker, *Three-dimensional reconstruction of the shigella t3ss transmembrane regions reveals 12-fold symmetry and novel features throughout*, Nature structural & molecular biology **16** (2009), no. 5, 477–485.

[93] Jin Huang, Tengfei Jiang, Yuanzhen Wang, Yiying Tong, and Hujun Bao, *Automatic frame field guided hexahedral mesh generation*, Tech. report, Technical Report MSU-CSE-12-9, Department of Computer Science, Michigan State University, East Lansing, Michigan, 2012.

[94] Giuseppe F Italiano, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen, *Improved algorithms for min cut and max flow in undirected planar graphs*, Proceedings of the 43rd annual ACM symposium on Theory of computing, ACM, 2011, pp. 313–322.

[95] Wen Jiang, Matthew L Baker, Qiu Wu, Chandrajit Bajaj, and Wah Chiu, *Applications of a bilateral denoising filter in biological electron microscopy*, Journal of structural biology **144** (2003), no. 1, 114–122.

[96] Lili Ju, Qiang Du, and Max Gunzburger, *Probabilistic methods for centroidal voronoi tessellations and their parallel implementations*, Parallel Computing **28** (2002), no. 10, 1477–1500.

[97] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren, *Dual contouring of hermite data*, ACM Transactions on Graphics (TOG) **21** (2002), no. 3, 339–346.

[98] Sagi Katz and Ayellet Tal, *Hierarchical mesh decomposition using fuzzy clustering and cuts*, ACM Transactions on Graphics (Proc. SIGGRAPH) **22** (2003), no. 3, 954–961.

[99] Gordon Kindlmann, Ross Whitaker, Tolga Tasdizen, and Torsten Moller, *Curvature-based transfer functions for direct volume rendering: Methods and applications*, Visualization, 2003, IEEE, 2003, pp. 513–520.

[100] Benjamin S Kirk, John W Peterson, Roy H Stogner, and Graham F Carey, *libmesh: a c++ library for parallel adaptive mesh refinement/coarsening simulations*, Engineering with Computers **22** (2006), no. 3-4, 237–254.

[101] Leif P Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel, *Feature sensitive surface extraction from volume data*, Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM, 2001, pp. 57–66.

[102] Jan J Koenderink and Andrea J van Doorn, *Surface shape and curvature scales*, Image and vision computing **10** (1992), no. 8, 557–564.

[103] Victor A Kostyuchenko, Petr G Leiman, Paul R Chipman, Shuji Kanamaru, Mark J van Raaij, Fumio Arisaka, Vadim V Mesyanzhinov, and Michael G Rossmann, *Three-dimensional structure of bacteriophage t4 baseplate*, Nature Structural & Molecular Biology **10** (2003), no. 9, 688–693.

[104] Michael Kremer, David Bommes, and Leif Kobbelt, *Openvolumemesh–a versatile index-based data structure for 3d polytopal complexes*, 2013, pp. 531–548.

[105] François Labelle and Jonathan Richard Shewchuk, *Isosurface stuffing: fast tetrahedral meshes with good dihedral angles*, ACM Transactions on Graphics (TOG) **26** (2007), no. 3, 57.

[106] Peter Lancaster and Kes Salkauskas, *Surfaces generated by moving least squares methods*, Mathematics of computation **37** (1981), no. 155, 141–158.

[107] Francis Lazarus, Michel Pocchiola, Gert Vegter, and Anne Verroust, *Computing a canonical polygonal schema of an orientable triangulated surface*, Proceedings of the seventeenth annual symposium on Computational geometry, ACM, 2001, pp. 80–89.

[108] Byungkook Lee and Frederic M Richards, *The interpretation of protein structures: estimation of static accessibility*, Journal of molecular biology **55** (1971), no. 3, 379–IN4.

[109] Andrew Leis, Beate Rockel, Lars Andrees, and Wolfgang Baumeister, *Visualizing cells at the nanoscale*, Trends in biochemical sciences **34** (2009), no. 2, 60–70.

[110] David Letscher and Jason Fritts, *Image segmentation using topological persistence*, Computer Analysis of Images and Patterns, Springer, 2007, pp. 587–595.

[111] Ronald M Levy, Linda Y Zhang, Emilio Gallicchio, and Anthony K Felts, *On the nonpolar hydration free energy of proteins: surface area and continuum solvent models for the solute-solvent interaction energy*, Journal of the American Chemical Society **125** (2003), no. 31, 9523–9530.

[112] Pascal Lienhardt, *Topological models for boundary representation: a comparison with n-dimensional generalized maps*, Computer-aided design **23** (1991), no. 1, 59–82.

[113] Lu Liu, Erin W Chambers, David Letscher, and Tao Ju, *A simple and robust thinning algorithm on cell complexes*, Computer Graphics Forum **29** (2010), no. 7, 2253–2260.

[114] M Lizier, J Shepherd, L Nonato, J Comba, and C Silva, *Comparing techniques for tetrahedral mesh generation*, Proceedings of the Inaugural International Conference of the Engineering Mechanics Institute, 2008.

[115] William E Lorensen and Harvey E Cline, *Marching cubes: A high resolution 3d surface construction algorithm*, ACM Siggraph Computer Graphics, vol. 21, ACM, 1987, pp. 163–169.

[116] Marius Lysaker, Arvid Lundervold, and Xue-Cheng Tai, *Noise removal using fourth-order partial differential equation with applications to medical magnetic resonance images in space and time*, Image Processing, IEEE Transactions on **12** (2003), no. 12, 1579–1590.

[117] Marian Manciu and Eli Ruckenstein, *On the chemical free energy of the electrical double layer*, Langmuir **19** (2003), no. 4, 1114–1120.

[118] Emilie Marchandise, Gaëtan Compère, Marie Willemet, Gaëtan Bricteux, Christophe Geuzaine, and J-F Remacle, *Quality meshing based on stl triangulations for biomedical simulations*, International Journal for Numerical Methods in Biomedical Engineering **26** (2010), no. 7, 876–889.

[119] Aleksandr V Marenich, Christopher J Cramer, and Donald G Truhlar, *Perspective on foundations of solvation modeling: The electrostatic contribution to the free energy of solvation*, Journal of Chemical Theory and Computation **4** (2008), no. 6, 877–887.

[120] Sean Mauch, *Closest point transform to a triangle surface*, `http://www.cacr.caltech.edu/~sean/projects/cpt/html3/`, 2004.

[121] Claudio Montani, Riccardo Scateni, and Roberto Scopigno, *A modified look-up table for implicit disambiguation of marching cubes*, The Visual Computer **10** (1994), no. 6, 353–355.

[122] Stephen P Muench, Markus Huss, Chun Feng Song, Clair Phillips, Helmut Wieczorek, John Trinick, and Michael A Harrison, *Cryo-electron microscopy of the vacuolar atpase motor reveals its mechanical and regulatory complexity*, Journal of molecular biology **386** (2009), no. 4, 989–999.

[123] Philipp Muigg, Markus Hadwiger, Helmut Doleisch, and Eduard Groller, *Interactive volume visualization of general polyhedral grids*, Visualization and Computer Graphics, IEEE Transactions on **17** (2011), no. 12, 2115–2124.

[124] David Mumford and Jayant Shah, *Optimal approximations by piecewise smooth functions and associated variational problems*, Communications on pure and applied mathematics **42** (1989), no. 5, 577–685.

[125] Elizabeth Munch, *Applications of persistent homology to time varying systems*, Ph.D. thesis, Duke University, 2013.

[126] Peter Murdoch, Steven Benzley, Ted Blacker, and Scott A Mitchell, *The spatial twist continuum: A connectivity based method for representing all-hexahedral finite element meshes*, Finite Elements in Analysis and Design **28** (1997), no. 2, 137–149.

[127] Stephan Nickell, Christine Kofler, Andrew P Leis, and Wolfgang Baumeister, *A visual approach to proteomics*, Nature reviews Molecular cell biology **7** (2006), no. 3, 225–230.

[128] Barrett O'neill, *Elementary differential geometry*, Academic press, 2006.

[129] Stanley Osher and Ronald P Fedkiw, *Level set methods: an overview and some recent results*, Journal of Computational physics **169** (2001), no. 2, 463–502.

[130] Stanley Osher and Leonid I Rudin, *Feature-oriented image enhancement using shock filters*, SIAM Journal on Numerical Analysis **27** (1990), no. 4, 919–940.

[131] Stanley Osher and James A Sethian, *Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations*, Journal of computational physics **79** (1988), no. 1, 12–49.

[132] Radosav S Pantelic, Rosalba Rothnagel, Chang-Yi Huang, David Muller, David Woolford, Michael J Landsberg, Alasdair McDowall, Bernard Pailthorpe, Paul R Young, Jasmine Banks, et al., *The discriminative bilateral filter: an enhanced denoising filter for electron microscopy data*, Journal of structural biology **155** (2006), no. 3, 395–408.

[133] Mark Pauly, Markus Gross, and Leif P Kobbelt, *Efficient simplification of point-sampled surfaces*, Proceedings of the conference on Visualization'02, IEEE Computer Society, 2002, pp. 163–170.

[134] Pietro Perona and Jitendra Malik, *Scale-space and edge detection using anisotropic diffusion*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **12** (1990), no. 7, 629–639.

[135] Ninad V Prabhu, Peijuan Zhu, and Kim A Sharp, *Implementation and testing of stable, fast implicit solvation in molecular dynamics using the smooth-permittivity finite difference poisson–boltzmann method*, Journal of computational chemistry **25** (2004), no. 16, 2049–2064.

[136] Sylvain Prat, Patrick Gioia, Yves Bertrand, and Daniel Meneveaux, *Connectivity compression in an arbitrary dimension*, The Visual Computer **21** (2005), no. 8-10, 876–885.

[137] Frederic M Richards, *Areas, volumes, packing, and protein structure*, Annual Review of Biophysics and Bioengineering **6** (1977), no. 1, 151–176.

[138] Vanessa Robins, *Towards computing homology from finite approximations*, Topology Proceedings, vol. 24, 1999, pp. 503–532.

[139] Carol V Robinson, Andrej Sali, and Wolfgang Baumeister, *The molecular sociology of the cell*, Nature **450** (2007), no. 7172, 973–982.

[140] Walter Rocchia, Sundaram Sridharan, Anthony Nicholls, Emil Alexov, Alessandro Chiabrera, and Barry Honig, *Rapid grid-based construction of the molecular surface and the use of induced surface charge to calculate reaction field energies: Applications to the molecular systems and geometric objects*, Journal of computational chemistry **23** (2002), no. 1, 128–137.

[141] Benoıt Roux and Thomas Simonson, *Implicit solvent models*, Biophysical Chemistry **78** (1999), no. 1, 1–20.

[142] Leonid I Rudin, Stanley Osher, and Emad Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D: Nonlinear Phenomena **60** (1992), no. 1, 259–268.

[143] Prihambodo H Saksono, Perumal Nithiarasu, and Igor Sazonov, *Numerical prediction of heat transfer patterns in a subject-specific human upper airway*, Journal of Heat Transfer **134** (2012), no. 3, 031022.

[144] Michel F Sanner, Arthur J Olson, and Jean-Claude Spehner, *Reduced surface: an efficient way to compute molecular surfaces*, Biopolymers **38** (1996), no. 3, 305–320.

[145] Guillermo Sapiro and Dario L Ringach, *Anisotropic diffusion of multivalued images with applications to color filtering*, Image Processing, IEEE Transactions on **5** (1996), no. 11, 1582–1586.

[146] Igor Sazonov and Perumal Nithiarasu, *Semi-automatic surface and volume mesh generation for subject-specific biomedical geometries*, International Journal for Numerical Methods in Biomedical Engineering **28** (2012), no. 1, 133–157.

[147] Igor Sazonov, Si Yong Yeo, Rhodri LT Bevan, Xianghua Xie, Raoul van Loon, and Perumal Nithiarasu, *Modelling pipeline for subject-specific arterial blood flow—a review*, International Journal for Numerical Methods in Biomedical Engineering **27** (2011), no. 12, 1868–1910.

[148] Joachim Schöberl, *Netgen an advancing front 2d/3d-mesh generator based on abstract rules*, Computing and visualization in science **1** (1997), no. 1, 41–52.

[149] S Pena Serna, A Stork, and DW Fellner, *Considerations toward a dynamic mesh data structure*, SIGRAD Conference, 2011, pp. 83–90.

[150] Chemical Abstracts Service, *Cas registry and cas registry number faqs*, `http://www.cas.org/content/chemical-substances/faqs`, 2012.

[151] James A Sethian, *Evolution, implementation, and application of level set and fast marching methods for advancing fronts*, Journal of Computational Physics **169** (2001), no. 2, 503–555.

[152] James Albert Sethian, *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, vol. 3, Cambridge university press, 1999.

[153] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or, *Consistent mesh partitioning and skeletonisation using the shape diameter function*, The Visual Computer **24** (2008), no. 4, 249–259.

[154] Kim A Sharp and Barry Honig, *Calculating total electrostatic energies with the nonlinear poisson-boltzmann equation*, Journal of Physical Chemistry **94** (1990), no. 19, 7684–7692.

[155] _____, *Electrostatic interactions in macromolecules: theory and applications*, Annual review of biophysics and biophysical chemistry **19** (1990), no. 1, 301–332.

[156] Jonathan R Shewchuk, *Delaunay refinement mesh generation*, Tech. report, DTIC Document, 1997.

[157] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser, *The princeton shape benchmark*, Shape Modeling Applications, 2004. Proceedings, IEEE, 2004, pp. 167–178.

[158] Hang Si, *Constrained delaunay tetrahedral mesh generation and refinement*, Finite elements in Analysis and Design **46** (2010), no. 1, 33–46.

[159] Daniel Sieger and Mario Botsch, *Design, implementation, and evaluation of the surface_mesh data structure*, Proceedings of the 20th International Meshing Roundtable, Springer, 2012, pp. 533–550.

[160] Thomas Simonson, *Macromolecular electrostatics: continuum models and their growing pains*, Current opinion in structural biology **11** (2001), no. 2, 243–252.

[161] Nir Sochen, Ron Kimmel, and Ravi Malladi, *A general framework for low level vision*, Image Processing, IEEE Transactions on **7** (1998), no. 3, 310–318.

[162] Octavian Soldea, Gershon Elber, and Ehud Rivlin, *Global segmentation and curvature analysis of volumetric data sets using trivariate b-spline functions*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **28** (2006), no. 2, 265–278.

[163] Hamid Soltanian-Zadeh, Joe P Windham, and Andrew E Yagle, *A multidimensional nonlinear edge-preserving filter for magnetic resonance image restoration*, Image Processing, IEEE Transactions on **4** (1995), no. 2, 147–161.

[164] Arne Stoschek and Reiner Hegerl, *Denoising of electron tomographic reconstructions using multiscale transformations*, Journal of structural biology **120** (1997), no. 3, 257–265.

[165] YUHUI SUN, PEIRU WU, GW WEI, and GE WANG, *Evolution-operator-based single-step method for image processing*, International Journal of Biomedical Imaging (2006), no. 1, 1–27.

[166] Tolga Tasdizen, Ross Whitaker, Paul Burchard, and Stanley Osher, *Geometric surface processing via normal maps*, ACM Transactions on Graphics (TOG) **22** (2003), no. 4, 1012–1033.

[167] Timothy J Tautges, Ted Blacker, and Scott A Mitchell, *The whisker weaving algorithm: A connectivity-based method for constructing all-hexahedral finite element meshes*, International Journal for Numerical Methods in Engineering **39** (1996), no. 19, 3327–3349.

[168] IGG Team, *Combinatorial and geometric modeling with generic n-dimensional maps*, `http://cgogn.u-strasbg.fr/Wiki/index.php/CGoGN`, 2012.

[169] Elitza I Tocheva, Zhuo Li, and Grant J Jensen, *Electron cryotomography*, Cold Spring Harbor perspectives in biology **2** (2010), no. 6, a003442.

[170] Carlo Tomasi and Roberto Manduchi, *Bilateral filtering for gray and color images*, Computer Vision, 1998. Sixth International Conference on, IEEE, 1998, pp. 839–846.

[171] Jane Tournois, Camille Wormser, Pierre Alliez, and Mathieu Desbrun, *Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation*, ACM Transactions on Graphics **28** (2009), no. 3, Art–No.

[172] Graham M Treece, Richard W Prager, and Andrew H Gee, *Regularised marching tetrahedra: improved iso-surface extraction*, Computers & Graphics **23** (1999), no. 4, 583–598.

[173] Peter van der Heide, Xiao-Ping Xu, Brad J Marsh, Dorit Hanein, and Niels Volkmann, *Efficient automatic noise reduction of electron tomographic reconstructions based on iterative median filtering*, Journal of structural biology **158** (2007), no. 2, 196–204.

[174] Jarke J Van Wijk and Arjeh M Cohen, *Visualization of seifert surfaces*, Visualization and Computer Graphics, IEEE Transactions on **12** (2006), no. 4, 485–496.

[175] Niels Volkmann, *Chapter two-methods for segmentation and interpretation of electron tomographic reconstructions*, Methods in enzymology **483** (2010), 31–46.

[176] George Vosselman, Sander Dijkman, et al., *3d building model reconstruction from point clouds and ground plans*, International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences **34** (2001), no. 3/W4, 37–44.

[177] Min Wan, Yu Wang, and Desheng Wang, *Variational surface reconstruction based on delaunay triangulation and graph cut*, International journal for numerical methods in engineering **85** (2011), no. 2, 206–229.

[178] Hong-Wei Wang and Eva Nogales, *Nucleotide-dependent bending flexibility of tubulin regulates microtubule assembly*, Nature **435** (2005), no. 7044, 911–915.

[179] Yang Wang, Guo-Wei Wei, and Siyang Yang, *Partial differential equation transform—variational formulation and fourier analysis*, International journal for numerical methods in biomedical engineering **27** (2011), no. 12, 1996–2020.

[180] _____, *Mode decomposition evolution equations*, Journal of scientific computing **50** (2012), no. 3, 495–518.

[181] Yuanzhen Wang, Beibei Liu, and Y Tong, *Linear surface reconstruction from discrete fundamental forms on triangle meshes*, Computer Graphics Forum **31** (2012), no. 8, 2277–2287.

[182] Nigel P Weatherill and Oubay Hassan, *Efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary constraints*, International Journal for Numerical Methods in Engineering **37** (1994), no. 12, 2005–2039.

[183] G. W. Wei, Y. H. Sun, Y. C. Zhou, and M. Feig, *Molecular multiresolution surfaces*, arXiv:math-ph/0511001v1 (2005), 1 – 11.

[184] Guo W Wei, *Generalized perona-malik equation for image restoration*, Signal Processing Letters, IEEE **6** (1999), no. 7, 165–167.

[185] _____, *Generalized perona-malik equation for image restoration*, Signal Processing Letters, IEEE **6** (1999), no. 7, 165–167.

[186] Guo-Wei Wei, *Differential geometry based multiscale models*, Bulletin of mathematical biology **72** (2010), no. 6, 1562–1622.

[187] Guo-Wei Wei, Qiong Zheng, Zhan Chen, and Kelin Xia, *Variational multiscale models for charge transport*, SIAM Review **54** (2012), no. 4, 699–754.

[188] GW Wei and YQ Jia, *Synchronization-based image edge detection*, EPL (Europhysics Letters) **59** (2002), no. 6, 814.

[189] Thomas P Witelski and Mark Bowen, *Adi schemes for higher-order nonlinear diffusion equations*, Applied Numerical Mathematics **45** (2003), no. 2, 331–351.

[190] Andrew P Witkin, *Scale-space filtering: A new approach to multi-scale description*, Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84., vol. 9, IEEE, 1984, pp. 150–153.

[191] Zoë J Wood, Hugues Hoppe, Mathieu Desbrun, and Peter Shröder, *Removing excess topology from isosurfaces*, ACM Transactions on graphics **23** (2004), no. 2, 190–208.

[192] Chunlin Wu and Xuecheng Tai, *A level set formulation of geodesic curvature flow on simplicial surfaces*, Visualization and Computer Graphics, IEEE Transactions on **16** (2010), no. 4, 647–662.

[193] Muh-Cherng Wu and CR Lit, *Analysis on machined feature recognition techniques based on b-rep*, Computer-Aided Design **28** (1996), no. 8, 603–616.

[194] Kelin Xia, Xin Feng, Zhan Chen, Yiying Tong, and Guo-Wei Wei, *Multiscale geometric modeling of macromolecules i: Cartesian representation*, Journal of computational physics **257** (2014), 912–936.

[195] Kelin Xia and Guo-Wei Wei, *Molecular nonlinear dynamics and protein thermal uncertainty quantification*, Chaos: An Interdisciplinary Journal of Nonlinear Science **24** (2014), no. 1, 013103.

[196] Ye Xiang, Marc C Morais, Anthony J Battisti, Shelley Grimes, Paul J Jardine, Dwight L Anderson, and Michael G Rossmann, *Structural changes of bacteriophage ϕ29 upon dna packaging and release*, The EMBO journal **25** (2006), no. 21, 5229–5239.

[197] Dexuan Xie, Yi Jiang, Peter Brune, and L Ridgway Scott, *A fast solver for a nonlocal dielectric continuum model*, SIAM Journal on Scientific Computing **34** (2012), no. 2, B107–B126.

[198] Shi-Qing Xin, Ying He, and Chi-Wing Fu, *Efficiently computing exact geodesic loops within finite steps*, Visualization and Computer Graphics, IEEE Transactions on **18** (2012), no. 6, 879–889.

[199] Y-L You and Mostafa Kaveh, *Fourth-order partial differential equations for noise removal*, Image Processing, IEEE Transactions on **9** (2000), no. 10, 1723–1730.

[200] Sining Yu, Weihua Geng, and GW Wei, *Treatment of geometric singularities in implicit solvent models*, The Journal of chemical physics **126** (2007), no. 24, 244108.

[201] Sining Yu and GW Wei, *Three-dimensional matched interface and boundary (mib) method for treating geometric singularities*, Journal of Computational Physics **227** (2007), no. 1, 602–632.

[202] Zeyun Yu, Michael J Holst, Yuhui Cheng, and J Andrew McCammon, *Feature-preserving adaptive mesh generation for molecular shape modeling and simulation*, Journal of Molecular Graphics and Modelling **26** (2008), no. 8, 1370–1380.

[203] Zeyun Yu, Michael J Holst, Takeharu Hayashi, Chandrajit L Bajaj, Mark H Ellisman, J Andrew McCammon, and Masahiko Hoshijima, *Three-dimensional geometric modeling of membrane-bound organelles in ventricular myocytes: bridging the gap between microscopic imaging and mathematical simulation*, Journal of structural biology **164** (2008), no. 3, 304–313.

[204] Eugene Zhang, Konstantin Mischaikow, and Greg Turk, *Feature-based surface parameterization and texture mapping*, ACM Transactions on Graphics (TOG) **24** (2005), no. 1, 1–27.

[205] Guo-Xin Zhang, Song-Pei Du, Yu-Kun Lai, Tianyun Ni, and Shi-Min Hu, *Sketch guided solid texturing*, Graphical Models **73** (2011), no. 3, 59–73.

[206] Yongjie Zhang, Chandrajit Bajaj, and Guoliang Xu, *Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow*, Communications in Numerical Methods in Engineering **25** (2009), no. 1, 1–18.

[207] Shan Zhao, *High order matched interface and boundary methods for the helmholtz equation in media with arbitrarily curved interfaces*, Journal of Computational Physics **229** (2010), no. 9, 3155–3170.

[208] _____, *Pseudo-time-coupled nonlinear models for biomolecular surface representation and solvation analysis*, International Journal for Numerical Methods in Biomedical Engineering **27** (2011), no. 12, 1964–1981.

[209] Xin Zhao, Bo Li, Lei Wang, and Arie Kaufman, *Texture-guided volumetric deformation and visualization using 3d moving least squares*, The Visual Computer **28** (2012), no. 2, 193–204.

[210] Q. Zheng, S. Y. Yang, and G. W. Wei, *Molecular surface generation using pde transform*, International Journal for Numerical Methods in Biomedical Engineering **28** (2012), 291–316.

[211] Qiong Zheng, Duan Chen, and Guo-Wei Wei, *Second-order poisson–nernst–planck solver for ion transport*, Journal of computational physics **230** (2011), no. 13, 5239–5262.

[212] Qiong Zheng and Guo-Wei Wei, *Poisson–boltzmann–nernst–planck model*, The Journal of chemical physics **134** (2011), no. 19, 194101.

[213] YC Zhou and GW Wei, *On the fictitious-domain and interpolation formulations of the matched interface and boundary (mib) method*, Journal of Computational Physics **219** (2006), no. 1, 228–246.

[214] YC Zhou, Shan Zhao, Michael Feig, and GW Wei, *High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources*, Journal of Computational Physics **213** (2006), no. 1, 1–30.