TOWARDS HIGH QUALITY HEXAHEDRAL MESHES: GENERATION, OPTIMIZATION, AND EVALUATION

A Dissertation

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

By

Xifeng Gao

May 2016

TOWARDS HIGH QUALITY HEXAHEDRAL MESHES: GENERATION, OPTIMIZATION, AND EVALUATION

Xifeng Gao

APPROVED:

Guoning Chen, Co-chairman
Dept. of Computer Science, University of Houston

Zhigang Deng, Co-chairman
Dept. of Computer Science, University of Houston

Jin Huang
Dept. of Computer Science, Zhejiang University

Pandurangan Gopal
Dept. of Computer Science, University of Houston

Stephen Huang
Dept. of Computer Science, University of Houston

Dean, College of Natural Sciences and Mathematics

Acknowledgments

While I am going to accomplish my Ph.D. study in a few weeks, during the time of writing my Ph.D. dissertation, I am feel enormous gratitude in my heart to the following people: my advisors, my colleagues, my friends, and my families.

First and foremost, I want to thank my two advisors: Prof. Guoning Chen and Prof. Zhigang Deng. Without their funding support, I could not finish or even start my Ph.D. study. They provided distinctive scientific guidance, while allowing me great freedom in pursuing my research interests. I hold great thanks to Prof. Chen's expertise during our numerous discussions of technical details, research ideas, and directions. He has charming personalities and is always very kind in giving me tons of advices and encouragements. I learn a lot from Prof. Deng's quick response and decisive manner in handling many of my research related issues. I also appreciate his strict yet lenient advising style in driving me to overcome my laziness and many other shortcomings. Both Prof. Chen's and Prof. Deng's rigorous attitude in doing research and great characteristics set role models for me. I cherish the times spent with them forever.

I would like to sincerely thank Prof. Wenping Wang as well. It's a fantastic experience for me to work as an intern in Prof. Wang's lab at the Hong Kong

University. Without his support, I will never have the memory of having had a visit to one of the best groups I ever stayed with. I learn quite a lot from Prof. Wang, not only from his style of doing research, but also his great personalities and characteristics.

I think I was lucky to know Prof. Jin Huang during my third year of Ph.D. study. I highly appreciate Prof. Huang's insights and invaluable involvement in our collaborated hex-mesh evaluation project. He is one of the smartest researchers I have ever met. He has a very sharp mind and strong critical thinking. I gain much about doing research and problem formulations every time I talk to him.

I would like to express my gratitude to Dr. Tobias Martin, together with Prof. Chen, who brought me the first exciting hex-meshing project and inspired my interests in geometry area. Our collaborations have led to multiple research problems and I am expecting to collaborate with him in the near future.

I would like to thank Prof. Nikolaos Tsekos, together with Dr. Nikhil Navkar and Prof. Deng, who introduced me into the medical imaging field. For the first time, I had the opportunity to do MR scanning of my own heart for my MR image real-time registration project. I thank Prof. Elaine Cohen and my collaborator Sai Deng for their involvement in my first hex-meshing project and contributions in the corner extraction and placement for the 2D skeletal structure construction.

I would like to thank my Ph.D. committee members: Prof. Jin Huang, Prof. Stephen Huang, Prof. Gopal Pandurangan, and my two advisors, for their time, efforts, and helpful comments.

I would also like to thank my colleagues: Nikhil Navkar, Xiaohan Ma, Binh Le,

Mario Rincon, Mingyang Zhu, Xiao Chen, Cheng Chen, and Li Wei from CGIM lab at the University of Houston; Lei Zhang, Kaoji Xu, Lieyu Shi, and Marzieh Berenjkoub from DAVIM group at the University of Houston; Siwang Li, and Tengfei Jiang from the State Key CAD&CG lab at the Zhejiang University; Zherong Pan from the GAMMA group at the University of North Carolina at Chapel Hill; Shuiqing He, Yujing Sun, Weikai Chen, Yating Yue, Xiaolong Zhang, Changjian Li, Lingjie Liu, Hui Zhang, Yuexin Ma, and Deng O from computer graphics lab at the Hong Kong University.

Finally, I would like to express my sincerest gratitude to my wife for her love, support, and accompany since the time of my graduate study in Shandong University. I would also like to give my deepest gratitude to my parents for their continuous support and selfless love in my whole life.

TOWARDS HIGH QUALITY HEXAHEDRAL MESHES: GENERATION, OPTIMIZATION, AND EVALUATION

An Abstract of a Dissertation

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

By

Xifeng Gao

May 2016

Abstract

Hexahedral meshes are a preferred volumetric representation in a wide range of scientific and engineering applications that require solving partial differential equations (PDEs) and fitting tensor product/trivariate splines, such as mechanical analysis, kinematic and dynamic analysis of mechanisms, bio-mechanical engineering, computational fluid dynamics, and physically-based simulations. Recently, the generation of a high quality all-hex mesh of a given volume has gained much attention, where a hex-mesh should have high surface conformity, regular element shapes, and simple global structure. This dissertation investigates the problem of obtaining a high quality hex-mesh with respect to the above quality requirements and makes the following contributions:

Firstly, I introduce a volumetric partitioning strategy based on a generalized sweeping framework to seamlessly partition the volume enclosed by an input triangle mesh into a small number of deformed cube-like components. This is achieved by a user-designed volumetric harmonic function that guides the decomposition of the input volume into a skeletal structure aligning with features of the input object. This pipeline has been applied to a variety of 3D objects to demonstrate its utility.

Secondly, I present a first and complete pipeline to reduce the complexity of the global structure of an input hex-mesh by aligning mis-matched singularities. Specifically, I first remove redundant cube-like components to reduce the complexity of the structure while maintaining singularities unchanged, and then perform a structure-aware optimization to improve the geometric fidelity of the resulting hex-mesh.

Thirdly, I propose the first practical framework to simplify the global structure of any valid all-hex meshes. My simplification was achieved by procedurally removing

base complex sheets and *base complex chords* that constitute the base complex of a hex-mesh. To maintain the surface geometric features, I introduced a parameterization based collapsing strategy for the removal operations. Given a user-specified level of simplicity, I identified the inversion-free hex-mesh with the optimal simplified structure using a binary search strategy from the obtained all-hex structure hierarchy.

Finally, given that there currently does not exist a widely accepted guideline for the selection of proper element quality metrics for hex-meshes, I performed the first comprehensive study on the correlation among available quality metrics for hex-meshes. My analysis first computed the linear correlation coefficients between pairs of metrics. Then, the most relevant metrics were identified for three selected applications – the linear elasticity, Poisson, and Stoke problems, respectively. To address the need of a large set of sampled meshes well-distributed in the metric space, I proposed a two-level noise insertion strategy. Results of this work can be used as preliminary yet practical guidelines for the development of effective hex-mesh generation and optimization techniques.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In a variety of scientific and engineering applications that require solving partial differential equations and fitting tensor product/high-order smoothness functions, such as mechanical analysis, kinematic and dynamic analysis of mechanisms, biomechanical engineering, computational fluid dynamics, and physically-based simulations, volumetric representations (e.g., tetrahedral/hexahedral/hybrid/T- meshes) are often required. There are automatic and open source libraries for producing good quality tet-meshes. While tetrahedral-mesh representation is simple and linear, (semi-) structured representations, e.g., hexahedral meshes, are often preferred over unstructured representations [71] because of the better convergence properties [14], less numerical stiffness, and more space efficiency [90]. The tensor product nature of a structured representation allows the convenient imposition of a simulation basis with a higher derivative smoothness between elements of the volumetric mesh.

This means that each individual basis function spans smoothly across multiple elements with visually smoother results. Finite element representations, which possess these properties can yield better numerical results for a variety of applications in engineering (e.g., see Ringleb-flow simulation in [41]).

Automatically decomposing a 3D volume domain with complex boundary into a high-quality hex-mesh remains a challenging task. A high-quality hex-mesh usually has low geometric distortion (a benefit for stable numerical simulations) and a simple global structure (ideal for high-order volumetric B-spline-fitting). Geometric distortion measures the preservation of the boundary of the 3D volume and the shape similarity of each contained hexahedral element to a cube. While Hausdorff distance can be employed to quantitatively measure the conformity to the surface features [75], it is generally acknowledged that there should be non-inverted elements in a hex-mesh, i.e., the Jacobian determinants for all the corners of all the elements should be positive. On the other hand, the global structure of a hex-mesh can be obtained by merging neighboring elements without crossing extraordinary (or irregular) edges (i.e., edges that have other than 4 hex-elements adjacent to it) and the separation surfaces starting from these edges. These irregular edges are singularities. Intuitively, the singularities as well as the separation structure starting from them partition the domain into smaller regions. Figure 1.1 provides examples of such partitions for the bunny hex-meshes. Each sub-volume, referred to as a component (i.e., the colored regions shown in Figure 1.1) of a partition can be topologically mapped to a regular cube. These components compose a global structure, which is referred to as the *base-complex* in this dissertation. The base-complex is the coarsest version

of the hex-mesh, which should be as simple as possible (i.e., with as few components as possible) to facilitate the subsequent spline-fitting.

Consequently, this dissertation focuses on pushing the hex-meshing research forward from two aspects. On the one hand, while the majority of techniques in hex-meshing focus on improving the surface geometric fidelity and minimizing the irregularity of element shapes of the resultant hex-meshes, I endeavor to improve the quality of the global structure of the hex-meshes. On the other hand, although maintaining a positive minimum Jacobian is necessary for most applications, to what degree that scaled Jacobian provides an effective indication to the quality of the subsequent applications has largely been understudied. Especially considering that there are nearly 40 quality metrics for hex-meshes, only a few metrics, including scaled Jacobian, are commonly used by hex-mesh generation techniques. I perform a first comprehensive evaluation of existing quality metrics in the hope to discover a general guidance for the development of hex-mesh generation and optimization techniques. Details of our contributions in these two directions are summarized as follows.

1.1 Hex-meshing with Simple Structure

Large number of components may pose challenges to the subsequent computations on the corresponding hex-meshes, such as high order spline-fitting [18] or using higher order smoothness simulation bases. This is because in applications that use higher order representations, a C^2 B-spline basis can be fitted to each component (not each element). However, in general, only C^0 continuity can be achieved across the

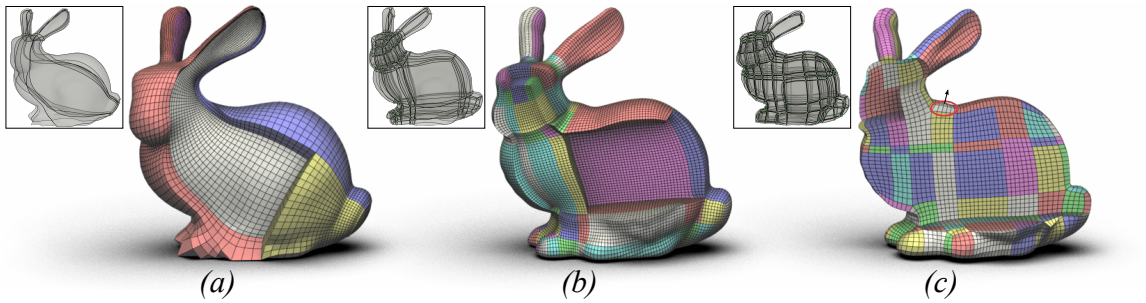


Figure 1.1: The hex-mesh in (a) has only 18 hexahedral components, while the hex-meshes in (b) and (c) have 259 and 422 components, respectively. The corresponding base-complexes of these meshes are shown in the inset images, respectively. Green dots are the corners of the individual components and the black lines are their edges. Note that some of the cuboid corners are extraordinary points, others are regular and introduced to avoid T-junctions.

boundaries of neighboring components. More components means that the extent of each C^2 continuous region is diminished. In addition, for a hex-mesh with too many components, it may contain some quite small or narrow components. Each of the small components will need to be subdivided multiple times (e.g., see a component near the boundary highlighted by an ellipse in Figure 1.1c) to get enough number of samples along each of the three parameterization directions, so that the higher-order basis function can be fitted. However, doing so will mean that its neighboring components have to be subdivided accordingly, leading to a very fine mesh with many elements if T-junctions should be avoided. This up-sampling process increases the number of elements in multiples of the original size of the hex-mesh, which makes a fast computation more difficult to achieve. Therefore, fewer components (i.e., a simpler base-complex) are desired for the task of spline-fitting.

After introducing the global structure concepts for a hex-mesh, i.e., *singularity-structure* and *base-complex*, in Chapter 2, as the first attempt, this dissertation makes

three contributions [30, 27, 31] in obtaining a hex-mesh with simple global structure while maintaining its geometric quality (to some extend).

First, in Chapter 4, I introduce a volumetric-partitioning strategy based on a generalized sweeping framework to seamlessly partition the volume of an input triangle mesh into a collection of deformed cuboids [30]. This is achieved by a user-designed volumetric-harmonic function that guides the decomposition of the input volume into a sequence of 2-manifold-level sets. A skeletal structure whose corners correspond to corner vertices of a 2D parameterization is extracted for each level set. Corners are placed so that the skeletal structure aligns with features of the input object. Then, a skeletal surface is constructed by matching the skeletal structures of adjacent-level sets. The surface sheets of this skeletal surface partition the input volume into the deformed cuboids. The collection of cuboids does not exhibit T-junctions, significantly simplifying the hexahedral mesh generation process, and in particular, it simplifies fitting trivariate B-splines to the deformed cuboids. Intersections of the surface sheets of the skeletal surface correspond to the singular edges of the generated hex-meshes. I applied our technique to a variety of 3D objects and demonstrate the benefit of the structure decomposition in data fitting.

Second, in Chapter 5, I present a first and complete pipeline to optimize the global structure of a hex-mesh [27]. Specifically, I first extract the *base-complex* of a hex-mesh and study the misalignments among its singularities by adapting the previously introduced hexahedral sheets to the base-complex. Second, I identify the valid removal *base-complex sheets* from the base-complex that contain misaligned singularities. I then propose an effective algorithm to remove these valid removal sheets

in order. Finally, I present a structure-aware optimization strategy to improve the geometric quality of the resulting hex-mesh after fixing the misalignments. Our experimental results demonstrate that our pipeline can significantly reduce the number of components of a variety of hex-meshes generated by state-of-the-art methods, while maintaining high geometric quality.

Third, in Chapter 6, I introduce an effective framework to simplify the global structure (i.e., the base-complex) of valid all-hex-meshes [31]. Our simplification was achieved by decomposing the base-complex of an input hex-mesh into a number of *base-complex sheets* and *base-complex chords*. By collapsing them, the structure was simplified. I provide detailed discussions on the configurations of sheets and chords, upon which different removal operations are proposed. I also propose a ranking metric to sort the sheets and chords to achieve efficient reduction of the structure complexity while preserving important surface features. To reduce the geometric distortion induced by simplification, I introduced a parameterization-based collapsing strategy for the removal operations. Given a user-specified level of simplicity, I choose the optimal simplified structure of a hex-mesh without a negative Jacobian using a binary search strategy from the obtained all-hex structure hierarchy. I have applied our simplification framework to a number of hex-meshes generated by state-of-the-arts hex-meshing techniques to demonstrate its effectiveness.

1.2 Element Quality Metrics Evaluation

Compared to the well-understood quality metrics for tet-meshes and their relation to the conditioning of finite element stiffness matrices and the accuracy of interpolation functions [81], most hex-meshing approaches often rely on the *average and minimal scaled Jacobian* metrics to measure the quality of the generated hex-meshes. That is, the meshes with larger scaled Jacobians are considered better. Although maintaining a positive minimum Jacobian is necessary for most applications, to what degree that scaled Jacobian provides an effective indication to the quality of the subsequent applications carried out on the corresponding hex-meshes has largely been understudied. Does a hex-mesh with good Jacobians (especially a high minimum-scaled Jacobian) always lead to accurate and stable computations for applications that involve with the solving of elliptical PDEs (e.g., linear-elasticity, Poisson and Stoke problems)? Figure 1.2 provides an example that the quality of the Poisson equation solving measured by its largest eigenvalue–indicator of the stability of the system (see Section 7.2.2.4 for more details), need not be improved with the improving minimum-scaled Jacobian. In addition to the scaled Jacobian, there are many other quality metrics [86] (see Table 3.1). However, there does not exist a guideline for the selection of proper metrics for the effective measurement of hex-mesh quality in practice. This has hampered the development of effective hex-mesh generation and optimization techniques [47] that can produce hex-meshes with properties suitable for specific downstream applications. This motivates our work presented in Chapter 7 [28, 29]. To address the above challenge, I conducted a first comprehensive study on the correlation among various hex-mesh quality metrics. I demonstrated that our analysis

framework can be effectively applied to reduce the number of quality metrics and identify the most reliable metric given a specific application. I applied the proposed analysis techniques to three different applications – the linear-elasticity problem, Poisson's equation-solving and Stoke equation-solving, respectively, on which I conducted a first study to understand the effectiveness of all existing hex-mesh quality metrics. I observed that average metrics greatly affect the accuracy of those applications, while minimum and maximum metrics have more influences on the stability of these applications. To the best of our knowledge, this work is the first step to quantitatively understand the correlation characteristics of a large number of existing hex-mesh quality metrics and their effectiveness to downstream applications. The encouraging results from our study can be used as practical guidelines for the development of effective hex-mesh generation and optimization techniques as well as the cornerstone for future research studies along this line.

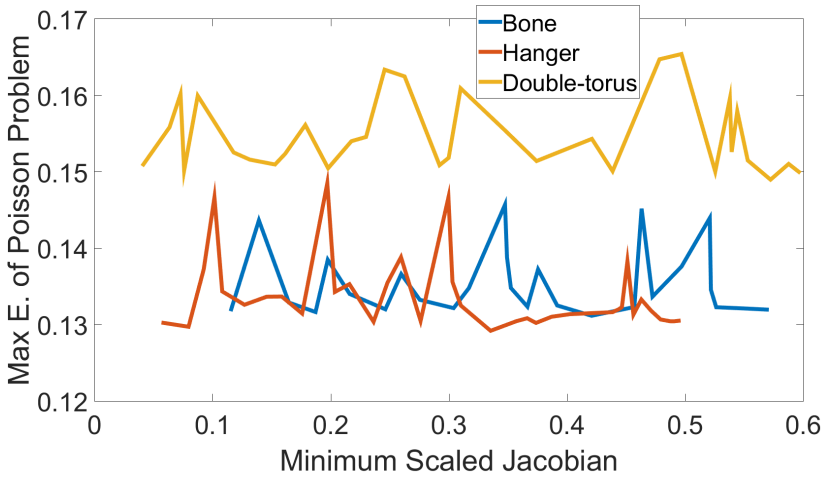


Figure 1.2: Minimum scaled Jacobian is not a good indicator for the convergence rate of Poisson equation solving, as the maximal eigenvalues of the systems do not decrease with the improvement of the minimum-scaled Jacobian as seen for all three test models.

Chapter 2

Global Structures of Hex-meshes

Consider a hex-mesh $\mathcal{H} = (V, E, F, H)$, where V is a set of vertices, E is a set of edges, F is a set of faces, and H is a set of hexahedral elements. Assume $\mathcal{H}$ is a valid manifold all-hex-mesh, where each edge has a neighborhood that is homeomorphic to a cylinder or a half-cylinder, and each vertex has a neighborhood that is homeomorphic to a sphere or a half-sphere. Edges with half-cylinder neighborhoods and vertices with half-sphere neighborhoods are on the boundary. The boundary of a manifold hex-mesh is a closed two-manifold mesh. Throughout the dissertation, I define the valence of a vertex or an edge with respect to the number of its neighboring hexahedral elements. An edge is said to be *irregular* if its valence is not 2 (on the boundary) or not 4 (in the interior). A vertex is called regular if its valence is 4 (on the boundary) or 8 (in the interior); otherwise, it is an irregular vertex.

In the following, I first review the singularity-structure of hex-meshes. Then, I extend the base-complex concept from 2D quad-meshes to 3D hex-meshes, and

introduce a robust algorithm to extract it.

2.1 singularity-structure

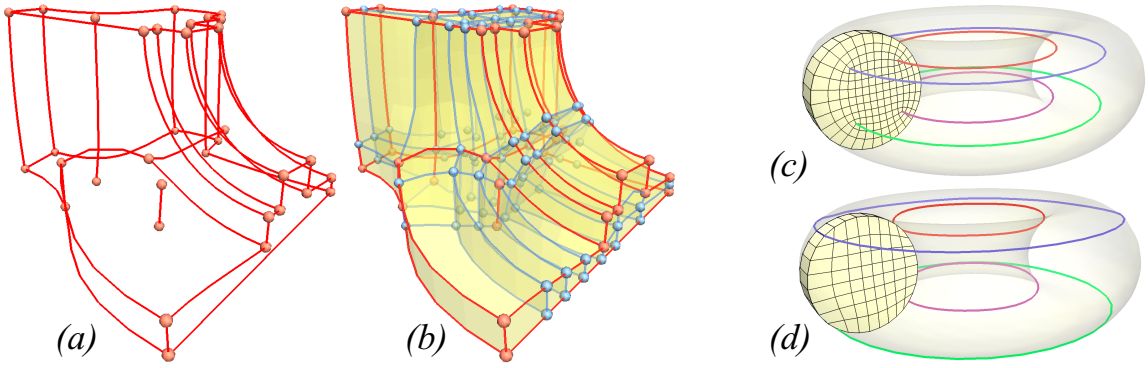


Figure 2.1: The singularity-structure (a) and base-complex (b) of a Fandisk hex-mesh. The singular nodes and singular edges of both the singularity-structure and the base-complex are in red, while the non-singular nodes and regular edges of the base-complex are in blue. (c) shows four closed singular edges in the interior of a Torus hex-mesh, while (d) shows four closed singular edges on the boundary.

The singularity-structure of $\mathcal{H}$, denoted by $\mathcal{S}$, consists of a set of singular nodes and singular edges. A singular edge is a 1D curve composed of a sequence of connected irregular edges with the same valence, either in the interior (Figure 2.1(c)) or on the boundary (Figure 2.1(d)) of the hex-mesh. The singular edges can be classified into two types: open or closed (Figure 2.1(c-d)). Specifically, the end points of an open singular edge are singular nodes (i.e., the intersections of some singular edges or their intersections with the boundary). It cannot simply start or end in the interior of the volume; instead, it either hits the boundary or connects with other open singular edges via a singular node. By contrast, no singular node exists on a closed singular edge, and it is completely either on the boundary or contained in the

volume. Note that a singular node can be either regular or irregular.

The above definition of the singularity-structure also self-describes an automatic algorithm to identify it. Figure 2.1(a) shows an example of the singularity-structure of a Fandisk hex-mesh.

2.2 Base-complex

After extracting the singularity-structure, I now describe the definition and computation of the base-complex of a hex-mesh. Analogous to the base-complex of a quad-mesh [8], the base-complex of a hex-mesh is an all-hexahedral structure.

Similar to the curve separatrices in 2D [89], which consist of edges in a quad-mesh, the surface-separation structures starting from any singular edge are needed to define the base-complex of a hex-mesh. I refer to these surface-separation structures as the *separation surfaces*. Each separation surface consists of a set of connected quads in the hex-mesh. For a singular edge with valence n, there are n separation surfaces originating from it. All the separation surfaces from singular edges form a surface graph network embedded in the hex-mesh, describing its topological structure.

I denote the base-complex of a hex-mesh $\mathcal{H}$ as $\mathcal{B} = (\mathcal{B}_V, \mathcal{B}_E, \mathcal{B}_F, \mathcal{B}_C)$ (Figure 2.1(b)). $\mathcal{B}_E$ is the set of the intersections of the separation surfaces. Each base-complex edge in $\mathcal{B}_E$, either singular or regular, consists of a sequence of connected hex edges of $\mathcal{H}$. The singular base-complex edges in $\mathcal{B}$ are illustrated as red curves in Figure 2.1(b), while the regular edges are illustrated as blue curves. $\mathcal{B}_V$ is

the set of the end points of the base-complex edges in $\mathcal{B}_E$, which are either singular (red nodes in Figure 2.1(b)) or non-singular (blue nodes in Figure 2.1(b)). The reason to include non-singular nodes and regular base-complex edges is to remove T-junction configurations. $\mathcal{B}_F$ is a set of base-complex faces, each of which has four edges in $\mathcal{B}_E$. $\mathcal{B}_F$ partitions the domain of $\mathcal{H}$ into a set of components in $\mathcal{B}_C$. Each component is a cuboid-like sub-volume.

Algorithm 1 describes the pseudo code for extracting the base-complex $\mathcal{B}$ from a given hex-mesh $\mathcal{H}$. It is important to note that our defined base-complex is distinct from the one mentioned in [55] which is actually a polycube structure.

Algorithm 1: Pseudo code of extracting $\mathcal{B}$ from $\mathcal{H}$

Input : $\mathcal{H}$
Output: $\mathcal{B}$
Extract $\mathcal{S}$ from $\mathcal{H}$;
foreach *singular edge* se_i *of* $\mathcal{S}$ **do**
    if se_i *is open* **then**
        Trace n separatrices for each of the two singular nodes of se_i, where n is the valence of se_i;
    if se_i *is closed* **then**
        Trace n separatrices for every irregular vertex on se_i;
    Trace n separation surfaces for se_i, which is guided by the separatrices;
Extract $\mathcal{B}_V$ and $\mathcal{B}_E$ from the intersections of all separation surfaces;
Extract $\mathcal{B}_F$ by decomposing separation surfaces into patches based on their intersections, i.e., $\mathcal{B}_E$;
Extract $\mathcal{B}_C$ by labeling hex elements that fall in different cuboid regions separated by $\mathcal{B}_F$ using a flooding algorithm.

Chapter 3

Previous Work

3.1 Hex-mesh Generation

Various methods exist to generate hexahedral meshes. Early non-parameterization based hex-mesh generation techniques include the paving and sweeping techniques as surveyed by Shepherd and Johnson [79] and the octree-based method [57]. They typically produce hex-meshes with excessive singularities and complex structure (e.g., 3-torus in Figure 6.15). Recent parameterization-based approaches, including polycube based methods [32, 55, 38] and frame-field based methods [69, 39, 52, 43], and skeleton guided sweeping techniques [83, 98] have better control on singularities and typically generate hex-meshes with simpler structures while having a high local element quality. In the following, I focus on such methods.

Polycubes: Polycubes allow the decomposition of an object into a set of larger hexahedral pieces. However, the quality of the resulting hexahedral representation strongly depends on the placement of polycube corners *on* the input triangle mesh. This challenging process has received a lot of attention recently [32, 94]. Automatic methods are usually difficult to control. Livesu et al. [55] and Huang et al. [38] introduced Polycuts and L1-Polycubes to improve the corner configuration of the conventional polycube. However, control of the interior structure of the volume is still missing. Recently, Li et al. [51] extended the conventional polycube to a generalized polycube (or GPC), which enables the curved cuboid representation of the elementary sub-volumes decomposed via shape analysis. While this enables the polycube map approach to be applied to more complex objects, the corners of the GPC are located on the surface, leading to degenerate elements (negative Jacobian) around the corners. The sub-volumes decomposed using our framework are curved cuboids, which is similar to GPC. I address the boundary degeneracy by finding the correspondence of the boundary corners of these cuboids in the interior of the volume.

Cross-field guided approaches: Cross-fields (or frame-fields) have been proven useful to assist the placement of quadrilateral elements when quadrangulating a triangular mesh [9, 13, 7, 72], as it provides consistent local frame information everywhere in the domain to guide the orientation of parameterization. Nevertheless, it is not clear on how to extend these methods from quadrangulation to hexahedralization. Huang et al. [39] proposed a first solution to creating a boundary-conformal 3D

cross-field via an expensive optimization. Due to insufficient control on the types of the singularities in the cross-field, their approach cannot be guaranteed to generate an all-hex-mesh. Nieser et al. [69] pointed out that only 10 types of singularities can lead to a valid all-hex-mesh. Recently, Li et al. [52] introduced the *Singularity Restricted Field (SRF)* that converts a general 3D cross-field to one that contains only these 10 types of singularities. After regularizing the SRF and fixing degeneracies, high-quality hex-meshes can then be generated using the CubeCover technique [69]. A similar work by Jiang et al. [43] also aims to derive a restricted cross-field from some initial cross-field using similar singularity operations as in [52]. The obtained cross-field was then used to compute a parameterization by solving a mixed-integer problem.

Both Polycubes and cross-field approaches may generate hex-meshes with degenerate boundary points (i.e., zero Jacobian). They apply an additional step, called *padding*, to relocate the degenerate boundary points into the interior of the volume. However, this was achieved by simply adding an additional layer of hex-elements, which may introduce additional small hexahedral components to the structure. This limits the order of splines that can be fitted in this layer.

Sweeping strategy and mid-structure for hex-meshing: There are other hex-meshing techniques that are based on sweeping and paving, such as the unconstrained paving and plastering [83] and a skeleton-based sweeping [98]. However, those techniques require the cross sections to be planar, and the structure of the obtained hex-meshes is uncontrollable. Therefore, they are typically suitable for the meshing

of CAD models rather than more organic appealing 3D models. And the introduced skeletal surface provides a means for explicit control on the structure of the generated mesh. The idea of utilizing a simplified mid-structure of an input model to assist the generation of hexahedral meshes has been explored by [59, 76].

3.2 Structure Optimization

Structure alignment: In 2D cases, the misalignment of singularities in the global structure can be greedily tackled by the methods introduced in [66, 8, 89]. Myles et al. [66] introduced T-mesh to simplify the patch domain by allowing the existence of T-junctions. While [8] aligns the base-complex by removing helical configurations, [89] simplifies the patch domain by optimizing the connectivity of separatrices originated from singularities. For hex-meshes, existing methods either simplify (or coarsen) the hex-meshes via local modifications of some hex sheets [78, 49] or alter the local hex-mesh structure via a hexahedral dual [91]. Both methods do not need to reduce the number of hexahedral components. To the best of our knowledge, our work presented in Chapter 5 is the first reported approach to fix the misalignment of singularities in 3D hex-meshes.

Structure simplification: Many methods have been proposed to produce level-of-details (LOD) quad-meshes by applying local operators, global operators, or their combinations to quad-meshes. Local operators affect only a small neighborhood of

a quad or a vertex, while a global operator can be arbitrarily large, and a single operation can affect the entire mesh. By interleaving the global and local operators, Daniel et al. [20] managed to produce a much coarse (or simplified) quad-mesh. Later, Daniels et al. [19] generate an all-quad mesh with a simple base domain while maintaining geometric fidelity through a keyframe-mapping technique. By adapting the simplification pipeline in [20] and using purely local operators, approaches in [21, 88, 11] can greatly simplify a quad-mesh while maintaining good element quality to some extent. For hex-meshes, existing methods either simplify (or coarsen) the hex-meshes via local modifications of some hexahedral sheets [10] or alter the local areas of the hex-meshes via hexahedral duals [91]. Neither method can guarantee to reduce the number of singularities and components in the base-complexes. In contrast, our work focuses on simplifying the global structures of 3D hex-meshes. Our technique described in Chapter 6 can efficiently reduce both singularities and components in the base-complex, and coarsen the hex-meshes much faster.

3.3 Hex-mesh Evaluation

Quality Metrics: Given a triangle/tetrahedral mesh input, generating an all-hex-mesh composed of hexahedral elements (topologically identical to cubes) has been studied for decades. As reviewed above, all these techniques are designed to generate hex-meshes consisting of hexahedra as close as possible to cubes. However, because of the arbitrary shapes of the boundaries of the input objects, it is generally impossible to maintain all the elements regular. Therefore, many metrics for evaluating the

quality of hex-meshes have been proposed, as summarized in Table 3.1. The Metric, Abbr., and Range columns of Table 3.1 stand for the indices, names, abbreviations, and valid value spaces of the metrics, respectively. Note that, I consider the left value of the range listed in Table 3.1 of a metric as its lower bound, while the right value of the range as its upper bound only when it is not $+\infty$. For the Trend column, $\uparrow$ means the higher value of the corresponding metric is preferred, while $\downarrow$ means the opposite.

For the calculations of the listed metrics, please refer to [86]. However, in most cases, the metrics for measuring hex-mesh quality are application-specific; users often have to balance the requirements of specific applications. For example, a highly inhomogeneous anisotropic problem may want some elements in "ugly shapes". The efficiency and robustness of some direct solvers (e.g. Cholesky decomposition) are less-dependent on the global structure and spectral features (eigenvalues) than some iterative solvers (especially Krylov space methods [33] and multi-grid methods [93]). To better understand the relationships among these characteristics, which can be further used to guide the development of specific hex-mesh optimization algorithms, in this dissertation, I investigate the correlation among these metrics for the first time.

Metric Evaluations: Over the past decades, intensive studies on the comparisons of tet-meshes and hex-meshes show that simulations using hex-meshes generally have a higher accuracy, a faster convergency rate, and more stable results [6, 87, 74]. However, to date, there exist few studies on hex-meshes with different properties.

Earlier, Muller et al. [64] investigated the effect of the scaled Jacobian and the number of hex-mesh elements on the simulation results, respectively. They concluded that the number of hex-mesh elements is important to the simulation quality. However, their experimental results suggested that with the minimum-scaled Jacobian above 0.1, the average scaled Jacobian did not have a notable impact on the accuracy of the solution. Since they only used a simple cylinder-like model throughout their experiments, it is difficult to judge the generality of their study. By performing FMA on electromagnetic, Motooka et al. [63] compared the accuracy and the computational time using meshes with different qualities. The quality is evaluated based on orthogonality, facet planarity, diagonal length ratio, and volume ratio of each hex element in the hex-mesh. However, these metrics are correlated; it's hard to tell which metric is most effective for the measurement of the convergence property. Knupp [47] evaluated a number of objective functions for hex-mesh optimization based on the local Jacobian matrix. Different from the work of [47] that evaluates their optimization framework using existing quality metrics, our work presented in Chapter 7 concentrates on evaluating the relationships among quality metrics. In additions, I applied our correlation analysis framework to study the relation of the existing metrics with the quality of three important hex-mesh applications, including the linear-elasticity problem [67], Poisson's equation-solving [45, 36] and Stoke equation-solving [3, 99], respectively.

Table 3.1: Statistics of hex-mesh quality metrics in [86]. Range columns with and without * represents the actual and user-specified bounds.

Metric	Abbr.	Range	Range*	Trend
minimum diagonal	Min. D.	[0, 1]	[0, 1]	↑
average diagonal	Ave. D.	[0, 1]	[0, 1]	↑
maximum dimension	Max. DM.	[0, +∞]	[0, +∞]	↓
mean dimension	mean DM.	[0, +∞]	[0, +∞]	↑
minimum distortion	Min. DIS.	[-∞, +∞]	[0, 1]	↑
average distortion	Ave. DIS.	[-∞, +∞]	[0, 1]	↑
maximum edge ratio	Max. ER.	[1, +∞]	[1, +∞]	↓
average edge ratio	Ave. ER.	[1, +∞]	[1, +∞]	↓
minimum Jacobian	Min. J.	[-∞, +∞]	[0, +∞]	↑
average Jacobian	Ave. J.	[-∞, +∞]	[0, +∞]	↑
maximum maximum-edge-ratio	Max. MER.	[1, +∞]	[1, +∞]	↓
average maximum-edge-ratio	Ave. MER.	[1, +∞]	[1, +∞]	↓
maximum aspect Frobenius	Max. AF.	[1, +∞]	[1, +∞]	↓
average aspect Frobenius	Ave. AF.	[1, +∞]	[1, +∞]	↓
maximum mean-aspect-Frobenius	Max. MAF.	[1, +∞]	[1, +∞]	↓
average mean-aspect-Frobenius	Ave. MAF.	[1, +∞]	[1, +∞]	↓
maximum Oddy	Max. O.	[0, +∞]	[0, +∞]	↓
average Oddy	Ave. O.	[0, +∞]	[0, +∞]	↓
smallest relative size squared	Min. RSS.	[0, 1]	[0, 1]	↑
average relative size squared	Ave. RSS.	[0, 1]	[0, 1]	↑
minimum Scaled Jacobian	Min. S. J.	[-1, 1]	[0, 1]	↑
average Scaled Jacobian	Ave. S. J.	[-1, 1]	[0, 1]	↑
smallest shape	Min. S.	[0, 1]	[0, 1]	↑
average shape	Ave. S.	[0, 1]	[0, 1]	↑
smallest shape size	Min. SS.	[0, 1]	[0, 1]	↑
average shape size	Ave. SS.	[0, 1]	[0, 1]	↑
smallest shear	Min. SE.	[0, 1]	[0, 1]	↑
average shear	Ave. SE.	[0, 1]	[0, 1]	↑
smallest shear size	Min. SES.	[0, 1]	[0, 1]	↑
average shear size	Ave. SES.	[0, 1]	[0, 1]	↑
largest skew	L. SK.	[0, 1]	[0, 1]	↓
average skew	Ave. SK.	[0, 1]	[0, 1]	↓
smallest stretch	Min. ST.	[0, 1]	[0, 1]	↑
average stretch	Ave. ST.	[0, 1]	[0, 1]	↑
largest taper	L. T.	[0, +∞]	[0, +∞]	↓
average taper	Ave. T.	[0, +∞]	[0, +∞]	↓
smallest volume	Min. V.	[-∞, +∞]	[0, +∞]	↑
largest volume	L. V.	[-∞, +∞]	[0, +∞]	↓

Chapter 4

Generating Simple Global Structures for Hex-meshing

As mentioned in Chapter 1.1, fewer components (i.e., a simpler base-complex) are desired for the task of spline-fitting. However, existing methods [52, 32, 51, 69, 43, 83, 98] cannot guarantee that a hex-mesh with simple structure is generated. Besides the importance of the simplicity of the global structure, given a set of simulation constraints, users may prefer a meshing orientation that is aligned with the anisotropic property of the simulation [81]. However, to date none of the existing techniques in the literature allows practitioners to efficiently generate a hex-mesh with controllable orientation.

In this chapter, I introduce a new volumetric decomposition technique for the generation of a simple and predictable structured hex-mesh. In our representation, I emphasize the creation of as few as possible hexahedral components to reduce the

number of extraordinary edges (Figure 1.1a), while allowing the orientation of the structure to follow a user-desired direction (kitten and bunny in Figure 4.10). Given an input surface/tetrahedral model, our pipeline employs a generalized sweeping strategy to decompose the volume enclosed by the input polygonal surface into a sequence of 2-manifold-level sets based on a user-specified 3D harmonic function. This effectively reduces the complex 3D spatial partitioning problem to a simpler partitioning problem in 2D. The final 3D partitioning strategy is constructed via matching the 2D solutions over the adjacent-level sets. The obtained level sets in our method are typically curved, which differentiates our method from planar sweeping strategies [83, 98]. Note that global structures of all the hex-meshes generated using our approach are well aligned. This property is guaranteed, since the global structure of a hex-mesh produced by our approach is controlled by the 3D skeletal structure without any misalignment through our sweeping strategy.

In particular, I make the following contributions.

1. I introduce the concept of a 2D *skeletal structure*. This structure has a simple quadrilateral configuration that reflects the primary characteristic of the 2D region. Figure 4.14a provides an example of such a structure (highlighted by different color components) in the cross section of a deformed torus.

2. I present a robust and automatic algorithm to extract the four corners of a 2D shape based on its medial axis. In comparison with corner placements found by following gradient lines of the surface harmonic function [51], our corner extraction can better align the hex-mesh structure with the surface features along the sweeping direction.

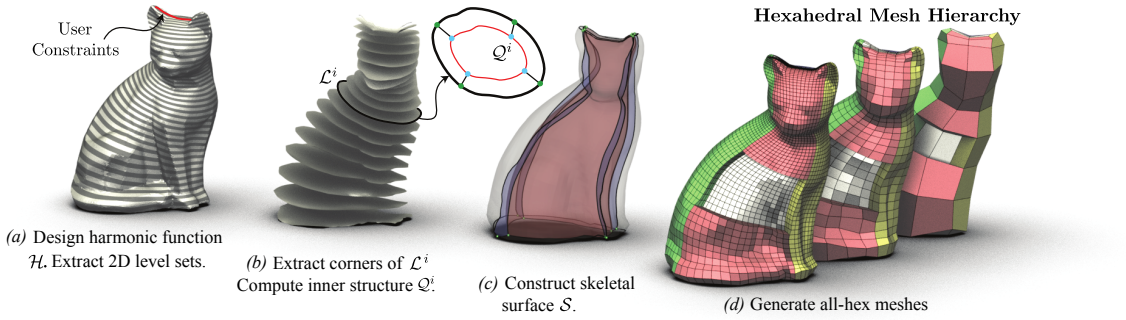


Figure 4.1: The pipeline of the proposed method. (a) A generalized sweep guided by a user-specified harmonic function; (b) corner extraction and inner structure construction in 2D parameter space; (c) the skeletal surface constructed by matching the 2D structure; (d) the multi-resolution hex-meshes generated based on the structured decomposition.

3. I introduce a *skeletal surface* to compute a structured volumetric decomposition. It is constructed by matching and connecting the 2D skeletal structures through adjacent-level sets with special attention paid to *bifurcations*. The resulting surface (Figure 1.1a) not only provides a valid all hexahedral partitioning strategy for the volume, but also serves as a user-controllable representation of the extraordinary structure for the subsequent 3D global parameterization.

4.1 Overview

Before presenting our method, I define the notation used in this chapter. I use calligraphic letters for objects in $\mathbb{R}^3$, for example, $\mathcal{L}$ denotes a 3D-level set generated by our sweeping process (see below), $\mathcal{T}$ denotes a set of triangles in 3D space, and $\mathcal{P}$ denotes a 3D point. Superscripts refer to level set identifications, while subscripts refer to the identifications of the geometric objects at a certain level set. For instance,

$\mathcal{P}_j^i$ represents the j^{th} point on the i^{th} level set. Regular letters, such as L, Q, and $\mathbf{p}$, represent objects in $\mathbb{R}^2$.

Let $(\mathcal{T}, \mathcal{V}_\mathcal{T}, \mathcal{C}_\mathcal{T})$ be a closed 2-manifold in 3D space represented by a triangle mesh, where $\mathcal{T}$ is the set of triangles, $\mathcal{V}_\mathcal{T}$ is the set of vertices, and $\mathcal{C}_\mathcal{T}$ is the connectivity of the mesh. The volume enclosed by $\mathcal{T}$ is filled with tetrahedral elements using an automatic tet-meshing method, such as Tetgen [82]. In Tetgen I use 0.5% of the diagonal of the bounding box of the model to create tetrahedral elements whose boundary triangles have similar sizes as the re-meshed input. Let $(\mathcal{H}, \mathcal{V}_\mathcal{H}, \mathcal{C}_\mathcal{H})$ define a tetrahedral mesh, where $\mathcal{H}$ is the set of tetrahedra, $\mathcal{V}_\mathcal{H}$ is the set of vertices defining the tetrahedra, and $\mathcal{C}_\mathcal{H}$ is the connectivity of the tetmesh.

Figure 5.8 illustrates our pipeline that generates the structured mesh from the given input triangle mesh $\mathcal{T}$, by executing:

1. Compute a harmonic function $h_u(x, y, z)$ on $\mathcal{H}$ based on user-specified constraints (Figure 5.8a). Decompose $\mathcal{H}$ into a sequence of non-planar level sets $\mathcal{L}^i$ (Figure 5.8b). $\mathcal{L}^i$ is flattened to a 2D level set using least-square conformal mapping (LSCM) [50]. Let $f_i : \mathcal{L}^i \rightarrow L^i$ be the resulting bijective mapping function (Sections 4.1.1 and 4.1.2).

2. Extract four corners for each level set L^i and align them with the adjacent-level sets. A skeletal 2D structure Q^i of L^i is constructed by projecting each of these corners to the interior (Section 4.2). The inner structure $\mathcal{Q}^i$ of $\mathcal{L}^i$ is obtained via $f_i^{-1}(Q^i)$ (Figure 5.8b).

3. Iteratively match $\mathcal{Q}^i$ through adjacent level sets and connect them to form a

3D skeletal surface $\mathcal{S}$ (Figure 5.8c) (Section 4.3).

4. Compute the 3D parameterization induced by the volumetric partitioning provided by $\mathcal{S}$. This parameterization can be used to generate structured all hex-meshes (Figure 5.8d) without T-junctions.

Note that there are four user-specified parameters in our pipeline: The resolutions along U, V, W directions (Section 4.4) and the number of level sets for the sweeping (Section 4.1.2). The remaining parameters mentioned in this chapter are set as either constant values or constant ratios, and do not require model specific tuning. In the following two sections, I describe the first step in this pipeline. Then, the remainder of the chapter is fully dedicated to the subsequent steps.

4.1.1 Computing 3D Harmonic Field

A scalar function $u(x, y, z)$ (in the remainder of this chapter, denote as u for simplicity), defined on $\mathcal{H}$, is harmonic if it satisfies Laplace's equation, i.e., $\nabla^2 u = 0$. Here, u satisfies the maximum principle, i.e., it only exhibits maxima and minima at user-specified locations on $\mathcal{H}$. This makes it a convenient and highly controllable tool to guide a meshing process. It has been used in a variety of mesh generation and volumetric-parameterization methods [23, 60], as well as in skeleton-extraction methods such as [35].

In this work, the harmonic function u is computed by first discretizing Laplace's equation using Galerkin's formulation [40]. The set of vertices $\mathcal{V}_H$ is decomposed into a set of constrained vertices $\mathcal{V}_B$, and a set of free vertices $\mathcal{V}_F$ for which a solution is

sought.

In our framework, the user has full control over u by defining $\mathcal{V}_B$ through vertex selections directly made on $\mathcal{H}$. As an example, Figure 4.10 illustrates two harmonic functions on the genus-1 kitten and bunny models, respectively, which creates two different yet valid structured hex-mesh representations. u is seen as a sweeping strategy as it is used to decompose the object into a set of slices, where a slice is a level set of u. This step is discussed in more detail in the following section. Note that if the resulted slices contain holes (non-multi-disk type), the user is prompted by our system to specify a different set of critical points to compute a new harmonic function.

4.1.2 Decomposition of $\mathcal{H}$

Given the harmonic function u, the object is decomposed into a set of slices $\mathcal{L}^i$. A slice $\mathcal{L}^i$ (Figure 5.8b), at value $u_i \in \mathbb{R}$ is the level set satisfying $u = u_i$. $\mathcal{L}^i$ is extracted using marching tetrahedra [16]. Depending on the choice of $\mathcal{V}_B$ and resulting saddle points [68], $\mathcal{L}^i$ can consist of multiple disjoint non-planar 2-manifolds represented as *triangle meshes* with boundaries [80].

If slices are placed such that every triangle in $\mathcal{T}$ is intersected by at least one slice, all features will be guaranteed to be present in $\mathcal{T}$. However, in this work, u_i is a uniform sample on the range of the harmonic function u. This uniform-sampling strategy allows the user to control the number of slices more conveniently. Except for some simple models, a dense cutting with about 500 slices is typically used for

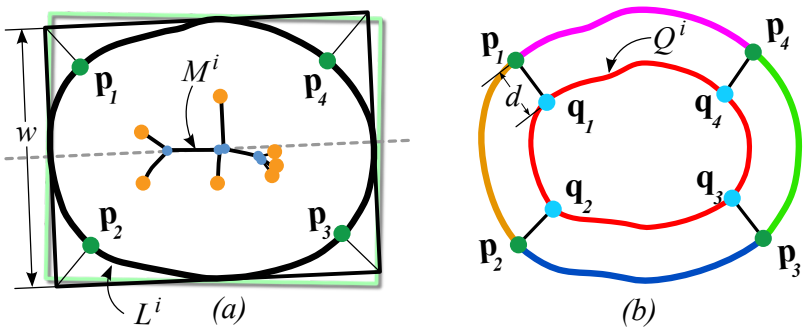


Figure 4.2: 2D structure extraction. (a) Flattened level set L^i with its bounding box guided by its medial axis M^i and boundary corners $\mathbf{p}_j$. This bounding box is different from the one (the light green box) obtained by performing PCA on L^i. (b) The corners of Q^i are found on the iso-contour by the one-to-one mapping between $\mathbf{p}_j$ and $\mathbf{q}_j$. The offset distance $d = d_m/3$, where d_m is the maximum scalar value of a distance field computed from B^i.

the examples shown in this chapter. This high number of slices assures capturing all the features present in $\mathcal{T}$. A more advanced slicing approach could be adopted (e.g., [58]). While it may improve accuracy to certain extent, it is computationally more expensive.

Finally, each slice $\mathcal{L}^i$ is flattened using the CGAL [1] implementation of LSCM [50]. The boundary of the flattened $\mathcal{L}^i$ is approximated with a periodic B-spline curve using the method proposed in [60].

4.2 Constructing 2D Skeletal Structure

In this section, I provide the definition and computation of the inner structure of a planar level set L^i.

Consider a 2D region L^i with closed boundary $B^i = \partial L^i$. A segmentation strategy

$\bigcup_{j=0}^{N-1} S_j^i$ of B^i partitions B^i into N segments separated by N boundary points, $\mathbf{p}_j^i \in B^i$. An inner structure Q^i is defined as a polygon with N vertices in the interior of L^i whose corners $\mathbf{q}_j^i$ map one-to-one to the boundary separating points $\mathbf{p}_j^i$. Q^i represents the characteristic of B^i, e.g., the primary orientation or curvature extrema with controllable complexity. For simplicity of the subsequent matching and the requirement to produce an all-hex configuration, in this work, I set $N = 4$.

As described previously, the most challenging part of extracting Q^i is to determine the locations of its corners. I adopt a two-stage approach to determine the corners of Q^i. First, I determine the locations of the corners on B^i based on its configuration. Second, I project the obtained corners on B^i to the interior of L^i without ambiguities in the gradient direction of a distance field computed with B^i being the zero level set. Figure 4.2 illustrates this process.

4.2.1 Corner Point Extraction

Given L^i I follow the approach in [58] to compute its medial axis. The medial axis is the locus of centers of maximally inscribed circles that are tangent to the boundary. The contact points of each maximal circle with the boundary curve are called foot points. The computed medial axis for L^i is effectively denoised because it is computed from a smoothed curve. It can be further simplified with an area based filtering. Figure 4.3a shows a slice of a twisted U-shape model and its simplified medial axis. For each branch of the medial axis, if the area (for example, the dotted area in Figure 4.3a) bounded by its branch point, corresponding foot points and

the boundary between foot points is smaller than 1/10 of the area of the slice, then I remove this branch. At this moment, if the simplified medial axis has only two end points, then I trace from the end points to the internal points, until the separation angle between the foot points of the current medial axis point is larger than a predefined value (I use 120° in this work). The corner points are the foot points of the traced medial axis points.

Otherwise there are more than two end points in the medial axis, and in this case we compute a principal component analysis (PCA) of the medial axis curves, to obtain the dominant direction from which to extract the bounding box. Once the bounding box is computed and the corners are mapped onto the flattened slices, the corner points are inversely mapped onto the original (non-planar) slices. Note that the bounding box of the medial axis may not produce desirable corners. In addition, for more symmetrical slices (e.g. circular shape), the orientation given by the PCA is less reliable. The corners of both situations will be adjusted by a smoothing process discussed next.

4.2.2 Corner Point Matching Over Level Sets

After the corner points at individual level sets are extracted, they are matched between adjacent 3D-level sets for the construction of a 3D inner-skeletal surface (Section 4.3). To determine the correspondence between corners at adjacent-level sets, I employ a distance-based greedy algorithm, modified by using the ratio of the eigenvalues of the PCA on the medial axis. If the ratio (λ^i in Algorithm 2) is larger

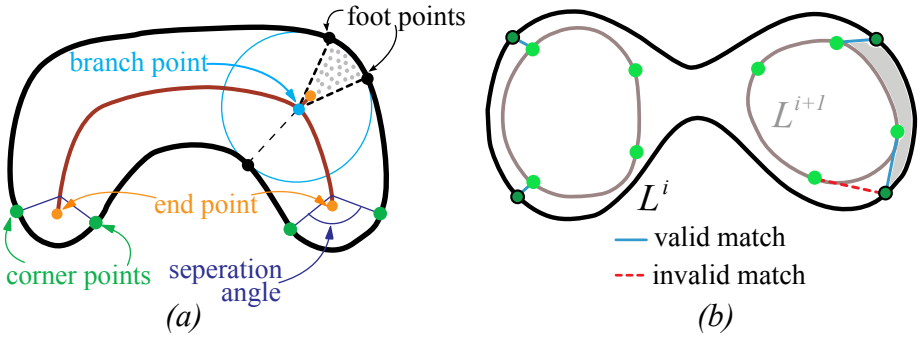


Figure 4.3: Corner extraction and matching. (a) illustrates the corners (i.e., green dots) extraction using a simplified medial axis (the dark red curve) of a slice of a twisted U shape. The simplified medial axis consists of one branch point (blue dot) and three end points (orange dots). Foot points of the branch point are shown as black dots. (b) shows the matching of the corners between two adjacent-level sets with a splitting bifurcation. The shaded area shows a highly distorted quad.

than a specified ϵ, each corner is matched to its nearest corner in the previous slice. Otherwise, the corners are recalculated by finding the points on the current slice that are closest to the corner points in the previous slice. Doing so enables us to avoid big matching jumps, by skipping the nearly symmetric slices with the ratio of eigenvalues close to one. In some symmetric slices, e.g., the slices crossing the pectoral fins of the dolphin model, by changing ϵ, users may control whether to put corner points close to shape features or away from them to get smoother results. Figure 4.4a shows the corners on the surface of a dolphin model with $\epsilon = 1.5$ and 2, respectively. For all models shown in this work, $\epsilon = 1.5$ was used.

Algorithm 2 provides pseudo-code for the extraction of the corner points on each level set and their matching over adjacent slices. Four or more chains can be computed in this way. Each chain that is formed by the matched corners is shown in the same color in Figure 4.4. When a bifurcation occurs, emphasis is placed on the

Algorithm 2: Pseudo code of corner point extraction and matching

Input : a set of slices $\{L^i\}, \epsilon, n$
Output: corner points $\{\mathcal{P}_j^i\}$
foreach L *in* $\{L^i\}$ **do**
  $\{m_j\} \leftarrow$ simplified medial axis of L;
  CM $\leftarrow$ covariance matrix of $\{m_j\}$;
  $\lambda_1, \lambda_2 \leftarrow$ eigenvalues of **CM** (λ_1 is the larger eigenvalue);
  $v \leftarrow$ the eigenvector associated with λ_1;
  if *there are two end points in* $\{m_j\}$ **then**
    trace from end points to m_1 and m_2 with the separation angle $> 120°$;
    $\mathbf{p}_1^i, \mathbf{p}_2^i, \mathbf{p}_3^i, \mathbf{p}_4^i \leftarrow$ foot points of m_1 and m_2;
  else
    $\mathbf{B_x} \leftarrow$ bounding box of L in direction v;
    $\mathbf{p}_1^i, \mathbf{p}_2^i, \mathbf{p}_3^i, \mathbf{p}_4^i \leftarrow$ the four closest points on ∂L to four corners of $\mathbf{B_x}$;
  $\lambda^i \leftarrow \sqrt{\lambda_1/\lambda_2}$;
map $\{L^i\}$ and $\{\mathbf{p}_j^i\}$ back to original space of $\mathcal{L}^i$;
foreach $\mathcal{L}$ *in* $\{\mathcal{L}^i\}$ **do**
  if $\lambda^i < \epsilon$ **then**
    $\mathcal{P}^i \leftarrow$ closest point on $\partial\mathcal{L}_i$ to $\mathcal{P}^{i-1}$;
match $\{\mathcal{P}_j^i\}$ across the slices and use them as the control points $\{\mathcal{P}_{c_j}{}^i\}$ to generate B-spline curves $\{\gamma_i^0\}$;
for *j=1 to n* **do**
  $(\{\mathcal{P}^i\}, \{\gamma_i^j\}) \leftarrow \text{SVDSA}(\{\mathcal{L}^i\}, \{\gamma_i^{j-1}\})$

continuity of the adjacent chains from the previous levels wherever possible to maintain good spacing of the corners. For example, in Figure 4.3b, the invalid match (shown by the red dash line segment) loses the continuity. The adjacent chain in the previous slice becomes non-adjacent after bifurcation. The correct matching is shown by the cyan line segments. However, forcing continuity of chains can lead to distortion. As shown in Figure 4.3b, the matching line segments in the right are almost tangent to L_{i+1}. Thus, the quad-region (shaded) formed by these four corners is skewed, leading to distortion in the subsequent hex-mesh. This is due to the independent nature of the extraction of the corners at their individual level sets and the rapid change of the surface features along the gradient of the harmonic function. I introduce a smoothing process to remove this noise and reduce distortion.

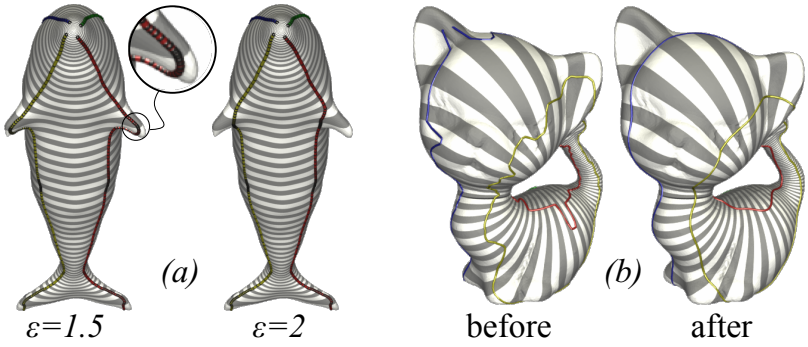


Figure 4.4: The corners of the dolphin model (left) with $\epsilon = 1.5$ and $\epsilon = 2$, respectively. The corners of the kitten model (right) before and after 100 smoothing steps. Corners are represented as colored dots. The harmonic function for kitten is designed by cutting the handle, where the vertices of the (two) cutting areas are set to minimum ($w = 0$) and maximum ($w = 1$), respectively.

For each chain, I use the corresponding vertices on each L^i to define a Schoenberg Variation diminishing-spline approximation (SVDSA) [17], $\gamma_i^0(t)$. I then find the closest points on each slice to the curve. Next, I use the new points to define

the control points of the next iterated SVDSA, $\gamma_i^j(t)$. If there is significant noise that results in sudden zig-zags in the original corner curves, this acts as a low-pass smoothing filter to the chains. The new set of control points have reduced wiggles. Here, the upper index indicates the number of iterations of this process. This process is fast and can be carried out as necessary to create smoothed corner curves, suitable for offsetting inwardly to create the inner volume structure. Algorithm 3 provides the pseudo-code of this smoothing process. Figure 4.4b shows the chains of corners of the kitten model before and after smoothing.

Algorithm 3: SVDSA

Input : $\{\mathcal{L}^i\}, \{\gamma_i\}$
Output: $\{\mathcal{P}^i\}, \{\gamma_i\}$
foreach γ *in* $\{\gamma_i\}$ **do**
  $\gamma\prime \leftarrow$ new B-spline curve;
  foreach *control point* $\mathcal{P}_c^i$ *in* $\gamma.\{control\_points\}$ **do**
    $s \leftarrow$ evaluate γ at the node value related to $\mathcal{P}_c^i$;
    $\mathcal{P}^i \leftarrow$ closest point on $\partial\mathcal{L}^i$ to s;
    $\gamma\prime.\{\text{control\_points}\} \leftarrow \mathcal{P}^i$;
  $\gamma \leftarrow \gamma\prime$

4.2.3 Computing the Corners of the Interior Structure

To obtain the corners of the inner structure Q^i, I first compute a distance field with the distance value as 0 for boundary vertices. Next, I compute an iso-contour in the interior of L^i corresponding to a distance value, d, based on the obtained distance field. The one-to-one mapping between the boundary and the interior contour is guaranteed because 1) I can always locate the four interior corners by following the

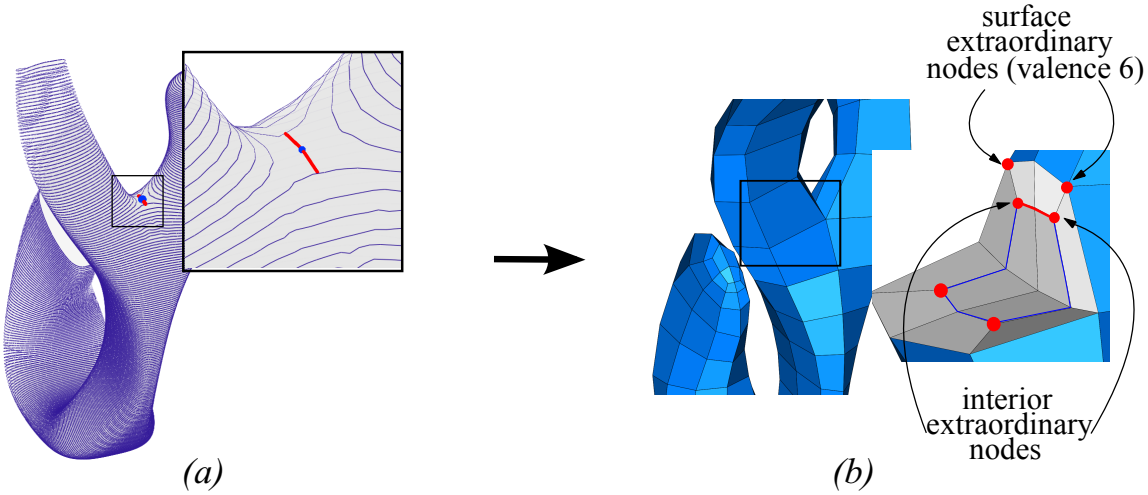


Figure 4.5: Bifurcation handling for the sculpture model. (a) shows the bifurcation of the sculpture model (zoom-in view shows the separatrix crosses the saddle point), (b) visualizes the obtained hex-mesh with the line structure being highlighted. The red nodes are the extraordinary nodes in the generated hex-mesh.

gradient directions of the distance field on L^i; 2) boundary and interior contour are partitioned into four pieces by the four corners. For each piece of the interior contour, the to be matched interior points can be found by accumulating chord length parameterization using the same ratios as boundary curve.

As L^i is the flattening of $\mathcal{L}^i$ via the bijective mapping f_i, $\mathcal{Q}^i$ can be recovered from Q^i via f_i^{-1}.

4.3 Extracting 3D Skeletal Surface

After constructing the inner structure $\mathcal{Q}^i$ for each individual level set $\mathcal{L}^i$, I now describe how to match the inner structure $\mathcal{Q}^i$ through adjacent-level sets to form a

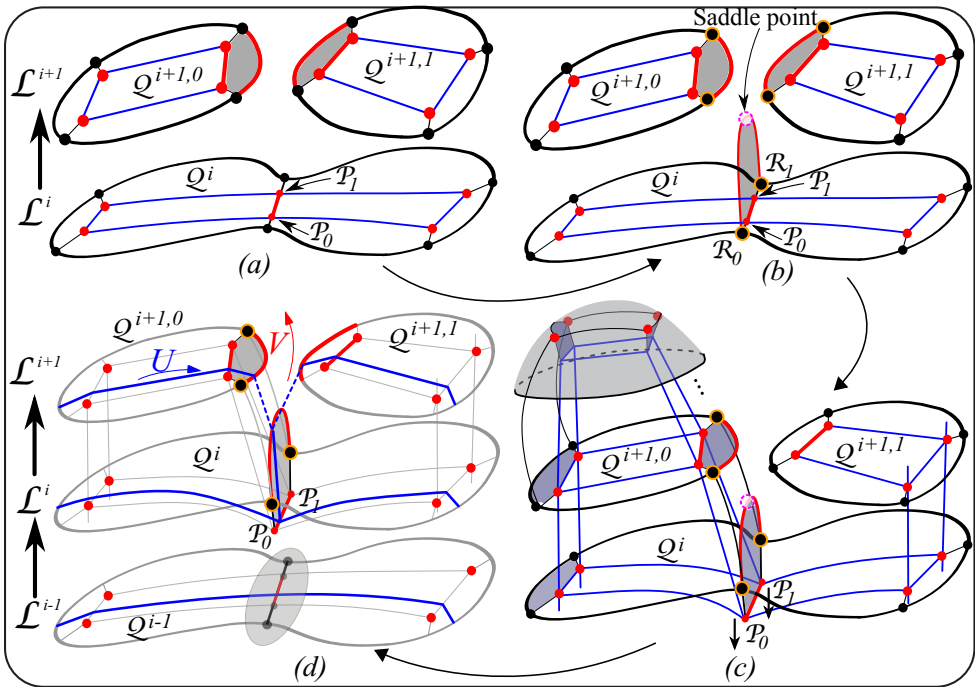


Figure 4.6: Illustration of bifurcation handling. (a) illustrates that a level set $\mathcal{L}^i$ splits into two components in $\mathcal{L}^{i+1}$, so does the corresponding inner structure. (b) demonstrates the two unmatched patches (i.e., the shaded regions) are mapped to a saddle patch formed by the separatrix segment at $\mathcal{L}^i$, (c) shows the connection between $\mathcal{Q}^i$ and $\mathcal{Q}^{i+1,0}$ and the hex element formed by the *last level set* and the surface, i.e. the cap. Note that $\mathcal{P}_0\mathcal{P}_1$ has been pushed down along the inverse gradient of harmonics function to eliminate the degeneracy, (d) illustrates the matching of the parameterizations of $\mathcal{L}^{i-1}$, $\mathcal{L}^i$, and $\mathcal{L}^{i+1}$ guided by the inner structure.

partitioning surface $\mathcal{S}$ (in 3D) (see Figure 4.8 for examples), which is referred to as the *inner-skeletal surface*. Based on the matched corners determined by Algorithm 2, I identify the corresponding matched edges of $\mathcal{Q}^i$ and $\mathcal{Q}^{i+1}$, which are connected to construct a quadrilateral face.

The configuration of the boundary surface and the selection of the harmonic function may cause bifurcations that split a preceding level set $\mathcal{L}^i$ into multiple connected components in the current level set $\mathcal{L}^{i+1}$, or vice versa, e.g., at the base of

the two branches of the sculpture model (Figure 4.5a). These bifurcations correspond to the saddle points in the harmonic function, and their identification can be performed automatically and robustly. Consequently, following the sweeping direction, the structure of $\mathcal{Q}^i$ also undergoes splitting or merging to accommodate such topological changes. Section 4.3.1 details the handling of the splitting scenario. The merging case is handled analogously. n-way bifurcations are divided into a sequence of 2-bifurcations and are handled individually.

4.3.1 Matching of $\mathcal{Q}^i$ Across Bifurcations

Figure 4.6a depicts a case of splitting a level set. During this change, $\mathcal{Q}^i$ of $\mathcal{L}^i$ splits into two components, $\mathcal{Q}^{i+1,0}$ and $\mathcal{Q}^{i+1,1}$ in $\mathcal{L}^{i+1}$. There is no one-to-one mapping between the sub-regions of $\mathcal{L}^i$ and $\mathcal{L}^{i+1}$ partitioned by $\mathcal{Q}^i$, $\mathcal{Q}^{i+1,0}$ and $\mathcal{Q}^{i+1,1}$, respectively. Specifically, an interior edge, $\mathcal{P}_0\mathcal{P}_1$ (the red line segment in Figure 4.6b) that splits $\mathcal{Q}^i$ into two components must be mapped to two respective edges of $\mathcal{Q}^{i+1,0}$ and $\mathcal{Q}^{i+1,1}$. In addition, if mapping different components of the two level sets as indicated by the colors shown in Figure 4.6a, two unshaded regions of $\mathcal{L}^{i+1}$ do not have a correspondence. This is addressed by performing the following steps:

First, computing a small segment on the surface that crosses the saddle point and intersects with $\mathcal{L}^i$ at $\mathcal{R}_0$ and $\mathcal{R}_1$, respectively (i.e., the red curved segment in Figure 4.6b). A new section, referred to as a saddle patch and formed by the curve $\mathcal{R}_0\mathcal{R}_1$ and the edges $\mathcal{P}_0\mathcal{P}_1$, $\mathcal{R}_0\mathcal{P}_0$, and $\mathcal{P}_1\mathcal{R}_1$ can then be mapped to the two unmatched components (the shaded regions in Figure 4.6c). In practice, a short segment across

the saddle point on the surface can be obtained by computing two separatrices starting from the saddle point along the incoming (outgoing) gradient flow direction for a splitting (merging) bifurcation. The computation of these separatrices is terminated when they reach the level set $\mathcal{L}^i$ ($\mathcal{L}^{i+2}$ for merging).

Second, pushing down $\mathcal{P}_0\mathcal{P}_1$ along the negative-harmonic gradient direction as long as $\mathcal{P}_0\mathcal{P}_1$ is still above $\mathcal{L}^{i-1}$ (Figure 4.6c), since $\mathcal{P}_0\mathcal{P}_1\mathcal{R}_1\mathcal{R}_0$ is a degenerate element as the four corners are almost collinear.

Third, by creating a saddle patch, computing the mapping between $\mathcal{Q}^i$ and $\mathcal{Q}^{i+1}$. However, this solution leads to a T-junction configuration when considering $\mathcal{Q}^i$ and $\mathcal{Q}^{i-1}$ since $\mathcal{Q}^i$ has been split into two components in the previous process. See Figure 4.6d for an illustration. Specifically, if a line segment enclosed by the shaded ellipse is not added in $\mathcal{L}^{i-1}$, a T-junction configuration occurs. More discussions are provided on this issue in Section 4.4.

Figure 4.5b shows the generated hex-mesh for the sculpture model at the bifurcation. The structure of this mesh formed by the irregular edges is highlighted in blue. The colored dots are the intersections of the irregular edges with the cutting planes.

Bifurcation improvement The above basic bifurcation handling introduces a valence-6 extraordinary node at each side of the bifurcation on the boundary quad mesh (orange dot in the left figure of Figure 4.7b). This may lead to large distortion in

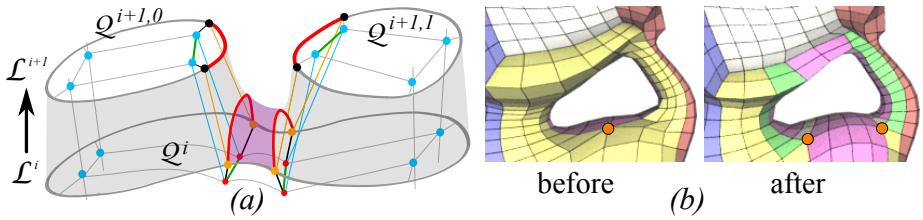


Figure 4.7: Improvement at bifurcations. (a) illustrates the process that splits the line segment across the saddle into two. Three hexahedral components are generated instead of two. Different colored lines indicate their correspondence relation. (b) shows the result of the kitten model before and after the bifurcation improvement process.

elements at bifurcations whose neighborhood is relatively flat. To lessen the distortion, instead of associating the two unmatched components to a section across the saddle point as shown in Figure 4.6c, I map them to two sections as shown in Figure 4.7a. This splits the valence-6 (the number of neighboring hexahedral elements) extraordinary point into two valence-5 nodes (Figure 4.7b). This adds an additional component at the bifurcation (e.g., the purple region in Figure 4.7a). Since the gradient of the harmonic field is curl-free, by transferring this configuration down (splitting bifurcation) or up (merging bifurcation) following the sweeping direction, T-junction configurations can be avoided. Note that this improvement is not required and can be selected by the user according to the surface characteristics around the bifurcations. I have applied this improvement to the kitten, fertility, blade, and rocker arm models in our experiments.

4.3.2 Properties of the Skeletal Surface $\mathcal{S}$

Figure 4.8 (top row) shows the skeletal surfaces $\mathcal{S}$ of a variety of 3D objects computed using the aforementioned framework. As explained earlier, the corners of the inner structure $\mathcal{Q}^i$ correspond to the extraordinary points of a 2D parameterization derived by $\mathcal{Q}^i$. By matching $\mathcal{Q}^i$ over adjacent-level sets to construct $\mathcal{S}$, the extraordinary points in 2D now form line structures in 3D as indicated by the intersections of different surface sheets (with dark blue and red in Figure 4.8) of $\mathcal{S}$. These line structures, referred to as the extraordinary edges, correspond to the irregular edges (or singular edges by Nieser et al. [69]) with valence not equal to four in the obtained hexahedral mesh.

The extraordinary edges in $\mathcal{S}$ can start and end at either some extraordinary nodes in the interior of the volume or on the exterior surface. Specifically, the extraordinary lines start and end on the boundary surface when a component of the level set either is born or vanishes (e.g., the four red points on the top of the shaded surface in Figure 4.6d). In the meantime, the extraordinary lines meet at the extraordinary nodes at the bifurcations. Figure 4.6f illustrates this. The two red points (corresponding to $\mathcal{P}_0$ and $\mathcal{P}_1$ in the previous steps) are valence-5 extraordinary nodes (i.e., there are five edges incident to each of them) in the hex-mesh. In the meantime, the two orange points (corresponding to $\mathcal{R}_0$ and $\mathcal{R}_1$) are valence-6 extraordinary nodes on the boundary quad-mesh.

The base-complex $\mathcal{C}_\mathcal{B}$ of the subsequent hex-mesh can be derived from the skeletal surface $\mathcal{S}$. Specifically, if the sweeping does not involve bifurcations, $\mathcal{S}$ and $\mathcal{C}_\mathcal{B}$ have

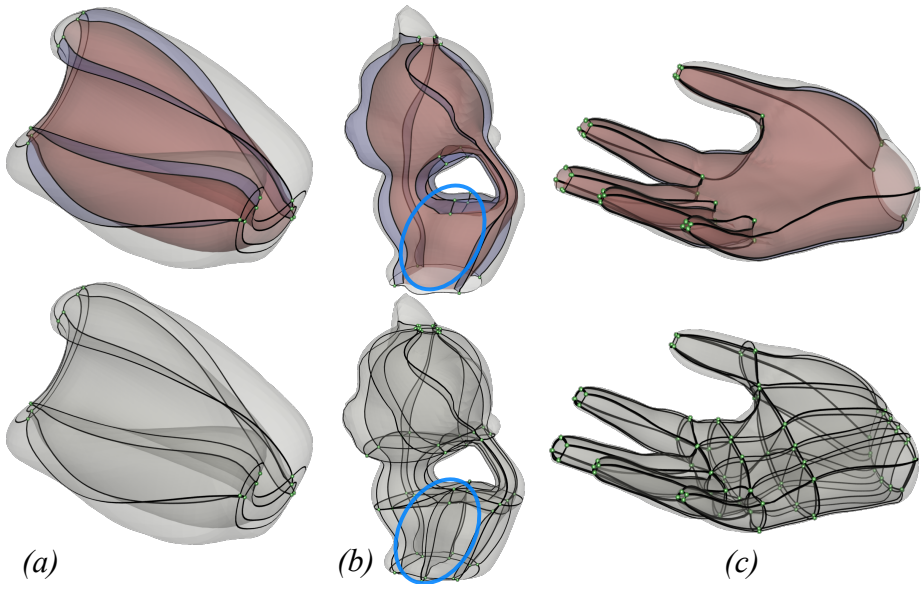


Figure 4.8: Inner skeletal surfaces $\mathcal{S}$ and base-complexes $\mathcal{C}_{\mathcal{B}}$ of the twisted U (a), kitten (b), and hand (c). The top row shows the skeletal surfaces with the singular line structure being highlighted (i.e., black curves). Red sheets show the interior surfaces, while blue sheets are the separation surfaces connecting the interior surfaces and the exterior boundary surfaces. The bottom row shows their corresponding base-complexes. Nodes and edges are the corners and edges of the cuboids, respectively.

a one-to-one mapping (Figure 4.8a). If bifurcations exist, the skeletal surfaces may contain T-junctions because of the way bifurcations are handled(see Figure 4.6 and Section 4.3.1). Nonetheless, their corresponding base-complexes do not contain T-junctions (Figure 4.8b and 4.8c) as a separation surface will be added at T-junctions to enforce an all-hex configuration in the base-complex structure. Therefore, the nodes in the base-complexes (green dots) consist of both, extraordinary points and regular points.

4.4 Multi-resolution Hex-Meshing

This section describes how to compute a seamless 3D parameterization and induce hex-meshes with large hexahedral components, after constructing $\mathcal{S}$.

Parameterization: Similar to computing $\mathcal{S}$, one parameterization direction w is provided by harmonic function $u(x, y, z)$. The computation of a 3D parameterization can be simplified by two steps: 1) compute a 2D parameterization f^i of $\mathcal{L}^i$ based on its $\mathcal{Q}^i$; and 2) match f^i and f^{i+1} to obtain a 3D parameterization $\mathcal{F}$.

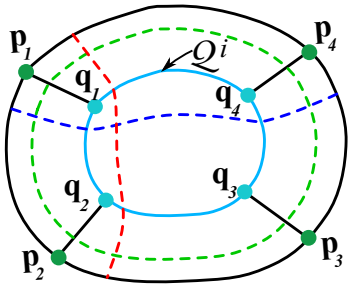


Figure 4.9: Iso-lines for 2D parameterization.

The 2D parameterization, $f^i = \bigcup_j f^i_j$ of $\mathcal{L}^i$, consists of five patches. Each patch $\mathcal{L}^i_j$ is triangulated and can be mapped to a rectangular region via $f^i_j : \mathcal{L}^i_j \rightarrow [0, n] \times [0, m] \subset R^2$. To compute f^i_j, I applied Floater's mean-value coordinates [24] to calculate the u and v values for each vertex of $\mathcal{L}^i_j$. To guarantee a continuous parameterization across the boundaries of different patches, I classify the parameterization directions into three groups, i.e., the red, green, and blue dotted curves, respectively, as shown in Figure 4.9.

Given the 2D parameterizations for all level sets, a 3D-volumetric-parameterization can be constructed accordingly. For the adjacent-level sets without bifurcation, the

correspondence of their 2D parameterizations is one-to-one. When a bifurcation occurs, I match f^{i-1}, f^i, and f^{i+1} via the correspondences between their inner structures obtained in Section 4.3.1 as illustrated by Figure 4.6e. In this case, the bifurcation occurs between level sets $\mathcal{L}^i$ and $\mathcal{L}^{i+1}$. To build the correspondence between f^i and f^{i+1} (composed of $f^{i+1,0}$ and $f^{i+1,1}$), I make use of the two sides of the saddle patch, which splits the inner structure $\mathcal{Q}^i$ (and f^i) into two components. The segment crossing the saddle is mapped to the line segment $\mathcal{P}_0\mathcal{P}_1$ parallel to the parameterization direction V (red). Its perpendicular direction is denoted as U (blue), and the integer value of $\mathcal{P}_0\mathcal{P}_1$ is u_P. To avoid a T-junction configuration, I set the numbers of isolines parallel to the U direction to be the same for both $f^{i+1,0}$ and $f^{i+1,1}$. That is, I use the same number of samples along the six red segments (i.e., along V parameterization direction) in both level sets $\mathcal{L}^i$ and $\mathcal{L}^{i+1}$, as illustrated by the intersections of the blue isolines with the six red segments (Figure 4.6e). The matching between f^{i-1} with f^i can be coordinated after the above process to avoid a T-junction configuration. Fortunately, the split of $\mathcal{Q}^i$ does not introduce a new singular structure at the splitting edge $\mathcal{P}_0\mathcal{P}_1$, i.e., both $\mathcal{P}_0$ and $\mathcal{P}_1$ are still valence-4 nodes. Therefore, T-junctions can be avoided by enforcing an isoline to be included in the parameterization f^{i-1} that has the same integer value u_P to the one that corresponds to $\mathcal{P}_0\mathcal{P}_1$ in f^i. This iso-line is highlighted by the shaded ellipse in level set $\mathcal{L}^{i-1}$ in Figure 4.6e.

Hex-meshing: The hex-mesh is constructed in two steps. First, an all-quad mesh is obtained by following the iso-lines of the integer values of the 2D parameterization for each level set. Second, all the quad meshes are matched in the same

fashion as the construction of the 3D parameterization. The numbers of samples, n_U, n_V, and n_W, are user-provided parameters that control the resolution of the hex-mesh.

Since I cut densely in the first step, when a coarse hex-mesh is preferred, I merge hex elements that are in two neighboring levels to keep an approximate regular aspect ratio of edge length in three parameterization directions. To improve the quality of the obtained mesh, I perform Laplacian smoothing on both the interior and exterior vertices and then optimize the mesh using the technique by Knupp [48]. Note that since the boundary extraordinary points are explicitly placed on the surface regions that correspond to the first and last slices (see the four red dots at the top of the surface in Figure 4.6c), a process similar to the padding [77] may be carried out to offset these extraordinary points to the interior. However, this process is optional and only used to improve the element quality in those regions if needed. Although all the hex-meshes tested in this work have positive Jacobians, I cannot guarantee that our pipeline can always generate hex-meshes without inverted elements.

4.5 Results

I have applied our proposed approach to various models. All the obtained hex-meshes have large hexahedral structure. Figure 4.10 provides the hex-meshes of a variety of 3D objects with three different resolutions. Hexahedral elements that belong to the same components of the structure of the mesh are shown in the same colors. For all the models shown in this chapter, at most 600 slices are used for the cutting,

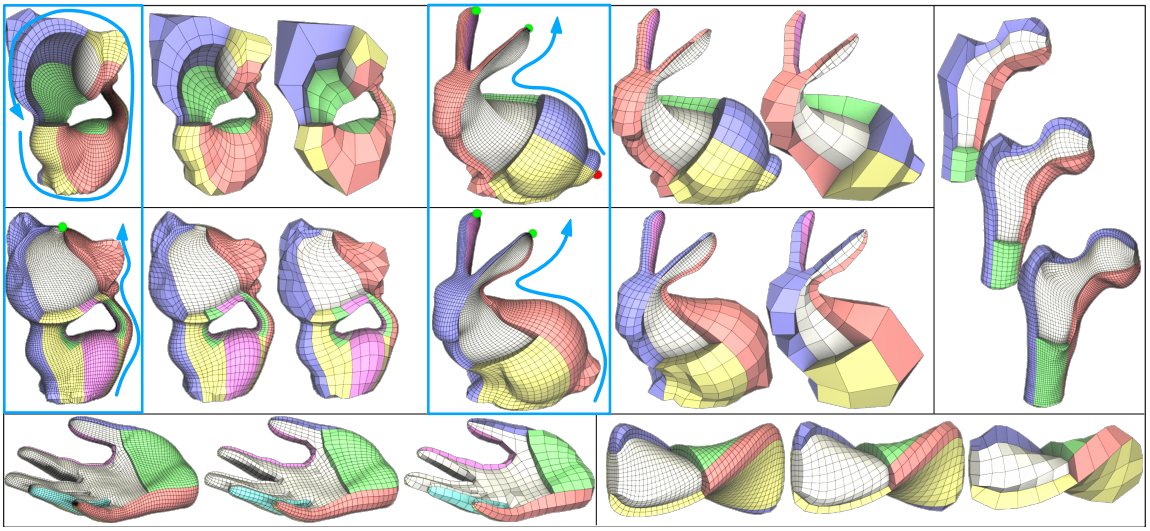


Figure 4.10: Different resolutions of the hex-meshes generated for a variety of 3D models. Note that the components with white colors are in the interior, while the others with different colors are at the exterior.

which takes up to 2 minutes to compute. The corner extraction (including medial axis computation) takes one min per slice. The time spent on the generation of hex-meshes ranges from 5 seconds to 2 minutes, depending on the mesh resolution. All timings are obtained on a PC with Intel i7 2670QM 2.2GHz CPU and 8GB RAM. In the following, I discuss our results with respect to control, comparison with existing methods, element quality, and volumetric B-spline-fitting.

In Figure 4.10, the kitten model is meshed from two different choices of harmonic functions, one with bifurcation and one without bifurcation. The blue arrows indicate the directions of the harmonic functions. Similarly, for the bunny model, both meshes have their maxima at the tips of the ears (in green). While the upper bunny has its single minimum at the tail (in red), the lower bunny has its whole base fixed as minimum (not visible from the given view). These examples demonstrate that

our pipeline outputs valid meshes for different user-specified harmonic functions. The choice of harmonic functions is mostly application-specific. For instance, the problem of establishing continuity among elements given the kitten model without bifurcation is significantly reduced, as one does not have to deal with a bifurcation point during a higher-order spline fitting process. A sweeping strategy could be derived automatically based on shape analysis of the input object. However, this is beyond the scope of this work.

Higher genus models: I applied our method to the fertility model (Figure 4.11) to demonstrate that our pipeline can be used for higher genus models. The left-most image is the base-complex of these meshes, while the right most shows the Jacobian visualization and histogram of the Jacobian distribution. This object is meshed based on a harmonic function with two critical points: One minimum (at the back of the mother), and one maximum (at the base opposite to the minimum). This harmonic function results in a sweeping that generates 3-way bifurcations in the level sets, which are handled by splitting 3-way bifurcations to two consecutive 2-way bifurcations (Section 4.4).

CAD models: Methods that output hex-meshes with large structure are often challenged by the complexity of input models. In this context, CAD models are especially difficult to mesh. To demonstrate that our method can be used to mesh a certain class of CAD models, I applied it to the blade and the rocker arm. Figure 4.12 shows the results. Note that the sharp features of these CAD models are properly preserved by our corner extractions. The element quality of the meshed rocker arm is comparable to those generated by existing methods. Even so, distorted elements can

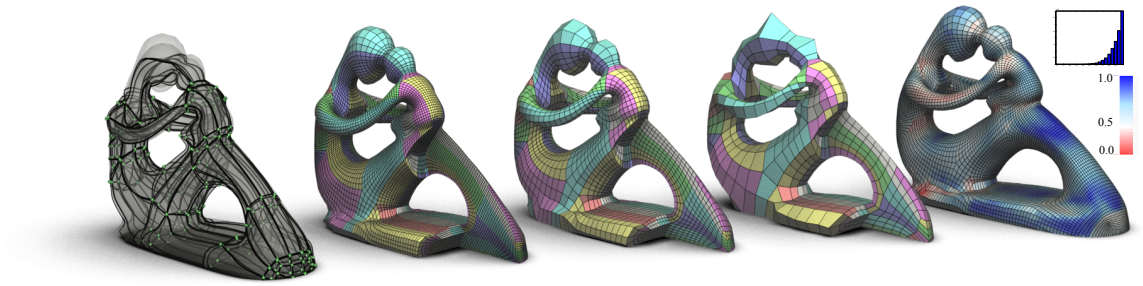


Figure 4.11: The hex-meshing result of the fertility model. The left-most image shows the base-complex; the middle three images show generated hex-meshes with decreasing resolution; and the right-most one shows the Jacobian visualization and the histogram of the Jacobian distribution.

be observed in the areas that the surface normal is almost parallel to the gradient direction of the harmonic field (e.g., the flat region of the blade as highlighted in Figure 4.12b). This may be addressed by inserting additional extraordinary points in this area in a similar fashion to the extended bifurcation handling shown in Figure 4.7.

4.5.1 Comparison With Existing Methods

Comparison with cross-field based methods: Figure 1.1 compares our method with the SRF approach [52]. The structure of the hex-mesh with SRF is *guided* by the structure of SRF (i.e., the singular graph). However, as pointed out by its authors, the structure of the obtained hex-mesh may not match the structure of SRF due to the parameterization. Furthermore, the extraordinary (or singularity) points are not always aligned, leading to many small components in the structure. For instance, there are 259 components in the hex-mesh of the bunny using SRF (Figure 1.1b) while ours has only 18 components (Figure 1.1a).

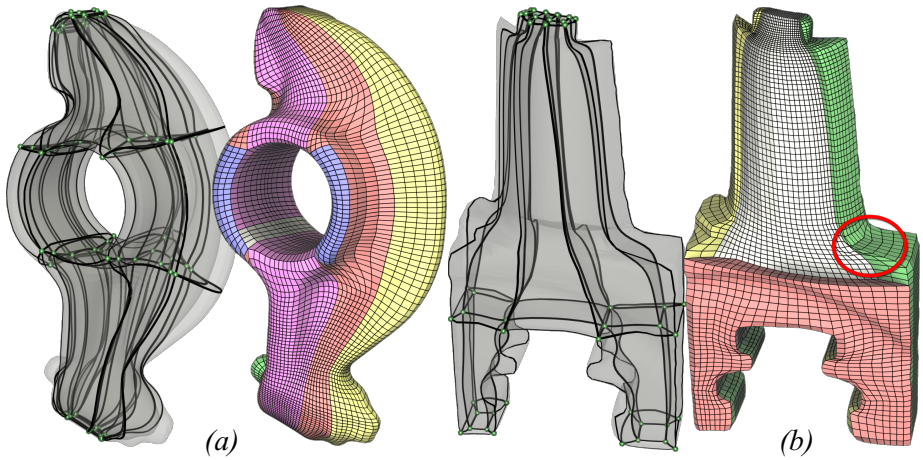


Figure 4.12: Results of two CAD models with our method: (a) the rocker arm and (b) the blade. The circle highlights the area with distorted elements in the blade mesh.

Figure 4.13 shows a deformed-rabbit model. While this frame-field based method fails to find a valid all-hex mesh for this model due to certain global degeneracies in the parameterization [43], our method successfully computes valid hex-meshes with various resolutions.

Compare to Polycube based methods: Figure 1.1 compares our method with the L1-Polycube method [38]. In order to remove polycube corners from the exterior, they are pushed into the interior of the object by adding one boundary layer. While it might be possible to apply this method to a wider range of input models, this offsetting process cannot guarantee to preserve the large polycube structure, and may result in additional T-junctions. Removing them leads to additional smaller hexahedral components. For instance, the bunny model meshed by the L1-Polycube approach has 422 components (Figure 1.1b) and 580 components with Polycut, which makes it difficult to establish a smooth basis almost everywhere in the model. In

contrast, the structure of the hex-meshes produced with our method is simpler, predictable and controllable.

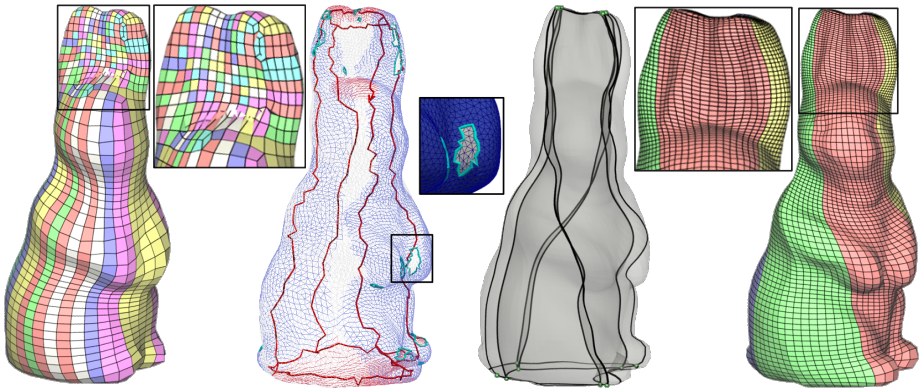


Figure 4.13: A rabbit model where the frame-field based method fails and our method succeeds. (a) Demonstrates the global degeneracy, resulting in a invalid hex-mesh. Red lines are inner singularities and cyan lines are surface singularities, they are processed by using the method proposed in [43], black rectangle shows a global degeneration; (b) our all-hex-mesh and base-complex.

Compare to GPC: With Generalized PolyCubes (GPC) [51], the faces of each cuboid can be curved, which is similar to our representation. However, an important difference is that the edges of the GPC cuboid are obtained by tracing along the gradient of a harmonic field, which may follow surface features only by coincidence. Figure 4.14 shows a comparison of the placement of the corners for the deformed torus with the proposed extraction technique guided by the medial axis of the 2D level sets (a) and the one (b) that aligns the corners by following the gradient of a harmonic function. From the comparison, I see that the one generated by following the harmonic field fails to capture the transition of the surface configuration, leading to the improper orientation of the structure of the obtained hex-mesh, i.e., the structure is not aligned with the anisotropy property of the cross section. Note that the

first four corners of the first slice for the harmonic field based approach are extracted using our proposed algorithm. This indicates that only focusing on obtaining the optimal corners at the initial slice is typically insufficient. Furthermore, for all the models, GPC places corners on the boundary rather than in the interior. This is generally avoided by most hex-meshing techniques, especially when dealing with natural shape models, since degenerate elements arise quite often around singularities on the boundary.

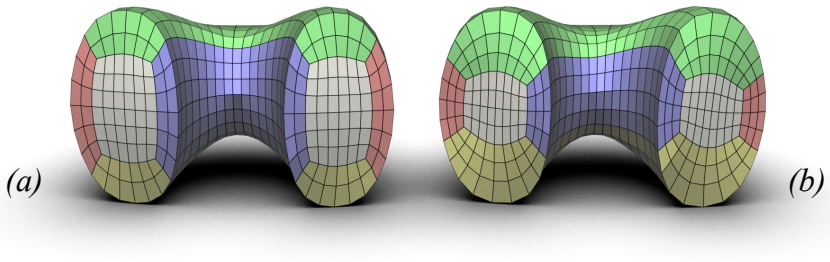


Figure 4.14: The issue of misalignment of the hex-mesh structure with the surface feature induced by tracing gradient line along the surface harmonic field (b) is addressed by our method (a). Meshes in (a) and (b) are cut to show the interior.

4.5.2 Element Quality

Figure 4.15 provides the visualization of the scaled Jacobian values [28] of the meshes shown in Figure 4.10 with blue denoting Jacobians close to one and red close to zero. The histograms show the distribution of Jacobians for the finest versions of the meshes with x-axis representing the Jacobian values (increasing from left to right) and y-axis being the number of hex-elements. For all histograms, the majority of our hex-elements have large Jacobians, i.e., the histograms have larger y-values close to the right, which is desired. The left value of the script below each histogram is the

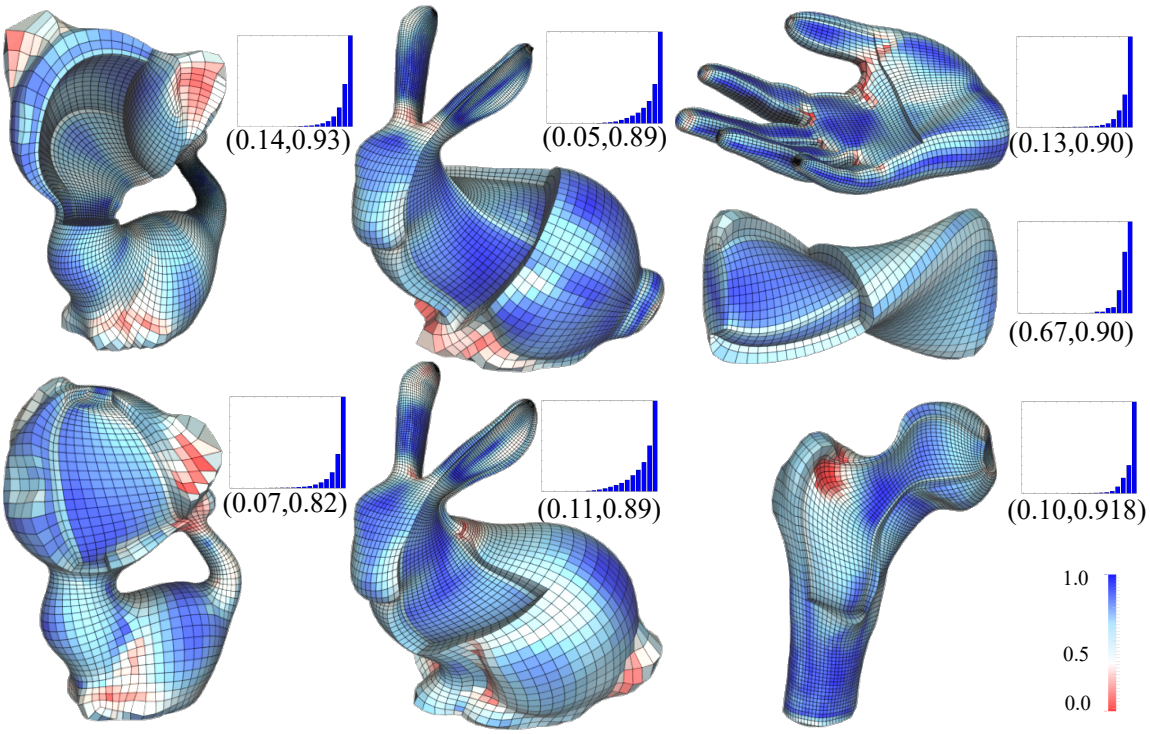


Figure 4.15: Visualizations and histograms of the Jacobian value distribution of a number of hex-meshes. The red-white-blue color coding is used with red indicating smaller Jacobian values. The numbers show the minimum and average Jacobian values.

minimum Jacobian of the corresponding mesh, while the right provides the average. Table 4.1 provides the component numbers and scaled Jacobians of a number of hex-meshes generated by our method and those produced by other methods. More results and statistics are provided in the supplemental material.

Based on the results and comparison shown above, the meshes generated by our method have much smaller numbers of components. Their average Jacobian are comparable to SRF, PolyCube and Polycut methods, while their minimum Jacobians may not be as good for more complex objects. This is especially the case near bifurcation areas and surface areas whose normals and the sweeping direction have

similar angles. This can be seen in the highlighted area of the blade model (Figure 4.12b). Ideally, their angle should be close to 90 degrees. This can be alleviated, to some extent, by refining the structure of the mesh (i.e., introducing additional extraordinary points). However, this refinement process needs in-depth investigation, which I plan to study in a future work.

Models	#Hex	Scal. Jac.	#Comp
Bunny[our method]	69984	0.891/0.108	18
Bunny[SRF]	133632	0.940/0.293	259
Bunny[Volumetric Polycube]	81637	0.953/0.138	1745
Bunny[L1-Polycube]	37734	0.926/0.382	422
Bunny[Polycut]	74084	0.958/0.274	580
Kitten[our method]	3445	0.866/0.257	5
Kitten[L1-Polycube]	7083	0.910/0.424	233
Fertility^{+}[our method]	20240	0.828/0.182	300
Fertility[SRF]	13584	0.911/0.351	1352
Fertility[Volumetric Polycube]	19870	0.949/0.196	635
Fertility[Polycut]	53702	0.900/0.259	693
Rocker-arm^{+}[our method]	11368	0.826/0.110	82
Rocker-arm[SRF]	10600	0.866/0.209	1149
Rocker-arm[L1-Polycube]	24346	0.920/0.439	686
Rocker-arm[Polycut]	56667	0.912/0.370	664

Table 4.1: Comparison of hex-meshes produced by our approach with those by SRF, Volumetric Polycube, Polycut and L1-Polycube methods.

4.5.3 Spline Fitting

Our method imposes constraints during the volumetric decomposition stage, resulting in a volume parameterization free of T-junctions. Because of this, a volumetric B-spline patch can be fit to each of the cuboids individually using standard fitting methods as used in [17]. The union of these patches resembles an approximation of

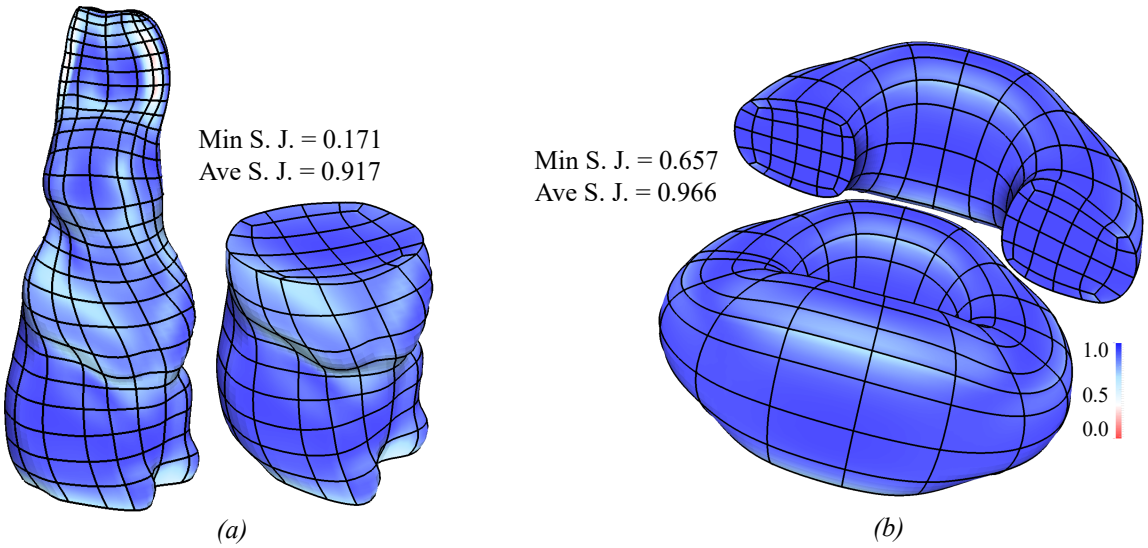


Figure 4.16: Spline fitting of a rabbit (a) and deformed torus (b).

the input object.

In the standard case, a volumetric B-spline patch is C^0 at its boundary, but has higher continuity properties (e.g., C^2) in its interior. Coarse structure as generated by our method may result in distorted elements, which may be seen as a disadvantage. However, points in the resulting union of volumetric B-spline patches are C^2 almost everywhere. The predictable and simple structure of our output potentially increases the continuity by reducing the number of boundaries between patches. This is a direction that I are actively investigating. Methods producing a hex-mesh lacking large structure such as SRF and Polycube methods, do not allow many regions to have smooth-basis functions across multiple elements.

The corner placement strategy proposed in Section 4.2 reduces distortion of the elements, which in turn reduces the distortion of the resulting volumetric B-spline representation. Figure 4.16 visualizes smoothly fitted spline of a rabbit (Figure 4.13b)

and deformed torus (Figure 4.14a) based on the obtained meshes. Compared to the rabbit and deformed torus hex-meshes, both, the average and minimum-scaled Jacobians are improved after B-spline-fitting.

4.5.4 Limitations

A number of limitations exist in the current method. First, the current algorithm cannot extract the 2D skeletal structure from a level set with interior holes, so do the skeletal surface from 3D models with interior boundaries. However, this may be addressed by properly splitting an object into several parts such that each part has a disk-topology. Second, one single sweeping direction may not be sufficient for complex objects, such as objects that have large scale change (e.g., the blade Figure 4.12b) over the sweeping direction and objects that have n-way symmetry. While this may be addressed by decomposing the complex objects into several components, each of which can be meshed separately, stitching the hex-meshes of the individual components together will need to take care of the transition of the topology change of hex-mesh structures. This is beyond the scope of the work in this chapter.

Third, the current pipeline places only four corners on the boundary of each 2D slice. This may not be sufficient for models whose cross sections possess more than four feature points, e.g., the fandisk.

Fourth, the manual design of a harmonic field for a complex object can be challenging. If extrema are not well placed, the induced harmonic functions will lead to highly distorted hex-meshes (Figure 4.17). Combining the information from shape

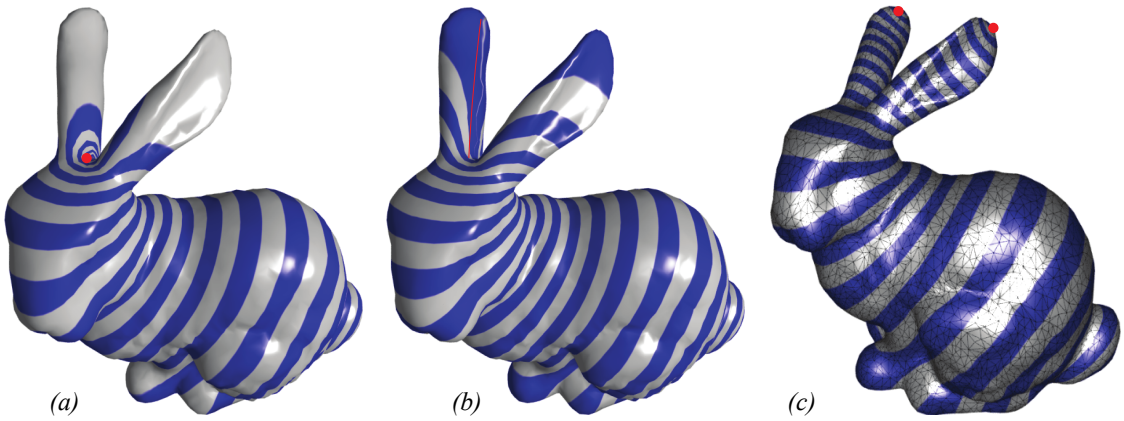


Figure 4.17: For a bunny model, harmonic functions that with extrema at (a) a point and (b) a curve connecting two ears, will result in hex-meshes with much larger distortions, comparing to the hex-mesh (Figure 4.10) guided by a harmonic function (c) that place extrema at the two ear tips.

analysis may help. Nonetheless, this information should only be used to assist the user selection of the harmonic field rather than providing an optimal answer, as the optimal-harmonic field direction may not be desired for the specific application.

4.6 Conclusion

In this chapter, I introduced a volumetric spatial partitioning strategy based on the construction of an inner-skeletal surface. This skeletal surface was computed via a sweeping strategy which was determined by a user-specified harmonic field. The gradient of this harmonic field provided the direction of the subsequent parameterization and hexahedral elements. A number of 2-manifold-level sets were extracted from this 3D harmonic field. A 2D inner-skeletal structure was extracted for each level set. These 2D inner structures were then matched over adjacent-level sets to form

the inner-skeletal surface which partitions the volumetric space into cuboids. Consequently, a 3D parameterization with large structure was derived. We demonstrated our method on a variety of 3D objects.

Compared to existing methods, the hex-meshes generated by our method typically had simpler structure, which was helpful in applications in graphics and engineering (e.g., [41]). However, achieving coarse structure typically results in parameterizations with distortions. This can be addressed by refining the structure locally as needed using the techniques presented in [51]. In our opinion, refining a coarse structure to achieve the tradeoff between the number of singularities and distortion of elements is easier to control than coarsening a fine structure to achieve the desirable structure. Another unique characteristic of our method was that it offers the user to specify the orientation of the generated hex-meshes globally, while existing methods provided only local control [52]. I believe our method enriches the existing tool box for hex-meshing.

In addition, the computational time was a major bottleneck of our pipeline. In the future, I plan to exploit parallel computing to speed up the computation of our algorithm. I also plan to extend our framework to handle CAD models whose 2D cut planes possess more than four sharp corners as well as determining more advanced decomposition strategies for objects that a single sweeping is not sufficient. I will develop techniques for locally refining the structure of the mesh, i.e., systematically introducing extraordinary points as needed, to reduce distortion in a controllable way. Finally, research into guiding the user to select the appropriate harmonic functions for the given application is important.

Chapter 5

Correcting Structure Misalignments for Hex Re-meshing

Although the criteria for evaluating the quality of hex-meshes are application dependent, hex-meshes that conform to boundary surface, preserve surface features, have regular distribution of parameterization lines, and have low element distortion, are generally preferred [63, 28]. It is noteworthy that hex-meshes with the above desired characteristics usually require a certain number of well-placed singularities that are distributed on either the boundary or in the interior of the volume [69, 39, 52, 43].

Given the same set of singularities, without careful control, it can often result in a partition with a large number of components (Figure 5.1(a)). A similar issue has been discussed in quadrangulation applications [66, 8, 89], and is attributed to the *misalignment* of singularities. [52] also pointed out a similar issue in hex-meshes. To the best of our knowledge, an effective and automatic pipeline has not yet been

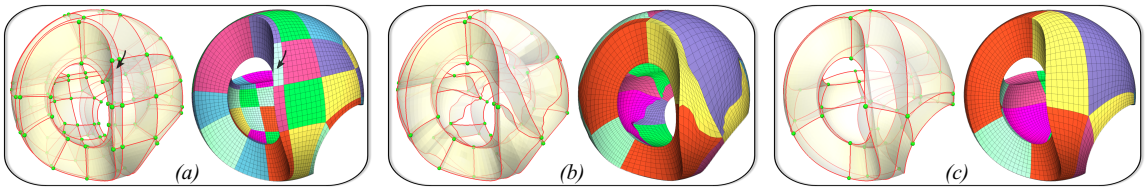


Figure 5.1: (a) An input hex-mesh [52]: The image on the left shows its base-complex that partitions the hexahedral mesh into different large components, illustrated with different colors on the right. Due to the misalignments between singularities, many (typically small) components arise. For instance, a strip of small components near the sharp feature is highlighted. (b) Our alignment algorithm reduces the complexity of the base-complex but leads to a hex-mesh with a large distortion. (c) Both the singularity placement and the element quality of the resulting hex-mesh are improved by our structure-aware optimization algorithm.

proposed to reduce the number of components by fixing the misalignment issue of hex-meshes.

In this chapter, I propose the first solution to reduce the number of hexahedral components in a given hex-mesh by procedurally removing the misalignments from its base-complex. Given an initial semi-structured hex-mesh as the input, I first extract its singularity-structure, that is the union of all the singularities, and compute its base-complex based on the algorithm in Chapter 2(Figure 5.1(a)). Second, I identify the *misalignment candidates* from the obtained base-complex. Each misalignment candidate characterizes a pair of misaligned singularities based on a concept, called *component sheet* that is adapted from the previously introduced hexahedral sheet [10]. These misalignment candidates are a subset of all the misalignments in the base-complex. I then develop an effective and automatic framework to procedurally remove these misalignment candidates in order. Since the obtained hex-mesh after fixing the misalignments could be highly distorted (Figure 5.1(b)), I then extend

the parameterization-based optimization from quad domain [89] to hex-meshes to further optimize the placement of singularities and reduce distortion (Figure 5.1(c)).

The main contributions of the work in this chapter are:

1. I define misalignments in 3D base-complex. From them, I define misalignment candidates for subsequent fixing based on the component sheets. Note that misalignment candidates are a subset of all the misalignments in a base-complex.

2. To remove a misalignment candidate, I propose a zig-zagged sheet extraction technique and enumerate all the possible configurations for this extraction. I propose a metric to rank the misalignment candidates, and devise an effective algorithm to remove them in order.

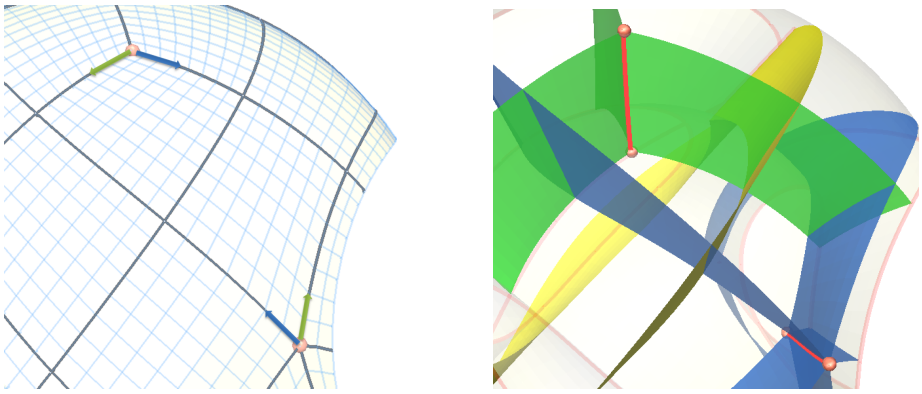
3. Finally, I extend the reparameterization-based optimization for 2D quad-mesh to 3D to improve the quality of the resulting hex-mesh.

5.1 Misalignments in Hex-meshes

In the following, I will first define the misalignments in 3D base-complex, and then introduce an misalignment candidate concept for the subsequent removal of them.

5.1.1 Misalignment Problem

The misalignment problem of quad-meshes is defined in terms of the mismatch of the separatrices of extraordinary (i.e., singular) nodes in surface domain, as illustrated



(a) *(b)*

Figure 5.2: A misaligned singularity pair on surface (a) and in volume domain (b), respectively. Other than the singular nodes and edges (both in red), other intersections shown in these two images are regular elements, i.e., regular nodes and regular edges.

in Figure 5.2(a). Analogously, the misalignment problem of volumetric hex-meshes is defined as the mismatch of the separation sheets of singular edges, as illustrated in Figure 5.2(b). From Figure 5.2 I can see that, the two singularities are not directly connected by a separatrix in 2D or a separation sheet in 3D, resulting in the crossing of two separation structures starting from these two singularities, respectively. This crossing results in non-singular elements (at the intersections of the two separation structures) in the base-complex, creating additional components in $\mathcal{B}$. In other words, misalignments have a close relation to the existence of non-singular elements in $\mathcal{B}$. Thus, reducing the number of components in $\mathcal{B}$ can be achieved by removing the non-singular elements in $\mathcal{B}$. For 3D, while it is beneficial to achieve fewer components in $\mathcal{B}$, a certain number of regular edges are needed to maintain the element quality and geometric fidelity of hex-meshes, for example, the regular edge (black) on the boundary of the model as shown in Figure 5.2(b). In this example, this regular edge (pointed by the black arrow) is the intersection of the surface boundary and

a separation sheet starting from a singularity edge inside the volume. Keeping this regular edge results in a more regular component at the boundary.

To the best of our knowledge, none of previously reported techniques can effectively remove all the misalignments in hex-meshes. In the following, I introduce a concept that enables us to effectively remove a subset of misalignments, referred to as *misalignment candidate* in this chapter.

5.1.2 Misalignment Candidate Detection

Similar to a *hexahedral sheet* [10] that is composed of hex-elements sharing parallel edges, a set of components in $\mathcal{B}_C$ that have parallel base-complex edges form a *component sheet*. The difference between the component sheet and the hexahedral sheet is that: the former is defined over the base-complex, while the latter is defined in terms of the hex-elements in $\mathcal{H}$. Figure 5.3(a) shows a component sheet extracted from the base-complex of a Fandisk model. A component sheet consists of a set of nodes, edges, faces, and components of $\mathcal{B}$. The set of components, i.e., the middle part shown in Figure 5.3(b), are sandwiched by two face sheets (the left and right surfaces in Figure 5.3(b)). In addition to those properties possessed by conventional hexahedral sheets, a component sheet has the following properties: 1) each of its two face sheets must contain at least one singular edge (the red line segments in Figure 5.3) unless it is at the boundary, and 2) some edges in the volume part may be parts of singular edges, e.g., the red line segments in the middle volume of Figure 5.3(b). A limited number of component sheets can be constructed from $\mathcal{B}$ by

grouping sets of parallel base-complex edges (the blue and red edges in the middle of Figure 5.3(b)).

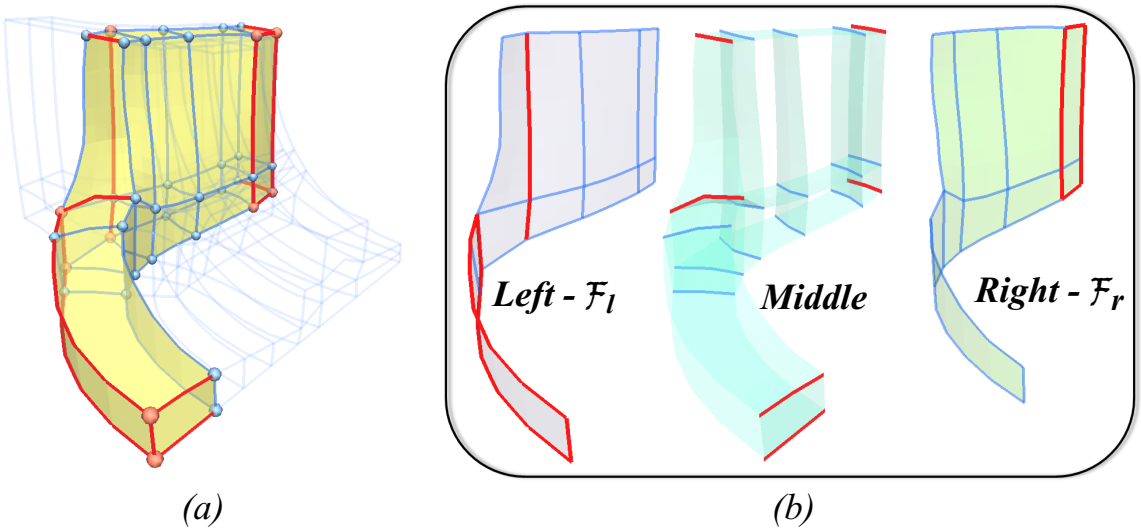


Figure 5.3: A component sheet (a) extracted from the $\mathcal{B}$ of a Fandisk hex-mesh consists of three parts (b): a left face sheet ($\mathcal{F}_l$), a middle volume, and a right face sheet ($\mathcal{F}_r$). Regular and singular edges of the base-complex are shown in blue and red, respectively.

Based on the above component sheet, I define the adjacency among singular edges in $\mathcal{S}$. I call two singular edges neighboring to each other if there is a component sheet containing both of them. Any two singular edges are aligned only when there is a separation sheet directly connecting them. In this case, all the singular edges that are contained in the same face sheet are well aligned. Let's assume that the two face sheets, $\mathcal{F}_l$ and $\mathcal{F}_r$, of a component sheet have a set of singular edges $\mathcal{C}_l$ and $\mathcal{C}_r$ (the red edges in the left and right surfaces of Figure 5.3(b)), respectively. Then, all the singular edges in $\mathcal{C}_l$ are well aligned (Figure 5.3), so do the singular edges in $\mathcal{C}_r$.

A *misalignment candidate* is defined in terms of the two singular edge sets $\mathcal{C}_l$ and

$\mathcal{C}_r$. If none of the singular edges in $\mathcal{C}_l$ is aligned with a singular edge in $\mathcal{C}_r$, then I say the component sheet containing sets of $\mathcal{C}_l$ and $\mathcal{C}_r$ is a misalignment candidate. A component sheet is a misalignment candidate, as long as for every base-complex edge in the volume part of the component sheet: 1) if it is regular, its two end nodes cannot be both on $\mathcal{F}_l$ and $\mathcal{F}_r$ at the same time; 2) if it is singular, it must be part of a singular edge (not the entire singular edge). I can easily find all the misalignment candidates in $\mathcal{B}$ by performing this check. The leftmost image of iteration 1 in Figure 5.5 shows all the detected-misalignment candidates for a Bone hex-mesh. When I visualize the misalignment candidates, I only show the interior parallel edges. Edges belonging to the same candidate are illustrated with the same color.

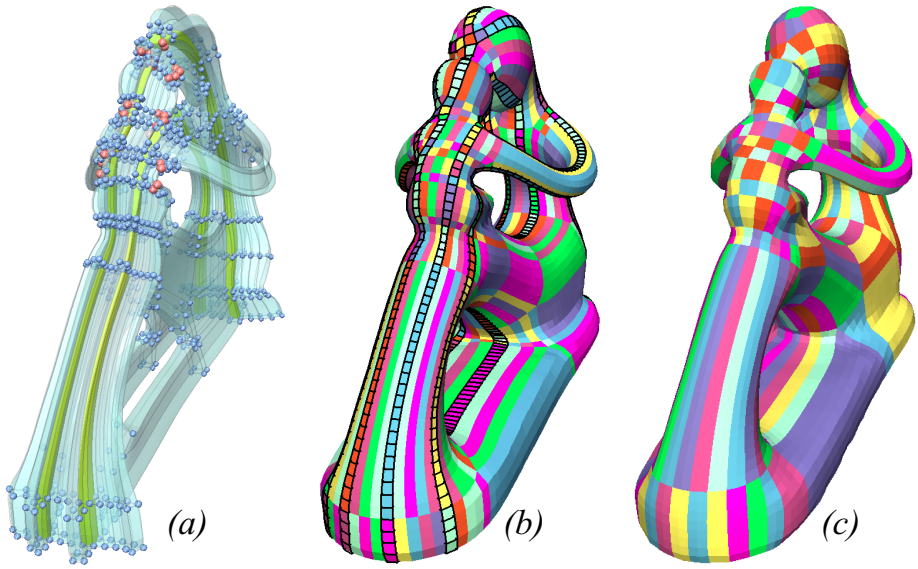


Figure 5.4: (a) A single entangled (crossing components in green) misalignment candidate (singular nodes and regular nodes in $\mathcal{B}$ are in red and blue, respectively), (b) original Fertility hex-mesh embedded with the entangled-misalignment candidate (black), and (c) the optimized Fertility hex-mesh with simplified base-complex.

Note that, the misalignment candidates may be entangled with themselves to form

complicated scenarios; Figure 5.4(a) shows such an example. For simplicity I explain our algorithm using a scenario where the misalignment candidates are not entangled. Entangled-misalignment candidates can also be removed by our alignment algorithm in a similar manner. As an example, Figure 5.4(c) shows the optimized hex-mesh of Fertility that removes the entangled-misalignments existing in Figure 5.4(b).

5.2 Alignment Algorithm

Given an input hex-mesh, I propose to solve the misalignment problem of $\mathcal{S}$ to obtain a simplified base-complex $\mathcal{B}'$ (reducing the number of components in $\mathcal{B}$ while preserving the singularity-structure $\mathcal{S}$).

5.2.1 Pipeline of Alignment Algorithm

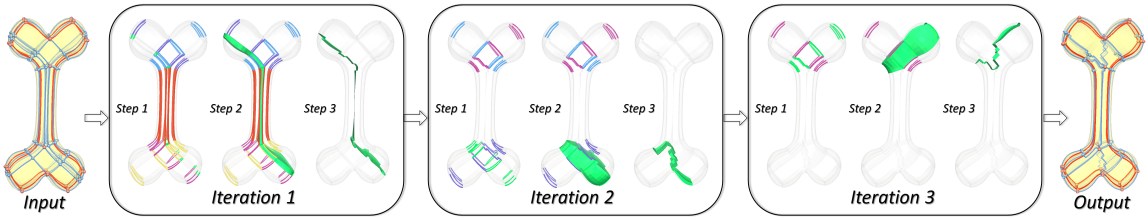


Figure 5.5: Alignment of singularity-structure algorithm is applied to a hex-mesh of Bone. Input: the singularity-structure (red) and constructed base-complex. Within each iteration, execute the following four steps: step 1, detect all misalignment candidates, where every misalignment candidate is represented by parallel edges in the same color; step 2, choose the top-ranked candidate based on the designed metric (green); step 3, the misalignment of singularity chosen at step 2 has been eliminated (sheets) and a new base-complex is constructed; step 4, evaluate the simplified base-complex (the simplified base-complex is valid at all the iterations). Output: the constructed aligned base-complex.

Given $\mathcal{H}$ and its extracted $\mathcal{B}$, our alignment algorithm corrects misalignments via the removal of misalignment candidates–an extension of the previous hexahedral sheet removal process [10], in a greedy fashion as described below:

1. Detect all the misalignment candidates in $\mathcal{B}$ (Section 5.1.2);
2. Rank all the misalignment candidates based on the induced geometric distortion and the reduced number of components via the removal of individual candidates (Section 5.2.2);
3. Correct the top-ranked misalignment candidate to obtain $\mathcal{B}'$ (Section 5.2.3);
4. Verify whether $\mathcal{B}'$ is valid or not (Section 2.2). If it is valid, assign $\mathcal{B}'$ to $\mathcal{B}$ and go to Step 1. Otherwise, re-parameterize $\mathcal{H}$ guided by $\mathcal{B}$ (Section 5.3.1) and go to Step 1.

The above process is repeated until no more misalignments can be found or the number of corrected misalignments meets p, where p is the number of misalignments that users want to remove.

At the above step 4, the resulting $\mathcal{B}'$ may not be valid because only a limited number of hexhedral element layers exist in the volume part of the component sheet of the misalignment candidate. This can be addressed by a re-parameterization step. Explanations will be provided in detail in the topology preservation Section 5.2.3.2. Figure 5.5 shows a pipeline overview of our alignment.

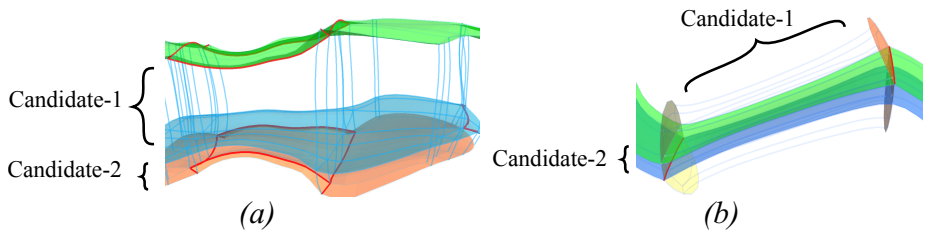


Figure 5.6: Removing one misalignment candidate eliminates the other one as well: (a) two parallel-misalignment candidates, and (b) two orthogonal-misalignment candidates.

5.2.2 Misalignment Candidate Ranking

Since I remove misalignment candidates sequentially, it is important to rank the candidates based on their priorities. The priority value of a misalignment candidate is determined based on the geometric quality of the affected mesh region and the number of reduced components.

In this work, the geometric quality is determined by two factors: the size of one of the two face sheets ($\mathcal{F}_l$ has the same number of quads as $\mathcal{F}_r$) and the width of b_c, i.e., the length of any one base-complex edge in the volume part. I introduce a metric, G_m, to measure the geometric quality of a misalignment candidate, m, as follows:

$$G_m = A_m + \beta {T_m}^2, \tag{5.1}$$

where A_m is the number of quad elements in one of the two side face sheets (Figure 5.3(b)), and T_m is the number of hexahedral edges of any one base-complex edge in the volume part of m. Here β balances the importance of A_m and T_m.

The removal of different misalignment candidates may reduce various number of components. When two or more misalignment candidates are parallel (sharing one face sheet, Figure 5.6(a)) or orthogonal (having crossing face sheets, Figure 5.6(b)) to each other, removing these candidates in different orders may lead to distinct numbers of components in $\mathcal{B}'$. This is because in either a parallel or an orthogonal case, correcting one misalignment candidate would align some singular edges at the two face sheets of the other candidate. In other words, eliminating one candidate may remove some other candidates at the same time. Therefore, it is natural to rank the candidate that achieves larger number of component reduction higher to expedite the alignment process. Considering this and combining with the geometric quality metric, I introduce the priority metric for a given misalignment candidate m:

$$W_m = \alpha G_m + N_{\mathcal{B}'}, \tag{5.2}$$

where $N_{\mathcal{B}'}$ is the number of components of the base-complex after correcting m. Here α balances the importance of the geometric deficiency of $\mathcal{H}$ and the simplicity of its base-complex. I use $\alpha = 1$ and $\beta = 6$ for all our experiments.

5.2.3 Misalignment Candidate Correction

The removal of a misalignment candidate is performed by merging its two side face sheets into one, which turns a component sheet into a face sheet. In this way, the two singular edge sets in the misalignment candidate can be aligned. This can be achieved by simply averaging the two face sheets of the misalignment candidate to obtain a

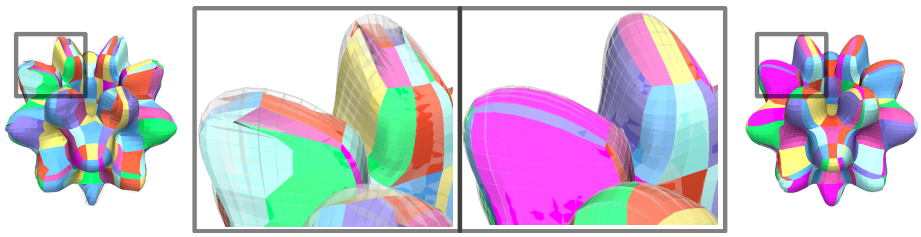


Figure 5.7: Aligned hex-meshes by (a) simply averaging the left- and right-face sheets of misalignment candidates without preserving the surface features, and (b) our approach.

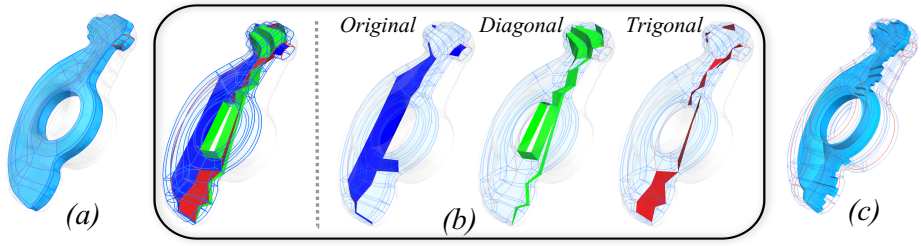


Figure 5.8: (a) A misalignment candidate for Bone (top) and Rockerarm (bottom) hex-meshes, respectively; (b) computed-patch configurations; and (c) the extracted-surface sheets.

single face sheet. While this interpolation procedure is ideal for some circumstances, it could result in hex-meshes with missing geometric features on boundary areas (Figure 5.7(a)), which will be difficult to recover by optimization.

To 1) preserve the boundary geometry of the hex-meshes after alignment and 2) ease the subsequent structure-based global optimization (see Section 5.3), I introduce a novel strategy to remove a misalignment candidate by directly connecting its two sets of singularities using a face sheet in a zig-zag manner. The resulting face sheet will traverse through the involved components of the misalignment candidate. To determine the patch of this surface sheet within each component, I proceed as follows

(Figure 5.8). First, for each component, I determine its configuration for the patch extraction (Section 5.2.3.1, Figure 5.8(b)). Second, based on this configuration, I further extract the patch using the elements of $\mathcal{H}$ (Section 5.2.3.1, Figure 5.8(c)).

As shown in the component at the left-top corner of Figure 5.2.3.1, let's assume a component of $\mathcal{B}$ has three principal directions, i.e., i, j, k. Its nodes, edges, and faces are in the form of $v_{i,j,k}$, $\varepsilon_{i,j}$, and τ_i, where $\varepsilon_{i,j}$ connects nodes $v_{\mathbf{i},\mathbf{j},k}$ and $v_{\mathbf{i},\mathbf{j},k+1}$, and τ_i denotes the face on the i-th plane.

5.2.3.1 Face Sheet Configurations

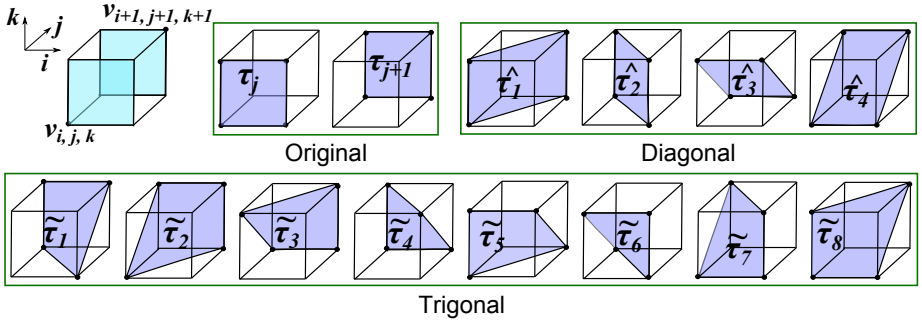


Figure 5.9: A patch at the left-top corner can be extracted from a component in three distinct cases: original (two configurations), diagonal (four configurations), and trigonal (eight configurations).

To extract the face sheet, $\mathcal{F}'_m$, from m, the following constraints are enforced: 1) if one or more edges of a patch of $\mathcal{F}'_m$ are singular, then these edges should remain unchanged; otherwise, the aligned base-complex would not sit in the original hex-mesh any more, and 2) any two neighboring patches of $\mathcal{F}'_m$ are connected via only one base-complex edge. Conditions 1 and 2 are needed to achieve a valid $\mathcal{B}'$.

For a component b_c in m, as shown in the left-top corner of Figure 5.9, I assume

the base-complex faces τ_j and τ_{j+1} belong to the two face sheets, $\mathcal{F}_l$ and $\mathcal{F}_r$, respectively. A patch of $\mathcal{F}'_m$ can be extracted from b_c in three topologically distinct cases: *original* (6 in total), i.e., four nodes are on the same face of b_c, *diagonal* (6 in total), i.e., two nodes on one face and the other two nodes on another face, and *trigonal* (24 in total), i.e., three nodes on one face and the fourth node on another face. For each of the three cases, I need to filter out those configurations that containing the base-complex edges of b_c that belong to the volume part of m (e.g., the parallel edges in Figure 5.3(b), middle). Otherwise, the subsequent $\mathcal{B}'$ will not be valid after a collapse process described in Section 5.2.3.2. As shown in Figure 5.9, the valid configurations (14 total) are: two *original* faces (τ_j and τ_{j+1}), four *diagonal* faces ($\hat{\tau_1}$, $\hat{\tau_2}$, $\hat{\tau_3}$, and $\hat{\tau_4}$), and eight *trigonal* faces ($\tilde{\tau_1}$, $\tilde{\tau_2}$, $\tilde{\tau_3}$, $\tilde{\tau_4}$, $\tilde{\tau_5}$, $\tilde{\tau_6}$, $\tilde{\tau_7}$, and $\tilde{\tau_8}$).

To maintain conditions 1 and 2, I need to determine the configuration type of each patch of $\mathcal{F}'_m$, i.e., how the four corners of the quad patch are extracted from a component b_c. To achieve this, I first initialize all the patches in $\mathcal{F}'_m$ with all the 14 configurations. Then, for each patch, I applied conditions 1 and 2: I retain the configurations that contained parts of singular edges as well as the configurations that share the same edges with those of its neighboring patches; other configurations were removed. Besides, a global update of the configurations of all the patches was needed. This is because removing invalid configurations of a patch may invalidate some configurations of its neighboring patches. In most cases, the above conditions were not sufficient to ensure every patch had only one configuration left. As such, I randomly choose one configuration of a patch that had multiple configurations remaining, and updated the configurations of its neighboring patches accordingly. By

repeating this process, a valid configuration of $\mathcal{F}'_m$ was constructed (Figure 5.8(b)).

5.2.3.2 Face Sheet Extraction

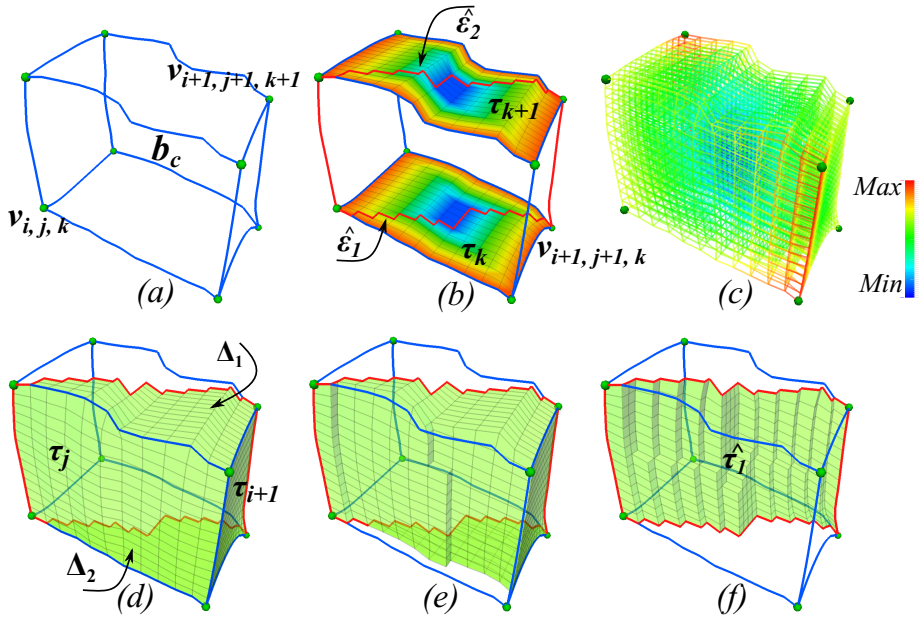
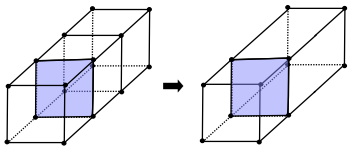


Figure 5.10: A base-complex face with *diagonal* configuration can be constructed from a component (a), by first extracting the two new base-complex edges (b); second, peeling hex-elements gradually from the surface (d) to the interior (e) by measuring the weights calculated for them (c). The constructed sheet is shown in (f).

After determining the configuration for each patch, I now describe the construction of a patch from its corresponding component b_c using elements of $\mathcal{H}$. Based on all the three topologically distinct cases of the configurations, this patch will be either set as one of two original faces of b_c, i.e., the configurations shown in the top row of Figure 5.9, or newly extracted from b_c, i.e., the other configurations in Figure 5.9. In all cases, new base-complex nodes will be chosen from $\mathcal{B}_V$. Specifically, I process each configuration as follows.

***Original* configuration:** As shown in the original configurations of Figure 5.9, I just use either τ_j or τ_{j+1} of b_c. Without loss of generality, let's assume τ_j is selected, if τ_{j+1} is contained in a component other than b_c, then that component will be merged with b_c to form one component in $\mathcal{B}'$ (see the inset). Otherwise, τ_{j+1} is on the surface boundary of $\mathcal{H}$, I simply discard the b_c component and as a result, τ_j will be on the boundary.



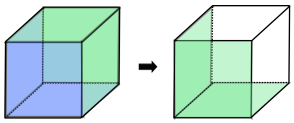
***Diagonal* configuration:** Let's assume $\hat{\tau_1}$ is selected (the first configuration of the diagonal case in Figure 5.9), I then describe how to extract it as follows, illustrated in Figure 5.10.

In the example shown in Figure 5.10, first, I extract the two new edges, $\hat{\epsilon_1}$ from τ_k and $\hat{\epsilon_2}$ from τ_{k+1}, respectively. Let us take the extraction of $\hat{\epsilon_1}$ as an example. Consider τ_k that consists of points and edges as an undirected graph, then $\hat{\epsilon_1}$ is the shortest path from $v_{i,j,k}$ to $v_{i+1,j+1,k}$ A point inside τ_k has a large weight when it is close to the boundary of τ_k. The $\mathbf{k}_{th}$Dijkstra algorithm [96] is used to compute this path. The standard Dijkstra algorithm cannot be applied because its computed shortest path would be problematic when three edges on the shortest path share the same hex-element in $\mathcal{H}$. In this case, a quad of τ_k will be contained in the patch that is to be extracted. Then, there could be a quad belonging to two base-complex faces in $\mathcal{B}'$ when two neighboring components are merged.

Second, with the four edges extracted for $\hat{\tau_1}$ (Figure 5.10(b)), I find its interior

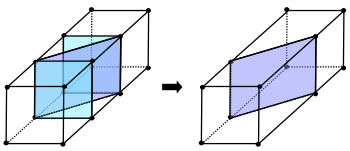
quads that are located inside b_c. A straightforward yet inefficient way to extract $\hat{\tau_1}$ from b_c is to exhaust all the scenarios in a brute force fashion. I now describe our algorithm to obtain the optimal $\hat{\tau_1}$. As shown in Figure 5.10(c), I first calculate a scalar function over the whole volume of b_c. The scalar value at a vertex v in b_c is calculated using $s(v) = d_{2e}(v) - d_c(v) - d_e(v) - d_f(v)$, where $d_{2e}(v)$, $d_c(v)$, $d_e(v)$, and $d_f(v)$ are the minimal distances from v to the two newly extracted edges $\hat{\epsilon_1}$ and $\hat{\epsilon_2}$, to the eight corners, to the twelve edges, and to the six patches of b_c, respectively. The distances are the numbers of hexahedral edges calculated through a bread-first search. The weight for each quad within b_c was then computed by averaging its four vertices' scalar values. As shown in Figure 5.10(c), the weights became smaller from the corners corresponding to $\epsilon_{i,j+1}$ and $\epsilon_{i+1,j}$ to the center, as the color is changed from red to blue.

Third, the interior of $\hat{\tau_1}$ can be obtained by a hex-element peeling process. The surface of b_c is separated into two parts based on the boundary of $\hat{\tau_1}$. Either of them can be used for initialization. Here, I initialize $\hat{\tau_1}$ as the partial surface consisting of faces τ_j, $\tau_i + 1$ and patches Δ_1, Δ_2 (Figure 5.10(d)). For a quad within b_c, if it is on $\hat{\tau_1}$ then I mark it as a boundary quad; otherwise, I mark it as an interior quad. All the hex-elements in b_c are marked as valid. Next, I check each valid hex-element, h, that has a boundary quad. As shown in the inset figure, if the sum of the weights of all the boundary quads of h (green, left) is larger than the sum of the weights of all the interior quads (blue, left), then I mark all the original boundary quads as invalid (white, right), set the original interior quads as new

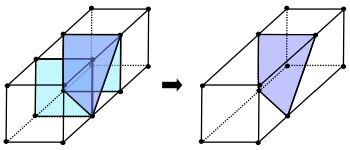


boundary quads (green, right), and label the hex-element as invalid. By repeating the above process, $\hat{\tau}_1$ is continuously moving towards the center of b_c. When all the valid hex-elements with boundary quads do not change their states any more, the final $\hat{\tau}_1$ is reached. Figure 5.10(e) shows an intermediate state of $\hat{\tau}_1$ during the above peeling. The final extracted $\hat{\tau}_1$ is shown in Figure 5.10(f).

After extracting $\hat{\tau}_1$, to maintain the cuboid property of $\mathcal{B}'$, some nodes, edges and faces of the neighboring components (either one or two) of τ_j and τ_{j+1} have to be reconstructed. As a result, the volumes of these components are expanded. The right inset shows such an example.



***Trigonal* configuration:** A patch can be extracted from b_c by following a similar process as detailed for the diagonal configuration. Assume $\tilde{\tau}_1$ is selected (the first configuration of the trigonal case in Figure 5.2.3.1), the two diagonal edges of $\tilde{\tau}_1$ need to be extracted. The boundary of $\tilde{\tau}_1$ separates the surface of b_c into two parts. One part consists of a face and two triangular patches, while the other part consists of three faces and two triangular patches. The interior of $\tilde{\tau}_1$ can be extracted by first intializing it using either part and then performing the peeling process. Similarly, the connectivities of the neighboring components of $\tilde{\tau}_1$ need to be modified. The right inset shows such an example.



Topology preservation: During the zig-zag face extraction step, while the *original*

configuration works perfectly well under all the cases I tested, there is a constraint for both the *diagonal* and *trigonal* configurations. If the configuration of a component is *diagonal*, then two of its three principal directions must have a resolution of at least two hexahedral elements; if the configuration of a component is *trigonal*, then all of its three principal directions must have a resolution of at least two hexahedral elements. The reason is that, for the diagonal configuration, the normal of the extracted patch (i.e., topologically equivalent to a plane) is perpendicular to one of the three principal directions, while for the trigonal configuration, it is neither parallel nor perpendicular to any principal directions.

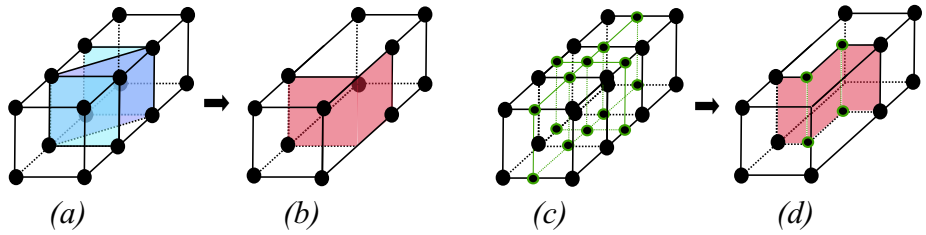


Figure 5.11: (a) A diagonal configuration within a component is comprised of only a single hex-element. The two light blue quads belong to $\mathcal{F}_l$ and$\mathcal{F}_r$, respectively; (b) the actual extracted patch that would lead to an invalid $\mathcal{B}'$; (c) the same component with additional refined hex-elements, and (d) the resulting correct patch from (c).

Figure 13 illustrates a potential problematic scenario when the resolution of the hex-elements within a component is insufficient. Figure 13(a) shows a diagonal configuration that consists of one single hex-element enclosed by two blue quads. The two quads are parts of the left and right surfaces of the component sheet. Figure 13(b) shows the actual extracted patch (red), which would lead to an invalid $\mathcal{B}'$ after collapse. Figure 13(c-d) illustrate that this issue can be easily resolved by inserting additional hex-element layers.

The above mentioned constraint can be satisfied through the re-parameterization (see Section 5.3.2) of $\mathcal{H}$ to increase the hex-element layers in the misalignment candidates, as described in Step 4 of our misalignment algorithm.

5.3 Parameterization and Optimization

After removing misalignments in $\mathcal{B}$, I obtain a simplified base-complex $\mathcal{B}'$. It has fewer components with the cost of certain geometric artifacts. Furthermore, the global orientation of $\mathcal{B}'$ may not be aligned with the surface features any more. To improve $\mathcal{B}'$ and the subsequent $\mathcal{H}'$, I propose a structure-aware re-parameterization algorithm.

5.3.1 Component-wise Volume Parameterization

Based on the aforementioned misalignment correction algorithm, components $\mathcal{B}'_C$ in base-complex $\mathcal{B}'$ are still valid cuboids with certain distortion (Figure 5.5(b-g)). Each hex-element of $\mathcal{H}$ belongs to only one cuboid in $\mathcal{B}'_C$. A cuboid can be mapped (or projected) to an axis-aligned cube domain (x, y, z), where $x, y, z \in [0, 1]$. The cuboid can then be re-meshed into a tetrahedral mesh by subdividing each hex-element into five tets [34]. The parameterization of a cuboid proceeds by assigning parametric values to corners, to the intermediate vertices via linear interpolation of the two end corners of a base-complex edge, to the six patches, and finally to the interior of volume, respectively. Each patch can be mapped to a planar rectangular

parametric-domain. Mean-value coordinate techniques for 2D [25] and 3D [44] are used to calculate the parametric coordinates of vertices inside patches and inside the volume, respectively. Based on the neighboring information of cuboids in $\mathcal{B}'$, the parametric positions of the corners, edges, and patches only need to be calculated once.

5.3.2 Discretization of Parameterization

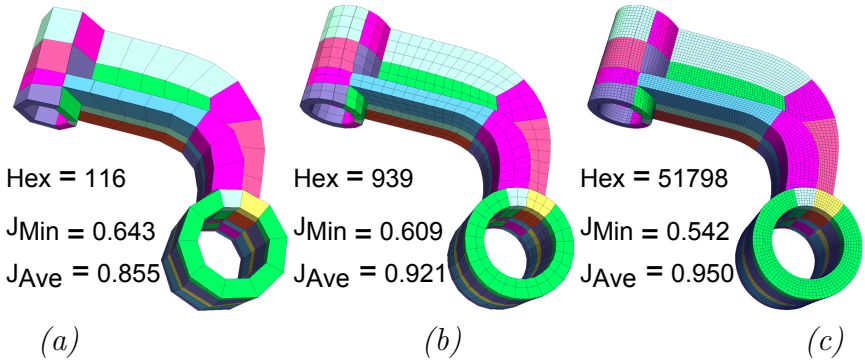


Figure 5.12: Multi-resolution hex-meshes can be generated by only providing different total numbers of hex-elements. The user-specified element numbers are: (a) 100, (b) 1000, and (c) 50000, respectively.

Typically, the parameterization resolution is determined by a scalar value, e.g., the average edge length. However, it is impossible for users to know beforehand how many elements will be produced. Given the desired number of elements as the input, $N_{\mathcal{H}}$, I present a strategy to re-parameterize the mesh with the element number reasonably close to $N_{\mathcal{H}}$. Recall that the resolution of a hex component was determined by three directions, i.e., the resolutions of its edges in i, j, k directions. Therefore, as long as I can determine the resolution of each base-complex edge, the total element number of the final hex-mesh can be soundly estimated. For each

component sheet, the base-complex edges in its middle part are expected to maintain the same resolution. For all the n_c base-complex edges in the same component sheet, I calculate a representative length, δ, for them as follows.

$$\delta = \frac{\sum_t \frac{l_{E_t}}{l_{\bar{e}_t}}}{n_c} \tag{5.3}$$

Here, δ is the sum of all the lengths of the base-complex edges (l_{E_t}) in the same component sheet divided by the average edge length ($l_{\bar{e}_t}$) of the original mesh. The total element number of the resulting hex-mesh should be $\sum w^3 \delta_i \delta_j \delta_k = N_{\mathcal{H}}$, which is the sum of the elements in all the components in $\mathcal{B}$, and w is a scalar weight, defined bellow.

$$w = \sqrt[3]{\frac{N_{\mathcal{H}}}{\sum \delta_i \delta_j \delta_k}} \tag{5.4}$$

Thus, to produce a hex-mesh with $N_{\mathcal{H}}$ elements, each base-complex edge belonging to the same component sheet should contain $w\delta$ hex-edges. Figure 5.12 shows an example of multiple resolution hex-meshes given different user-specified numbers of elements.

5.3.3 Global Optimization

The initially re-parameterized hex-mesh after misalignment correction may contain a large distortion (see Figure 5.13(a)), which is less suitable for downstream applications. Therefore, to obtain a high-quality hex-mesh, a post-processing step such as optimization has to be employed. For certain cases, commonly used optimization

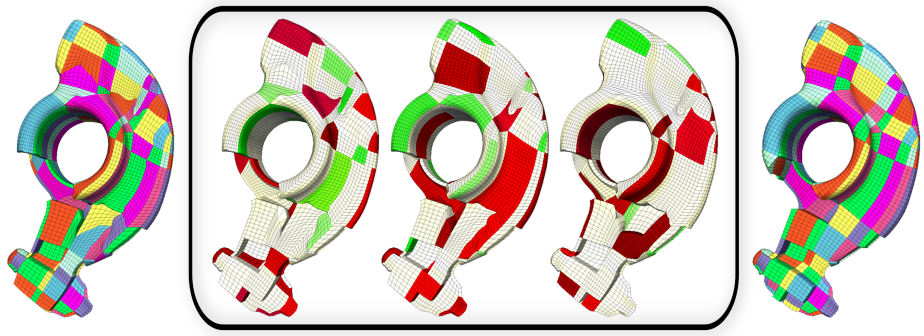


Figure 5.13: (a) The re-meshed hex-mesh after alignment for a Rockerarm model before optimization. Different components are colored differently. (b). Different sets of parameterization domains for each iteration. Isolated edge and face domains are in red and green colors, while domains colored in white are fixed for current iteration. (c). The optimized hex-mesh.

techniques, such as different Laplacian smoothing and geometric flow [97], may optimize some interior vertices onto the boundary, which would pose challenges for the subsequent untangling process [12]. The reason is that after alignment, singularities may not be at their optimal locations any more. To improve the quality, I perform a structure-aware global parameterization over the base-complex domain $\mathcal{B}'$ of $\mathcal{H}'$, as detailed below. Our method efficiently optimizes the base-complex structure $\mathcal{B}'$, while ensuring that the geometrically and topologically interior nodes in the input volume remain inside after parameterization. The local quality of the generated hex-mesh from $\mathcal{B}'$ is improved consequently.

For each interior-base complex patch and edge, I build a parameterization domain, as shown in Figure 5.14. This strategy is inspired by [22, 73, 89], which introduced the interpolation-space parameterization for 2D surfaces. Specifically, for a patch, the parameterization domain stitches its two neighboring hex-components together

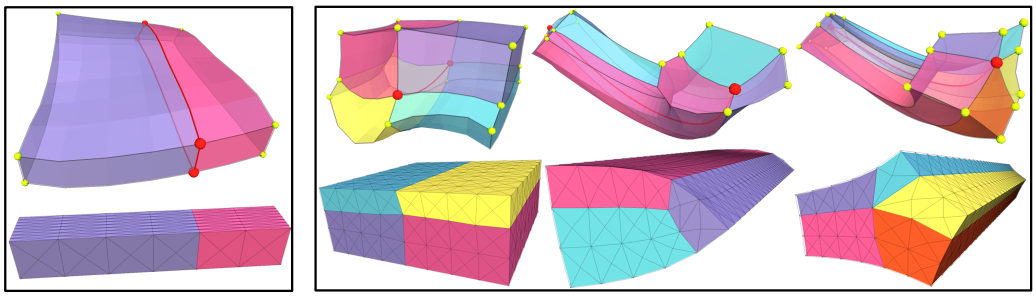


Figure 5.14: Base-complex patch based (left) and base-complex edge based (right) parameterization domains. Domains shown in (b) have valence 4, 3, and 5, respectively.

to form a larger component, which can be mapped to a regular cuboid. For a base-complex edge $b_{e_i} = (b_{v_i}, b_{v_j})$ with valence n, its adjacent n components form a unified-parameterization domain. In the parameterization domain, let k axis points at the direction of b_{e_i}, and the i, j axes are perpendicular to b_{e_i}. Origin $O(0, 0, 0)$ of the parameterization domain is set to be at b_{v_i}, so that b_{v_j} will be mapped to $q(0, 0, k)$. For each ij layer, its parametric coordinates are expressed in the form of polar coordinates (ρ, θ), and transformed to $(\rho^t, t\theta)$ via the exponential map, where $t = 4/n$.

Through face- and edge-based domain re-parameterization, distorted-base complex faces and edges can be optimized. With the above parameterization domains, I perform our global optimization iteratively, i.e., edge- and patch-parameterization domains are executed, alternately. At each iteration, parameterization domains are set to be isolated from each other, as shown in Figure 5.13(b), thus parallel techniques can be employed to significantly speed up the optimization performance.

Surface feature preserving: Typically no feature exists inside the volume unless I define one. In this work, I only tackled the isotropic volume space; thus, only boundary features were considered. Surface feature can be preserved by projecting specific parameterizaiton lines onto the detected features of the boundary surface. After global optimization, the hex-mesh was further improved using the Mesquite software [12].

5.4 Results

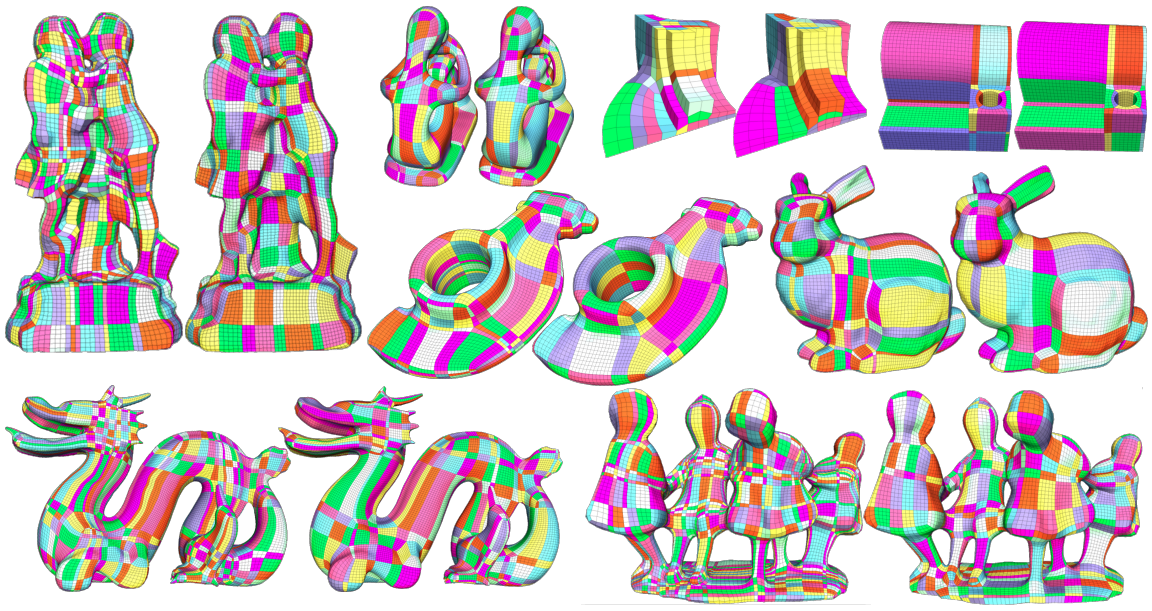


Figure 5.15: The hex-meshes (from left to right and top to bottom) of the Kiss [32], bone, fandisk, impeller [52], angel, dragon, dancing children [38], fertility, bunny [55], and rocker-arm [38] before (the left of each model) and after misalignment correction and optimization (the right of each model).

I have applied the proposed approach to several datasets, including hex-meshes

provided by the authors of [32, 52, 55, 38]. The datasets covered a spectrum of mechanical and natural objects, with various complexities. Table 2 provides the statistics of the tested hex-meshes before and after alignment. Specifically, the statistics information provided in Table 5.1 includes the number of hex elements($|\mathcal{H}|$), the detected and corrected misalignment candidates (|m|), the average and minimal Scaled Jacobians (S. J.), and the number of components ($|\mathcal{B}_C|$) in $\mathcal{B}$. I removed all the detected-misalignments for hex-meshes as seen in Table 5.1. The improvement of singularity alignment was measured as the ratio between $|\mathcal{B}_C| - |\mathcal{B}'_C|$ and $|\mathcal{B}_C|$, which was also provided for each mesh (i.e., the AR column in Table 5.1). As seen in the $|\mathcal{B}_C|$ column in Table 5.1, our alignment algorithm can significantly simplify the base-complexes of input hex-meshes, while preserving comparable local element quality (the S. J. column in Table 5.1). Figure 5.15 provides visual results before and after optimization on a number of hex-meshes listed in Table 5.1. Hexahedral elements that belong to the same components of $\mathcal{B}$ were rendered with the same colors. In our experiments, the computational time varied from one minute (e.g., Fandisk model) to half an hour (e.g., Dragon model). All the timing information was recorded on a PC with Intel Xeon (E5-1620) processor and 16 GB memory.

User control: Our misalignment correction algorithm is intuitive for users to control. Although I remove all the detected-misalignments for hex-meshes as seen in Table 6.1, I also allow users to specify parameter, p, to decide how many misalignments they wants to remove. Bumpy torus (Figure 5.16) is used to demonstrate various optimized results controlled by p.

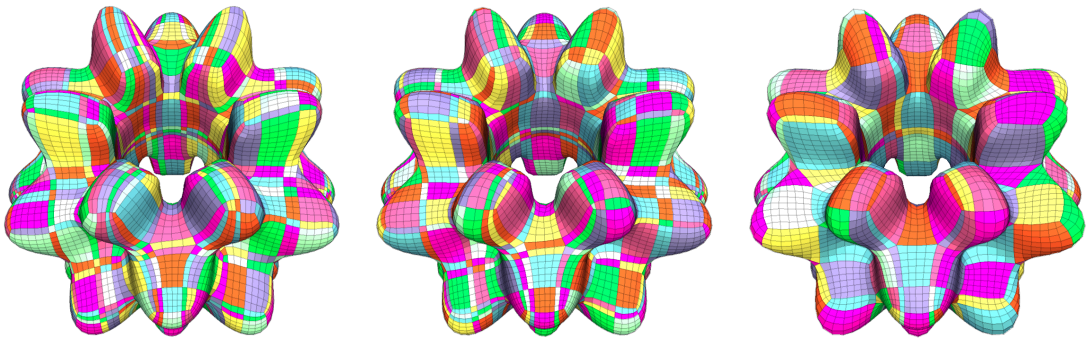


Figure 5.16: p controls the simplicity of the optimized global structure. The larger the p value, the smaller the number of components in the resulting $\mathcal{B}'$.

Adapt to quad-mesh domain: Our alignment algorithm can be straightforwardly adapted to deal with the misalignments in a quad-mesh. The misalignment candidate for hex-meshes defined in Section 5.1.2 was reduced to a restricted poly-chord [20] operating on the base-complex of a quad-mesh. Figure 5.17(a) shows the initially optimized results of the Drillhole and Botijo quad-meshes from [8]. Figure 5.17(b) provides the results by further removing misalignments in the meshes shown in Figure 5.17(a) using our algorithm for quad-meshes with $p = 2$ and 4, respectively. Figure 5.17(c) demonstrates that if too many misalignment candidates ($p = 5$ and 9 for Drillhole and Botijo, respectively) were removed from the base-complex, to maintain the isometry, the edges of the base-complex may not be conformal to the features of the input models. This demonstrated that our alignment algorithm acted as an additional process to further improve the simplicity of quad meshes.

Models	$\|\mathcal{H}\|$	|m|	S. J.	$\|\mathcal{B}_C\|$	AR	Models	$\|\mathcal{H}\|$	|m|	S. J.	$\|\mathcal{B}_C\|$	AR
girl/BU*	193k **44k**	16	.925/.235 **.894/.121**	1098 **401**	63%	Gargoyle♯	26k **23k**	52	.906/.196 **.911/.214**	7563 **1920**	75%
Bumpy-torus* (Fig. 5.7(b), 5.16(c))	35k **35k**	33	.891/.271 **.866/.189**	2518 **590**	77%	Angel-1♯	14k **15k**	11	.923/.470 **.915/.296**	1284 **698**	46%
Bunny*	82k **33k**	7	.930/.138 **.906/.200**	1324 **240**	82%	Angel-2♯	16k **16k**	5	.919/.212 **.914/.260**	302 **196**	35%
Fertility-refine*	20k **22k**	15	.911/.196 **.884/.364**	598 **390**	35%	Angel-3♯	14k **17k**	2	.898/.222 **.867/.157**	78 **56**	28%
Kiss-coarse* (Fig. 5.15)	27k **59k**	24	.901/.163 **.910/.190**	3690 **1365**	63%	Bumpy-torus♯	39k **39k**	23	.929/.335 **.879/.270**	2254 **910**	60%
Bone†	3k **9k**	6	.930/.620 **.924/.577**	87 **48**	45%	Bunny♯	38k **48k**	2	.926/.382 **.937/.373**	273 **221**	19%
Bunny†	134k **96k**	2	.940/.293 **.930/.276**	259 **184**	29%	Bustle♯	12k **22k**	3	.934/.302 **.929/.393**	348 **292**	16%
Rod†	6k **7k**	1	.947/.658 **.937/.527**	66 **33**	50%	Dancing-children♯(Fig. 5.15)	35k **38k**	48	.870/.143 **.876/.190**	5482 **1458**	73%
Sculpture-A†	24k **24k**	6	.961/.689 **.890/.426**	51 **16**	69%	Dragon♯(Fig. 5.15)	118k **113k**	22	.857/.150 **.899/.120**	3977 **1958**	51%
Fandisk†(Fig. 5.15)	.4k **.8k**	3	.936/.609 **.940/.413**	49 **30**	39%	Elephant♯	172k **55k**	26	.878/.221 **.890/.170**	2842 **1008**	65%
Fertility†(Fig. 5.4)	14k **28k**	7	.911/.351 **.88/.300**	1352 **934**	31%	Rockerarm-1♯(Fig. 5.13)	24k **26k**	12	.920/.439 **.890/.329**	686 **395**	42%
Hanger†	5k **9k**	1	.964/.599 **.950/.448**	60 **41**	32%	Rockerarm-2♯	25k **26k**	13	.905/.378 **.899/.366**	835 **263**	69%
Impeller†(Fig. 5.15)	11k **19k**	8	.924/.185 **.925/.238**	944 **399**	58%	Bunny‡(Fig. 5.15)	74k **46k**	13	.938/.274 **.928/.263**	580 **194**	67%
Joint†(Fig. 5.15)	18k **18k**	4	.984/.729 **.983/.724**	83 **59**	29%	Fertility‡(Fig. 5.15)	54k **74k**	11	.872/.259 **.885/.272**	693 **396**	43%
Rockerarm†	11k **18k**	9	.866/.209 **.862/.145**	578 **291**	50%	Girl/BU‡	56k **77k**	9	.926/.401 **.899/.279**	580 **252**	57%
Sculpture-B†	6k **6k**	0	.892/.055 **.889/.049**	51 **51**	0%	Rockerarm‡	57k **57k**	10	.890/.370 **.893/.297**	664 **335**	50%

Table 5.1: The number of components of base-complex and quality comparisons of hex-meshes before and after alignment. For each model, the original hex-mesh and optimized hex-mesh by our method are shown in the upper row and bottom row, respectively. Original hex-meshes with *, †, ♯, and ‡ are obtained from [32, 52, 38, 55], respectively.

5.5 Conclusion

In this chapter, I presented the first framework to optimize the hex-mesh by reducing the number of components in $\mathcal{B}$ in two steps: 1) detect and remove misalignment candidates in $\mathcal{B}$, and 2) improve the geometric quality of a hex-mesh through a structure-aware re-parameterization optimization. The pipeline can be applied to quad-mesh domain as well.

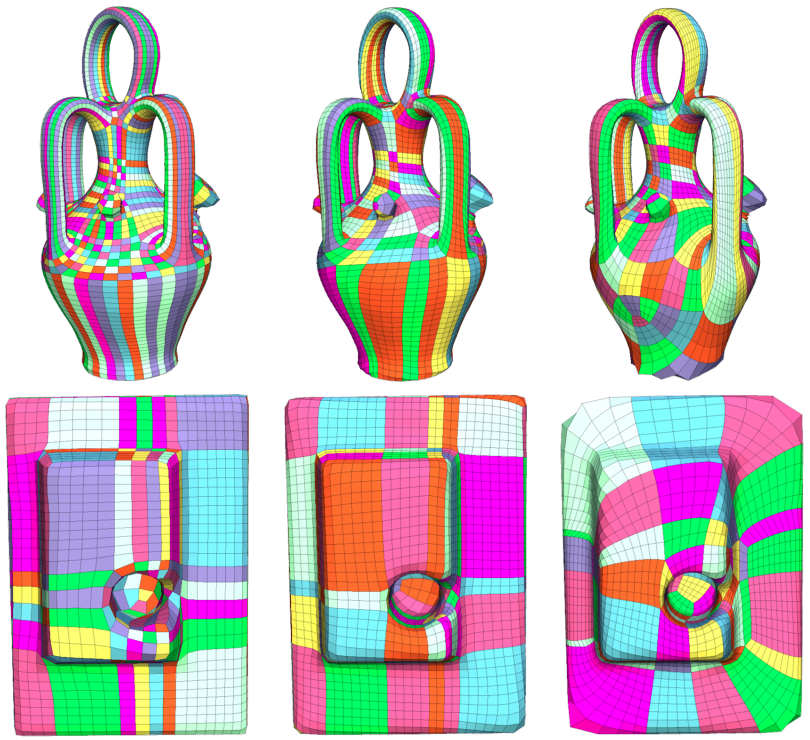
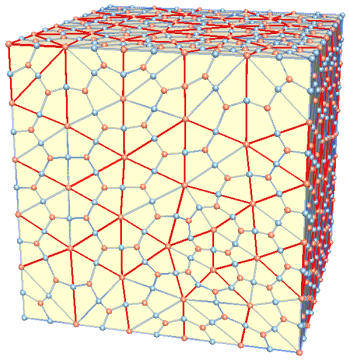


Figure 5.17: Given the optimized quad-meshes generated from [8](a), our method can further reduce the number of patches of the quad-meshes, as shown in (b-c). (c) has the smallest number of patches with the cost of the largest distortion among the three results.

Limitations: Similar to the alignment problem on surface domain, the major limitation of our work was the input hex-mesh, if the singularities were at a higher order, the number of singularities were large, or the singularities were badly placed, its simplified-base complex was still too complicated to be used for some applications. The inset demonstrated such an example, where the hex-mesh is directly subdivided from a tet-mesh. However, the complexity of $\mathcal{B}$ in each tested hex-mesh



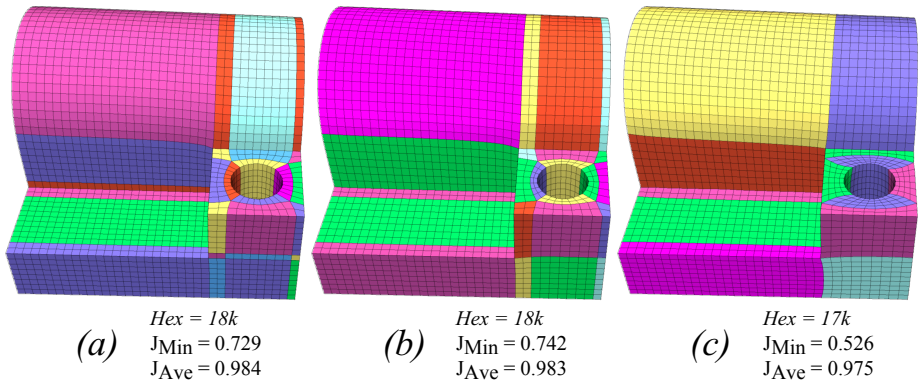


Figure 5.18: Given the original Joint hex-mesh as the input (a), comparing to the base-complex of the mesh aligned by the presented algorithm (b), fewer components can be achieved with the modification of the singularity-structure in the mesh (c).

was reduced, unless the input hex-mesh had an already well aligned singularity-structure, such as the Sculpture-B and Rod models as seen in Table 6.1. In this case, our framework directly performed the optimization step, while the hex-mesh after optimization cannot guarantee a higher geometric quality than the original input.

While our alignment algorithm simplified the complexity of the base-complex of a hex-mesh without changing its singularity-structure, the number of components can be further reduced by allowing the merging of singularities. As an example, the Joint hex-mesh shown in Figure 5.18(c) has fewer components than the one in Figure 5.18(b), which was achieved by allowing the change of $\mathcal{S}$. How to manipulate the singularities of a hex-mesh in a general manner to achieve better results is an interesting topic for future research. Another limitation was that although all the misalignment candidates in the meshes used in this chapter were detected and corrected by our approach, other configurations of misalignment candidates may exist that are not reported here. Furthermore, in theory our approach cannot guarantee

that the optimized base-complex is free of any misalignments. Here, misalignments arise if any component in $\mathcal{B}_C$ contains at least one regular base-complex edge, which is not the same as the misalignment candidates defined in this work. I plan to address these limitations in the future work.

Chapter 6

Simplifying Global Structures for Hex Re-meshing

Up to date, generating usable, high-quality hex-meshes (i.e., with positive minimum-scaled Jacobian) for arbitrary 3D objects remains a challenging task, despite many recent advances [55, 38, 52, 43, 27, 30, 53]. While persistent efforts have been made to focus on effectively generating all-hex-meshes from surface inputs, optimizing the initial obtained meshes with existing hex-meshing techniques is often necessary. This includes the improvement of the quality of individual elements and the simplification of the global structure of the meshes (i.e., reducing the number of irregular elements in the meshes). While the former has been extensively studied [12, 54], there has been lack of attention on the latter.

As discussed in Chapter 1.1, the number of components in the base-complex was used to measure its complexity. A base-complex with fewer components (i.e.,

a simpler structure) is typically preferred by applications that seek a smooth representation (e.g., splines) with high level smoothness throughout the entire volume, such as Isogeometric Analysis (IGA) [4]. However, most existing hex-meshing techniques cannot control the complexity of the structure of the obtained hex-mesh. This requires post-processing to reduce the complexity of the mesh structure for the subsequent applications. Recently, Gao et al. [27] introduced a structure simplification framework for hex-meshes by procedurally aligning the mis-matched singularities (i.e., mis-alignments). However, this process does not remove any singularities from the hex-mesh, which constrains the overall continuity that the corresponding 3D parameterization can achieve. It also greatly limits the level of simplicity that a hex-mesh can achieve, which is related to the number of singularities (see two examples in Figure 6.15).

In this chapter, I propose a new structure simplification strategy for valid all-hex-meshes by procedurally removing the *base-complex sheets* and *base-complex chords* extracted from their base-complexes. Different from the previous hexahedral sheet removal for hex-mesh coarsening [10, 91], our approach simplifies the structure of the hex-mesh explicitly, which coarsens the hex-meshes much faster than individual element removal. Also, in contrast to the mis-alignment problem, the involved singularities may be removed or altered after the removal of the corresponding sheets and chords, which enables us to greatly simplify the structure of the mesh.

Our approach shares some similarities to the quad mesh coarsening approaches proposed by Daniel et al. [20] and Tarini et al. [88] (i.e., the base-complex sheet collapsing vs. the poly-chord collapsing, and the base-complex chord collapsing vs.

the quadrilateral collapsing). However, due to one additional dimension in hex-mesh simplification, there are new challenges that need to be addressed compared to quad mesh simplification. First, the topological configurations of sheets and chords in 3D are far more complex than their 2D counterparts. Second, directly adapting the collapsing operations for element levels [20, 10] to base-complex may result in large geometric distortion and the loss of important surface features due to its coarse representation. Third, to facilitate the subsequent optimization of the hex-mesh with as simple as possible structure, the extracted sheets and chords for removal need to be properly sorted to avoid early termination of the simplification or undesired topological configurations, which has not been addressed by existing ranking strategies.

To address the above new challenges, I first systematically classify the base-complex sheets and chords into different configurations, for which different simplification operators are developed to warrant a topologically correct output (Section 6.1). Second, I propose an effective simplification framework that procedurally removes the base-complex sheets and chords using the developed simplification operators (Section 6.2). A comprehensive ranking metric (Section 6.2.1) is developed for prioritizing operators to achieve efficient reduction of the structure while preserving geometry fidelity. A parameterization-based collapsing strategy is introduced to address the distortion due to the removal of sheets and chords (Section 6.2.2). I have applied our simplification framework to a number of hex-meshes generated with state-of-art hex-meshing approaches to demonstrate its effectiveness (Section 6.3).

6.1 Preliminaries

As shown in Figure 6.1(a-b), for a quad-mesh a sequence of connected quads can be removed simultaneously via a poly-chord collapsing; a quad can be removed via a quadrilateral collapsing (Figure 6.1(c-d)) [20]. Similarly, a sequence of components in the base-complex can be removed together to obtain a reduced base-complex (Figure 6.1(e-h)). While collapsing a poly-chord or a quadrilateral configuration may not reduce (or even complicate) the structure of the mesh (Figure 6.1(a-d)), collapsing a sequence of components in the base-complex will monotonically reduce the complexity of the base-complex. To achieve that, in what follows I describe the 3D counterparts of poly-chord and quadrilateral configurations, which are referred to as the *base-complex sheets* and *base-complex chords*, respectively.

6.1.1 Base-complex Sheet

Recall that the base-complex $\mathcal{B}$ of a hex-mesh $\mathcal{H}$ is a coarse hex-mesh. Therefore, a base-complex sheet is a hexahedral sheet of the coarse hex-mesh defined by $\mathcal{B}$ [27]. Given a base-complex edge e, I locate the other base-complex edges that are all parallel to e and in the components (i.e., cuboids) that have e as an edge. For these new base-complex edges, I similarly locate other edges that are parallel to e in the components adjacent to these new edges. By recursively doing so, a unique set of parallel base-complex edges is identified (Figure 6.2(a)). The components having one of these edges as their edge form a base-complex sheet S.

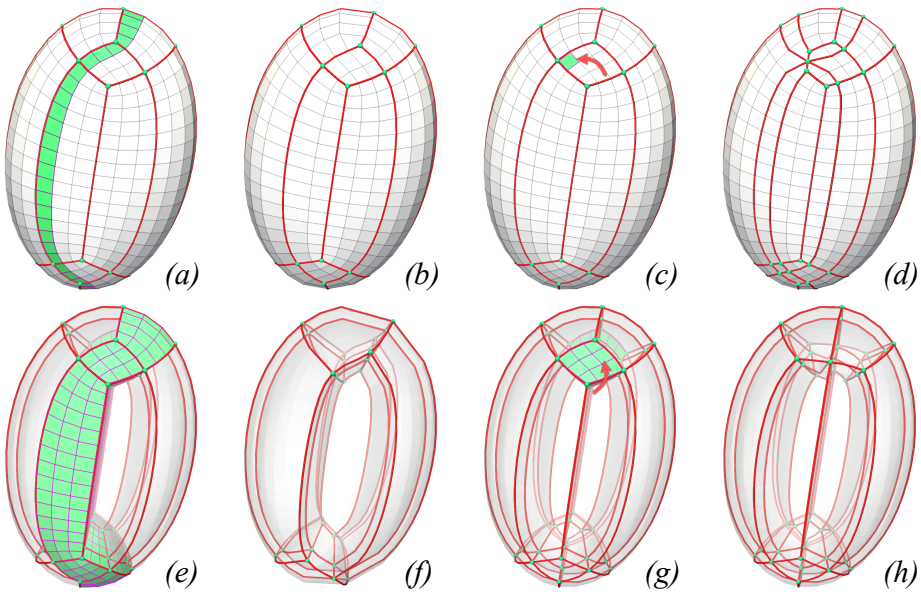
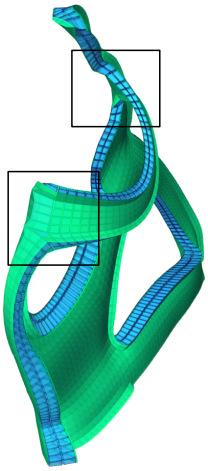


Figure 6.1: For a quad-mesh, removing a poly-chord (green) may not affect its patch layout (a-b), while collapsing a quad may even increase the complexity of the structure (c-d). In contrast, performing either (e) a base-complex sheet or (g) a chord removal on the global structure of a hex-mesh will reduce the number of its components (f, h).



Specifically, $S = \{S_V, S_E, S_P, S_C\}$, where S_V, S_E, S_P, and S_C are the nodes, edges, patches, and components in $\mathcal{B}$ that belong to S, respectively. In particular, edges in S_E can be classified into two groups S_{E_m} (blue edges in Figure 6.2(b)) and S_{E_s} (blue edges in Figure 6.2(c-d)), where S_{E_m} is the aforementioned set of parallel edges and S_{E_s} are the other edges in S. Patches in S_P can be similarly classified into two groups with S_{P_m} (green patches in Figure 6.2(b)) being the patches that are adjacent to the edges in S_{E_m} and S_{P_s} (blue in Figure 6.2(c) and yellow in (d)) being the other patches.

In general, a base-complex sheet can be decomposed into two parts: side surfaces

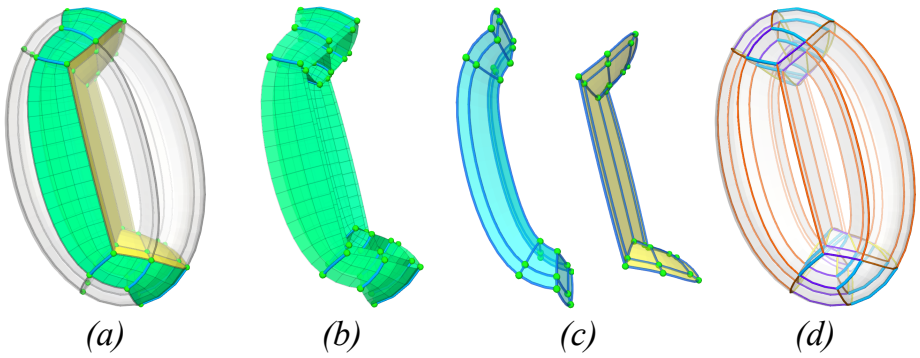


Figure 6.2: A base-complex sheet in (a) can be decomposed into a middle part (b) packed by its two side surfaces (c-d). All the base-complex sheets contained in the model are rendered only by their middle edges in (e).

(Figure 6.2(c-d), containing elements S_V, S_{E_s}, and S_{P_s}) and a middle volume part (Figure 6.2(b), containing S_V, S_{E_m}, S_{P_m}, and S_C). Note that, although the side surfaces shown in Figure 6.2(c-d) consist of two disconnected components (i.e., a left and a right surfaces), there could be cases (as shown in the inset) that the side surfaces form a Mobiüs surface (i.e., the green surface).

By the above definition, a base-complex edge can belong to the set S_{E_m} of only one sheet; a patch can be included in S_{P_m} by at most two sheets; and a component can be shared by at most three sheets. Therefore, a finite set $\boldsymbol{S}$ of base-complex sheets can be extracted from a base-complex. Each base-complex edge has a unique tag, which marks the sheet whose S_{E_m} contains this edge. As shown in Figure 6.2(e), an ellipsoid hex-mesh has a total of six sheets where their corresponding edge sets S_{E_m} are colored differently.

6.1.2 Base-complex Chord

The concept of hexahedral chord has been applied to the hex-mesh matching [84] and editing [37]. Similar to the base-complex sheet, I adapt hexahedral chords for a hex-mesh to its base-complex and define the base-complex chord (Figure 6.3). Given a base-complex patch p, I find its opposite (or parallel) patch in the component that has p as one of its faces. I repeat this search for a newly located patch. By recursively doing so, a unique set of parallel base-complex patches is identified (green patches in Figure 6.3(a)). The components having one of these patches as their face form a base-complex chord C. Specifically, $C = \{C_V, C_E, C_P, C_C\}$, where C_V, C_E, C_P, and C_C are the nodes, edges, patches, and components in the base-complex that belong to this chord, respectively. In particular, C_P consists of those parallel patches C_{P_m} (Figure 6.3(b)) from the above search and the other patches C_{P_s} (Figure 6.3(c)) of C. C_E consists of edges C_{E_m} (blue edges in Figure 6.3(b)) that are the boundary edges C_{P_m} of the patches and the other edges C_{E_s} (blue edges in Figure 6.3(c)).

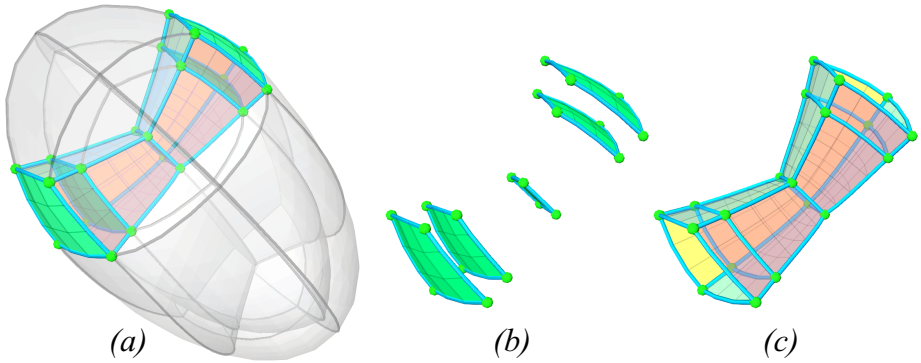


Figure 6.3: A base-complex chord in (a) can be decomposed into a middle part (b), i.e., set of parallel patches, and four side surfaces (c).

Similar to a base-complex sheet, a base-complex chord can be decomposed into

two parts: (four) side surfaces C_{P_s} (Figure 6.3(c), containing elements C_V, C_{E_s}) and a middle volume part (Figure 6.3(b), containing C_V, C_{E_m}, C_{P_m} and C_C). By the above definition, a base-complex patch can belong to the set C_{P_m} of only one chord, and a component can be shared by at most three chords. Therefore, a finite set $\boldsymbol{C}$ of base-complex chords can be extracted from a base-complex. Every patch in the base-complex has a unique tag that marks the chord whose C_{P_m} contains this patch.

Relations between sheets and chords: For the four edges of a patch in C_{P_m} of a chord, they could have either the same tag or at most two different tags of sheets. Thus, all the edges in C_{E_m} could have either the same or two tags for sheets. Therefore, a chord is either fully contained within a sheet (one tag) or a crossing of two different sheets (two tags). Furthermore, for a patch in the set S_{P_m} of a sheet, all its parallel patches that form a chord are contained in set S_{P_m}. Therefore, if there are k distinct chords of the patches in S_{P_m} of a sheet, then the union of the k chords form the sheet. This relation between sheets and chords will be utilized to reduce the complex sheets to simpler ones for subsequent removal (illustrated in Figure 6.11). In the following, I will discuss the possible configurations of sheets and chords, which will affect the subsequent collapsing process for removal.

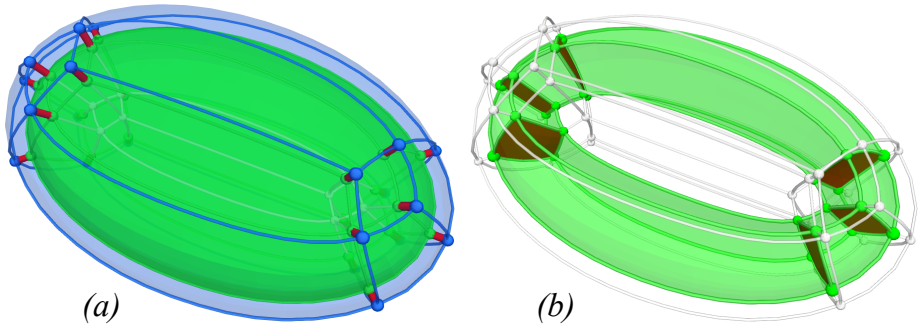


Figure 6.4: A closed base-complex sheet (a) and chord (b), respectively.

6.1.2.1 Base-complex Sheet/Chord Configurations

The base-complex for a hex-mesh is basically a layered structure in multiple directions that is further complicated by various orientations of the singularities. This makes the base-complex sheet is not a trivial extension from the 2D poly-chord to 3D. Based on whether there are boundary edges in S_{E_m}, a sheet S can be either open (Figure 6.2) or closed (Figure 6.4(a)). Similarly, a chord can be either open (Figure 6.3) or closed (Figure 6.4(b)), depending on whether there are patches in C_{P_m} on the boundary surface.

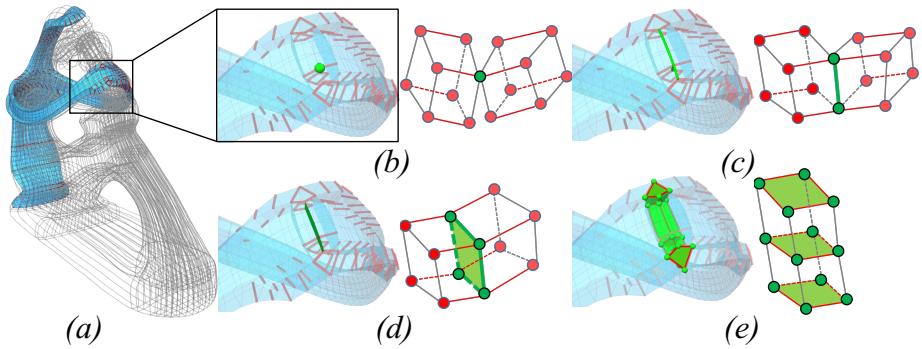


Figure 6.5: (a) A sheet has combined configurations with tangent at node (b), tangent at edge (c), tangent at patch (d), and tangled (e) configurations. For each configuration, the left image is the zoom-outed view of the sheet, while the right image is a simple illustration.

Based on the neighborhood properties of the elements in a sheet, it can be classified as *regular* (all nodes in S_V have only one neighboring edge in S_{E_m}, Figure 6.2), *tangled at components* (i.e., components in S_C whose six patches are in S_{P_m}, Figure 6.5(e)), *tangent at nodes* (i.e., nodes in S_V that have more than one neighboring edges in S_{E_m} yet are not contained in tangled components, Figure 6.5(b)), *tangent at edges* (i.e., edges in S_{E_s} that have more than one neighboring patches in S_{P_s}

yet are not contained in tangled components, Figure 6.5(c)), or *tangent at patches* (i.e., patches in S_{P_s} that have more than one neighboring components in S_C yet are not contained in tangled components, Figure 6.5(d)). A sheet may have both tangent and tangled configurations (Figure 6.5(a)). This will complicate the subsequent collapsing. Note that, based on the relations between sheets and chords described earlier, the tangled components of a sheet form a complete base-complex chord. By removing the base-complex chord at the tangled region, this complex configuration can be reduced to a simpler one that only has tangent configurations, which can be further reduced. More details will be provided later (Section 6.2.2).

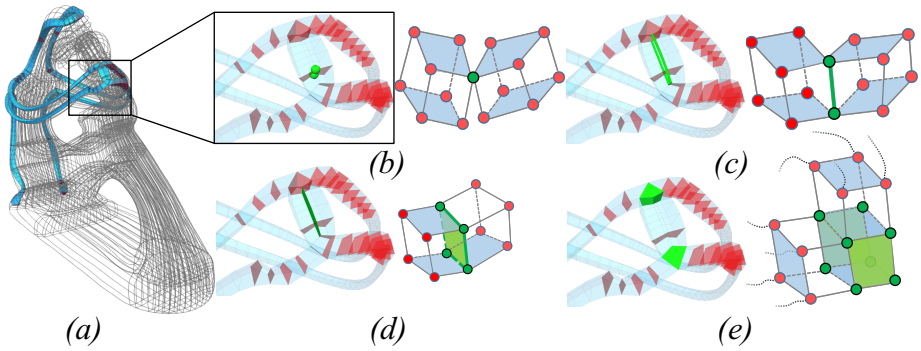


Figure 6.6: (a) A chord has combined configurations with tangent at node (b), tangent at edge (c), tangent at patch (d), and tangled (e) configurations. For each configuration, the left image is a zoom-outed view of the chord, while the right image is a simple illustration.

Similarly, a chord could be *regular* (all nodes are contained in one patch in C_{P_m}, Figure 6.3), *tangled at components* (i.e., components in C_C that have more than four patches in set C_{P_m}, Figure 6.6(e)), *tangent at nodes* (i.e., nodes in C_V that have more than one neighboring patches in C_{P_m} yet are not contained in tangled components, Figure 6.6(b)), *tangent at edges* (i.e., edges in C_{E_s} that have more than one neighboring patch in C_{P_s} yet are not contained in tangled components, Figure 6.6(c)), or

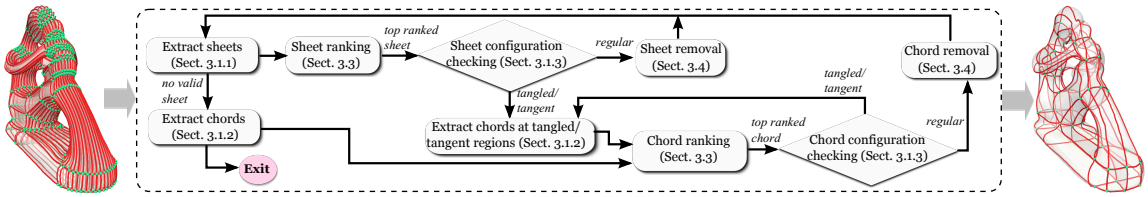


Figure 6.7: Simplification pipeline.

tangent at patches (i.e., patches in C_{P_s} that have more than one neighboring component in C_C yet are not contained in tangled components, Figure 6.6(d)). It also removes all the patches in C_{P_m} while not creating new patches.

6.2 Simplification Algorithm

Given an all-hex-mesh as the input, I first extract its base-complex and the base-complex sheets described above. I then perform the following iterative process (Figure 6.7). First, I prioritize all of the sheets (Section 6.2.1) and identify the top-ranked sheet. If this sheet is regular, I remove it (Section 6.2.2) and process the second top-ranked sheet. If this sheet is tangled or tangent (Section 6.1.2.1), I identify the chords at the tangled or tangent regions of the sheet and rank them. If the top-ranked chord is regular, I remove it (Section 6.2.2) and put the newly obtained sheets to the list of the candidate sheets, then, repeat the above sheet removal process. If the top-ranked chord is tangled or tangent, I identify the chords at the tangled or tangent regions of this chord and rank them. I then attempt to remove the top-ranked chord according to its configuration (Section 6.1.2.1) as above. Second, if there is no sheet available for removal because of the topology constraints (Section 6.2.3), I extract all the

base-complex chords from the current base-complex and rank them. I then attempt to remove the top-ranked chord according to its configuration as above. If there is no chord available for removal, I exit the simplification. Note that to facilitate the subsequent removals, after the successful removal of each sheet or chord, I perform a global re-parameterization of the mesh based on the simplified base-complex to regularize the base-complex components [27].

While it may be easier for the user to specify the desired number of components for the simplified base-complex, the resultant hex-mesh from the simplified base-complex may possess many tangled hex-elements (i.e., with negative Jacobians) that existing optimization techniques cannot completely resolve. To avoid this, I introduce a simple yet effective binary search pipeline that enables us to produce a hex-mesh with desired structure while having high-quality elements to some extent. To achieve that, I first save the simplified result of each iteration as a sequence of hex-meshes. After simplification is terminated, I start to optimize the hex-mesh with the simplest base-complex. If the optimized hex-mesh fails the quality test (e.g., having negative Jacobians), I backtrack to the hex-mesh in the half-way of the simplification sequence. If the obtained hex-mesh is acceptable (e.g., positive minimum Jacobians), I locate the hex-mesh in the mid-way of the second half of the simplification sequence; otherwise, I retrieve it from the first half. I repeat the same process until convergence. For the hex-mesh optimization, I first perform the parameterization-based optimization [27], then apply Mesquite [1] [12] to further improve its geometric quality.

[1]Other optimization techniques can be applied here. However, I found Mesquite is fast, robust and an open source tool that is available for our purpose.

6.2.1 Sheet and Chord Ranking

To improve the topological layout of the structure and maintain its geometric fidelity, I employ a ranking strategy for sheets and chords that takes into account the valence information and the geometric impact of their removal on the resulting mesh.

I first introduce a ranking metric $w(S)$ for a base-complex sheet S as follows.

$$w(S) = \alpha e^{w_g(S)} + \beta e^{w_v(S)}, \tag{6.1}$$

where $w_g(S)$ and $w_v(S)$ measure the volume change and valence change induced by the removal of S, respectively. Empirically, I use $\alpha = 0.9$, $\beta = 0.1$ for sheet ranking in all our experiments. I use the same generic metric for a chord with a different set of parameters, i.e., $\alpha = 0.2$, $\beta = 0.8$. In the following sections, I describe the individual terms in the proposed metric in detail.

6.2.1.1 Geometric Term

To reduce distortion due to the removal, the geometric term prefers a thin sheet with a smaller volume, as described in Equation 6.2.

$$w_g(S) = \sum_{i=0}^{|S_C|} L^{\gamma}(e_i) A(f_i) \tag{6.2}$$

where $|S_C|$ is the number of components in S, $L(e_i) \in S_{E_m}$ is an edge of the i^{th} component that has the longest length in the component, $A(f_i) \in S_{E_m}$ is a patch of the same component that has the largest area, and γ is a constant that I set as 2, which put more weights on the thickness of a sheet.

Similarly, the geometric measurement for a chord is described by Equation 6.3.

$$w_g(c) = \sum_{i=0}^{|C_C|} D_i^{\gamma} A(f_i) \tag{6.3}$$

where $|C_C|$ is the number of components in C, $D(a_i, c_i)$ computes the largest diagonal length for all the patches in C_{P_m} contained in the i^{th} component, $A(f_i) \in C_{P_s}$ is the side patch that has the largest area in the same component, and $\gamma = 2$.

6.2.1.2 Topological Term

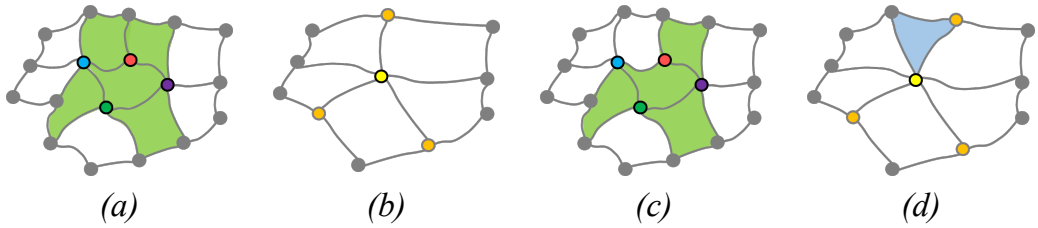


Figure 6.8: Removing green patches in (a) leads to an all-quadrilateral patch layout (b), while the valences of the newly created nodes are the same as the results computed by Equation 6.4. This valence updating even works for the case when removing green patches in (c) results in a triangular patch in (d).

To prioritize the impact of removing an operator (either a base-complex sheet or a chord) on the topological changes of the structure, I compare the valence changes of base-complex edges in the structures of hex-meshes before and after an operator removal. This means it is necessary to compute the valence of edges in the resultant structure before actually removing an operator. Since (base-complex) components could be removed during the collapsing of base-complex sheets or chords, and the valence for a base-complex edge is measured by the number of its neighboring components, for the simplified structure, I only need to consider the valences of two

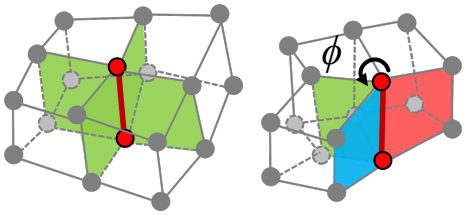
types of edges, i.e., those that are kept from the original structure but have some of their neighboring components being removed, and those that are newly created. During the simplification, for an edge e not being removed, the set of its neighboring components can be simply computed as $c'(e) = c(e) - r(e)$, where $c(e)$ is the set of its neighboring components in the structure before simplification, and $r(e)$ is the set of its neighboring components being removed. The new valence of e would be simply $l'(e) = l(e) - |r(e)|$, where $l'(e) = |c'(e)|$ and $l(e) = |c(e)|$. If an edge $\bar{e}$ is newly created from the collapsing of a group of edges $\{e_0, e_1, ..., e_n\}$, then the set of components surrounding this edge can be computed by Equation 6.4,

$$c'(\bar{e}) = (c(e_0) - r(e_0)) \cup (c(e_1) - r(e_1)) \cup ... \cup (c(e_n) - r(e_n)) \tag{6.4}$$

The first type of edges (i.e., not removed) can be treated using Equation 6.4 as well, if I consider they are newly created by collapsing to themselves. The computation of the valence for newly created edges in a hex-mesh can be easily adapted to measure the valence of a newly created node by collapsing a set of nodes in 2D quad-mesh, as illustrated in Figure 6.8. Note that, while all our operator removals introduced later maintain a valid all-cuboid structure, as long as the resulting mesh is still manifold, our formula for computing the valences of the newly created elements is not restricted to whether the resulting component (in 3D) is a cuboid or not, or the resulting patch is a quadrilateral (in 2D) or not. Figure 6.8(b) illustrates a 2D case that the valences for newly created nodes are consistent with the values computed using Equation 6.4 even it contains a triangular patch.

As illustrated in the left panel of the inset, in the interior of the volume, a base-complex edge with valence 4 (i.e., has four neighboring components) is preferred, as

it could be easier for each of the four neighboring components to form a $\pi/2$ volume angle surrounding this edge, facilitating the subsequent geometric optimization. The optimal number of neighboring components surrounding a boundary edge is highly related to the volume angle ϕ (see the right panel of the inset). In the metric terms described below, I estimate the optimal valence for boundary edges as $2\phi/\pi$, which could be a floating point value.



Based on the above descriptions, I consider the valence changes of the various scenarios for collapsing sheets and chords that involve interior and boundary edges as follows. Given a base-complex sheet S, since all edges in S_{E_m} will be removed after collapsing, its valence change is only determined by edges in S_{E_s}, which can be decomposed into N_E groups. Each group, denoted by $E_i = \{e_{i,0}, e_{i,1}, ..., e_{i,N_i}\}$ $(i = \{1, ..., N_E\})$ will be merged into a single edge $\bar{e}_i$ in the resulting structure. Note that if all the edges in a group are interior edges, $\bar{e}_i$ is an interior edge, whereas if one of the edges in a group is on the boundary, $\bar{e}_i$ is a boundary edge. I then measure the valence change for removing a sheet by Equation 6.5.

$$w_v(S) = \sum_{i=0}^{N_E} (\lambda_{i1} w_{iI} + \lambda_2^i (w_{iB})^\eta) \tag{6.5}$$

where w_{iI} measures the valence differences when all the edges in E_i are in the interior, w_{iB} measures the parametric-domain distortions when there are edges in E_i on the boundary surface. η is a weight that prefers the resulting boundary edges with better

valences. Specifically, if all $e_{i,j}$ in E_i are in the interior, then $\lambda_{i1} = 1$, $\lambda_{i2} = 0$ and

$$w_{iI} = \sum_{j=0}^{N_{E_i}} \sigma_{i,j}(|l(e_{i,j}) - 4| - |l(\bar{e}_i) - 4|), \tag{6.6}$$

$$\sigma_{i,j} = \begin{cases} 0, & \text{if } |l(e_{i,j}) - 4| \leq |l(\bar{e}_i) - 4| \\ 1, & \text{otherwise} \end{cases}$$

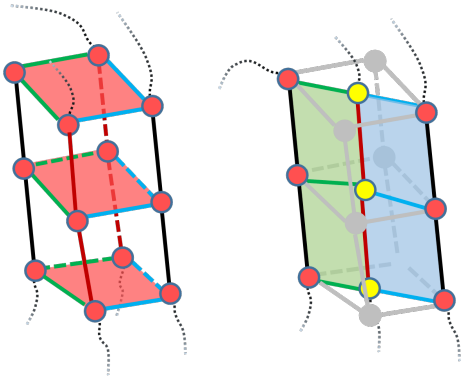
If there are N_m edges in E_i on the mesh boundary, then $\lambda_{i1} = 0$, $\lambda_{i2} = 1$, and

$$w_{iB} = \sum_{j=0}^{N_m} \sigma_{i,j}(|1 - \frac{2\phi_j}{\pi l(e_{i,j})}| - |1 - \frac{2\phi}{\pi l(\bar{e}_i)}|), \tag{6.7}$$

$$\sigma_{i,j} = \begin{cases} 0, & \text{if } |1 - \frac{2\phi_j}{\pi l(e_{i,j})}| \leq |1 - \frac{2\phi}{\pi l(\bar{e}_i)}| \\ 1, & \text{otherwise} \end{cases} \tag{6.8}$$

A boundary edge is preferred when the ratio of its valence divided by its volume angle ϕ and $\pi/2$ is closer to 1. Therefore, w_{iB} penalizes base-complex sheets whose collapsing results in new boundary edges with non-ideal valences, and this penalty is amplified by η.

Compared to the removal of a base-complex sheet where all newly generated edges are created by the collapsing of edges only in S_{E_s}, for a base-complex chord, collapsing edges both in sets C_{E_m} and C_{E_s} will result in new edges. Consider a chord shown in the left image of the inset.



Assume red patches are from C_{P_m}, red and black edges belong to C_{E_s}, and green and blue edges belong to C_{E_m}, respectively. By collapsing this chord, while edges from both sets C_{E_m} and C_{E_s} will be collapsed to form new edges (i.e., the red edges in the

right image of the inset), black edges in C_{E_s} will be kept yet with updated valences. The valences of edges with all these collapsing configurations can be accurately computed based on Equation 6.4. Therefore, following the similar procedure as described previously for a base-complex sheet, I can first decompose all the edges in C_{E_m} and C_{E_s} into distinct groups (each group will be collapsed to a new edge), and then construct the valence term for evaluating a chord based on whether there are edges in a group on the boundary surface or not.

6.2.2 Removal Operators

While directly collapsing a base-complex sheet onto its dual is ideal for some cases, as pointed out by Gao et al. [27], severe geometric distortion or the loss of important surface features may occur. Gao et al. [27] introduced a zig-zag strategy to reconnect misaligned singularities to maintain surface fidelity. However, it cannot be employed to handle our case where singularities are encouraged to be removed. Similar distortion may happen during the removal of a base-complex chord. Here, I present a new parameterization based removal strategy for both operators (sheet and chord) to address this issue.

Given a quadrilateral patch in a 2D base-complex, each patch can be mapped to a 2D rectangle. For the one-ring patches of a node, as shown in Figure 6.9(a), through the combination of rotation, translation, and an exponential mapping ($[\rho, \alpha] \rightarrow [\rho^k, k \cdot \alpha]$) of the spherical coordinates in the parameterization space, I can build a node interpolation domain as described in [89], where the exponent $k = (2\pi/n)/(\pi/2) = 4/n$.

In this way, all the vertices inside the parametric-domain are optimized. Similarly, I adapt this strategy to handle the case where the neighboring patches of a boundary node form only a fan-shape (i.e., half circle). As illustrated in Figure 6.9(b), by placing the node with parametric coordinate $(0, 0)$ on the boundary of the parametric-domain and fixing its position, I can also build a spherical parameterization which optimizes the vertices inside the boundary domain. Here, the exponent $k = (\phi/n)/(\pi/2) = 2\phi/(n\pi)$, where $\phi < 2\pi$ and is estimated by firstly projecting the fan-shape patches onto a 2D plane, and then computing the angle spanned by the domain around the node.

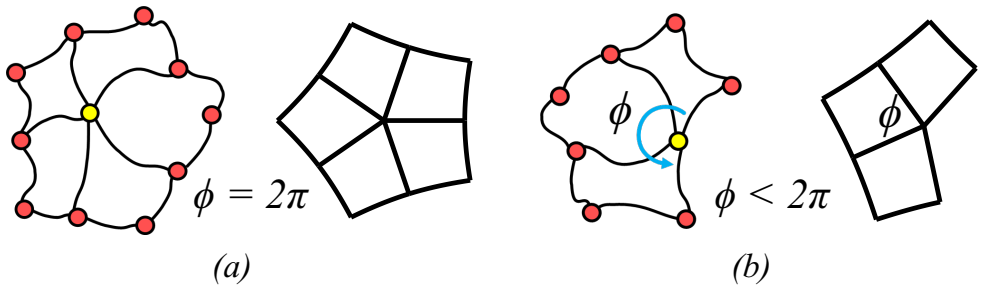


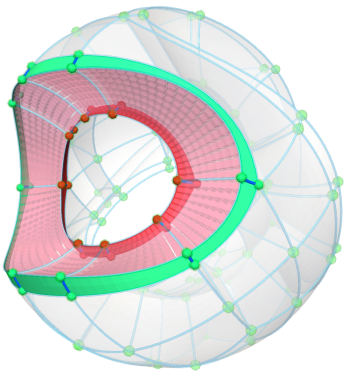
Figure 6.9: (a) a one-ring patch domain of a node, (b) a spherical parametric-domain with $\phi = 2\pi$, (c) a fan-shape patch domain of a node, and (d) a spherical parametric-domain with $\phi = 1.3\pi$.

I now describe the removal of sheets and chords using the above parameterization strategy.

Base-complex sheet removal: As explained earlier, removing a base-complex sheet that has some patches located on the boundary surface may cause large distortion and feature loss. I refer to these patches as the *boundary patches*. To address this, I classify these boundary patches into two types: patches in the S_{P_s}, and patches

in S_{P_m}. Since the removal of a sheet is to erase all the patches in S_{P_m} and merge two patches P_i and P_j in S_{P_s} (i.e., with one on the left surface and the other on the right surface of the sheet) into one P_{ij}, P_i and P_j cannot be both on the boundary surface. Without loss of generality, if P_i is on the boundary, I set P_{ij} as P_i. If both are in the interior, I set P_{ij} as the middle patch of P_i and P_j. With this consideration, collapsing patches in S_{P_s} will not cause surface distortion. In contrast, removing patches in S_{P_m} may impact surface fidelity greatly. In the following I concentrate on reducing the surface distortion when removing boundary patches in S_{P_m}. In this case, I only need to deal with open sheets, because all the patches in S_{P_m} of closed sheets are in the interior of the volume.

For an open sheet, if it is regular and its boundary surface has genus larger than zero, then its boundary patches in S_{P_m} may form more than one surface ring. The inset image shows an open and regular sheet with genus-2 that has two surface rings (red and green).



Consider a surface ring (partially shown as the red corridor in Figure 6.10(a)), it is embedded in a patch network. In this scenario, each surface ring can be processed independently with the following strategy. The network is the deduction of boundary patches in S_{P_s} from the 2D surface base-complex. Consequently, from Figure 6.10(a) I see that, some red nodes in Figure 6.10(a) of the ring are on the boundary. That means they have neighboring patches in S_{P_s} on the boundary. Conversely, those red nodes in the interior of the network mean all of their neighboring patches in S_{P_s} are

in the interior volume of the mesh.

I sequentially collapse each edge in S_{E_m} (red in Figure 6.10 (a)) into a node (yellow in Figure 6.10 (b)). For each of its two nodes (red in Figure 6.10 (a, c-f)), based on its neighboring patches, I construct a node domain. The constructed domain could be either disk-like (Figure 6.9(a)) or fan-like (i.e., a partial disk, Figure 6.9(b)). Given the valence relationship (i.e., Equation 6.4) between the newly created node and the two end nodes of an edge before collapsing, respectively, I can obtain the parametric-domain relationship $\bar{P} = (P_1 \cup P_2) - (P_1 \cap P_2)$, where $\bar{P}$ is the parametric-domain of the node after collapsing (Figure 6.10(b)) that can be constructed topologically. For example, for the middle edge of the three red edges in the top image of Figure 6.10 (f), its two node parametric-domains are $P_1 = \{A, B, C, D\}$ and $P_2 = \{B, C, E, F, G\}$, respectively, where both of them are disks. After removing the current sheet, the parametric-domain of the node corresponding to this edge is $P_{12} = \{A, D, E, F, G\} = (P_1 \cup P_2) - (P_1 \cap P_2)$ (bottom image of Figure 6.10 (f)). In fact, when collapsing the parametric-domains of the two end nodes of an edge: 1) if both P_1 and P_2 are disks, then P_{12} will be a disk; if either P_1 or P_2 is a fan, then P_{12} will be a fan.

To avoid geometric distortion, I try to compute the node domain after removal within the edge domain before collapsing. Therefore, I need to build a one-to-one mapping between the boundary of the common domain of v_1 and v_2 and the boundary of the node domain of v_{12} (e.g., Figure 6.10 (c-f)). To achieve that, for an edge, I can temporarily split all the other edges in S_{E_m} with their corresponding nodes that they will collapse to (i.e., yellow dots with dashed boundaries in Figure 6.10(c-f)) to simulate the collapsed parametric-domain. For a temporarily split edge, depending

on whether there is a boundary node at one of its ends or not, its temporarily computed collapsed node could be the boundary node (Figure 6.10 (c-e))) or in the middle of the edge (Figure 6.10 (d-f)). In this case, all the elements within the parametric-domain of this edge will be re-meshed and optimized simultaneously.

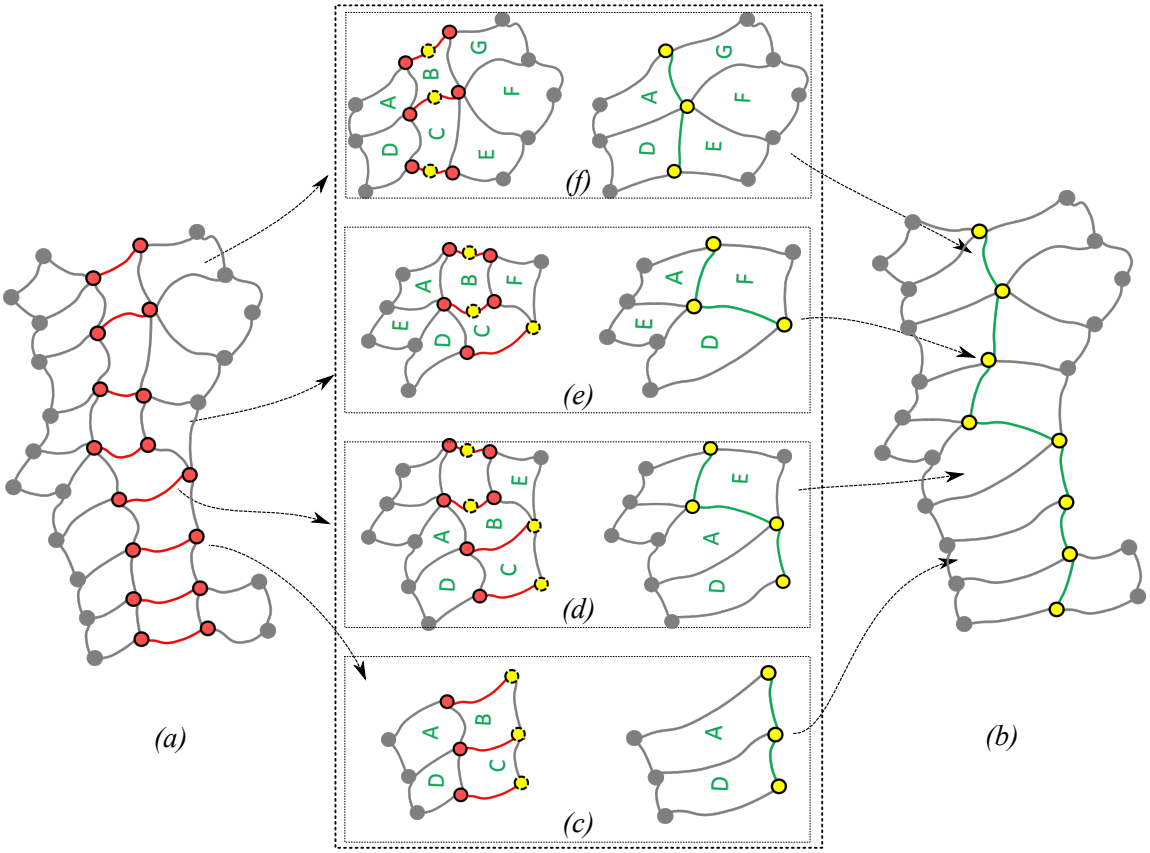


Figure 6.10: Parametrization based removal of a surface ring, represented by a sequence of red nodes and edges in (a), of a sheet. (c-f) show a number of node domains before (top) and after (bottom) collapsing. The yellow dots with dashed boundary are the tentative nodes that the corresponding red nodes will collapse to. The collapsing result is shown in (b).

Handling tangled and tangent configurations: For a sheet tangled at open chords, I remove the top-ranked chord from them. Recall that chords can be formed

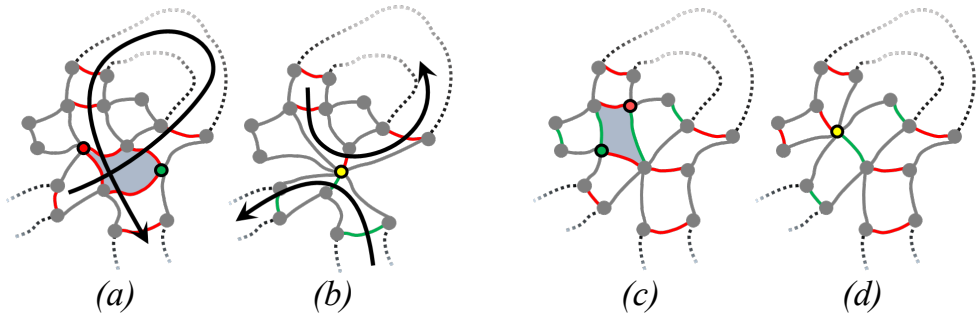


Figure 6.11: The tangled sheet in (a) can be removed by collapsing the highlighted patch in (a) to obtain two sheets in (b); by collapsing the highlighted patch in (c), the tangent configuration can be removed in (d).

by the components of the sheet where tangling occurs. By the removal of such a chord, the tangled sheet may be separated into multiple sheets, which requires the re-ranking of all sheets to identify a possibly better sheet. Figure 6.11(a-b) illustrate a 2D example. For an open sheet tangent at boundary vertices or edges, I first collect all the surface patches neighboring to these vertices and edges. Recall that each of these patch corresponds to an open chord. I can remove the top-ranked chord from these open chords associated with these boundary patches. Similarly, I re-rank all the sheets to identify a better one. Figure 6.11(c-d) provide a 2D example.

Base-complex chord removal: Similarly, the patches of a chord that are on the boundary surface can be classified into two types, i.e., patches in set C_{P_s} and patches from set C_{P_m} of an open chord (recall that all the patches in set C_{P_m} of a closed chord are in the interior volume). Removing a chord is achieved by collapsing the four side surfaces into two surfaces. Since the two corresponding patches that will be collapsed into one should not be both on the mesh boundary, I can always

collapse the patches in the interior to the patches on the boundary. I now focus on the removal of boundary patches that are in C_{P_m}.

For an open chord with a regular configuration, its boundary patches in C_{P_m} are at the two end positions of the chord. Their removal can be performed by the procedure as detailed below. Given an open chord, assume the collapsing is in the diagonal direction pointing from b to d, as shown in the 2D case in Figure 6.12(a). There could be cases that 1) both b and d have node parametric-domain as a disk-like shape (Figure 6.12, top), and 2) either b or d have their corresponding parametric-domain as a fan-shape (Figure 6.12, bottom). For both cases, the parametric-domain of the collapsed node (yellow) will be $P_{bd} = (P_b \cup P_d) - (P_b \cap P_d)$. Since the parametric boundaries of $P_b \cup P_d$ and P_{bd} are the same, I can perform the node parameterization of P_{bd} directly on the combined domain of P_b and P_d to re-mesh and optimize the positions of the elements inside the domain. The top and bottom rows of Figure 6.12 illustrate the handling of the two cases, respectively. For an open chord with tangent or tangled configurations, I process it similarly to handle the tangled or tangent sheet described previously.

6.2.3 Topology Preservation

In our algorithm, the genus of a hex-mesh during optimization is maintained by checking both the surface (#V+#F-#E) and volume (#V+#F-#E-#H) Euler characteristics for the meshes before and after a sheet or chord removal. Both of the Euler characteristics should remain unchanged during simplification. This excludes

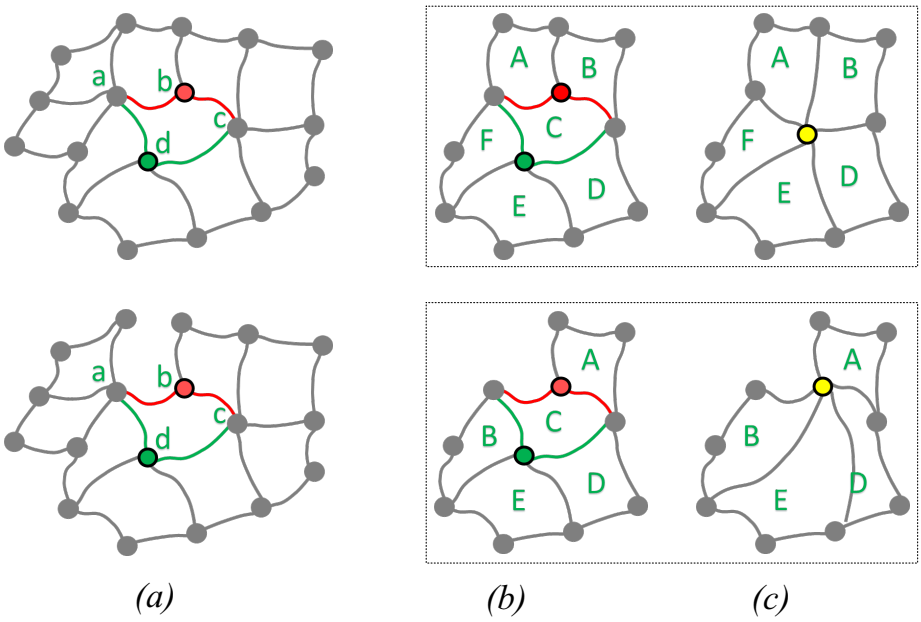


Figure 6.12: Parametrization based collapsing of chords. (a) shows two chords that are to be collapsed (e.g., b to d or vice versa), whose node parametric-domains around b and d are a disk (top) and a fan (bottom), respectively. (b-c) show that the collapsing of the disk-like domain (top) and the fan-like domain (bottom), respectively.

the base-complex sheets and chords whose two side patches are both on boundary from the candidate list, as their removal will alter these Euler characteristics. For a hex-mesh that is three-manifold, its boundary is a two-manifold quad-mesh, which prohibits cases such as two components on the surface share either a node or an edge, as shown in Figure 6.12(a-b).

I also set a lower bound (i.e., 3) and an upper bound (i.e., one plus the maximum valence of the edges in the input hex-mesh) for the singularity valences, and a low bound (i.e., 3) for the surface vertex valence, as hard constraints of the simplified meshes. While it is easy to understand that high valence singularities can greatly narrow the geometric optimization space, an interior singular edge with low valence

(e.g., 2) can fail all the untangling algorithms, as well as posing challenges to the base-complex extraction. As shown in Figure 6.13 (c-d), the maximum Jacobian values of the involved hex elements are zero, which is not acceptable for subsequent applications. Figure 6.13 (e) shows an example that the eight red nodes represent two adjacent components.

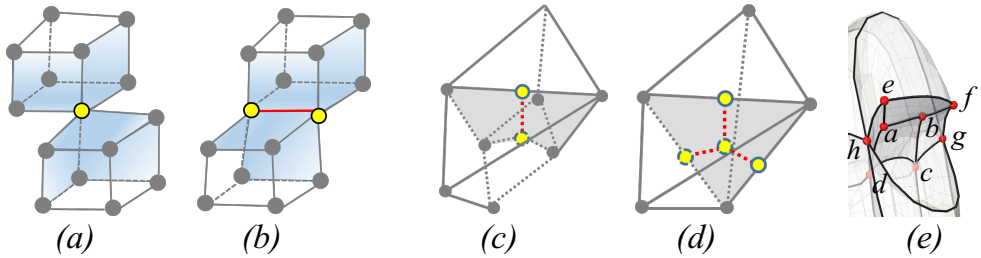


Figure 6.13: Cases such as two components share a point (a), and an edge (b) are not allowed, while when base-complex edges with valence two presented, topological configurations such as 10 nodes (c), 7 nodes (d), and 6 nodes (e) share two hex elements may occur.

6.2.4 Feature Preservation

In this work, since I treat the interior volume of a hex-mesh as isotropic, I try to preserve features on surface boundary. As commonly done in previous work [89, 27], during the parameterization, feature edges on surface can be considered as hard constrains and snapped to appropriate parametrization curves in the node parameterization domain.

6.3 Results

I have applied the proposed approach to hex-meshes generated by a variety of state-of-the-art algorithms including the Octree-based [61], Polycube-based [55, 38], and frame-field based [52] approaches, as well as meshes after alignment [27]. The selected meshes covered a spectrum of man-made and natural shapes, with various topological and geometric complexities. Table 6.1 provides the statistics of the tested hex-meshes before and after simplification, including: the name of each model used in their original work, number of hex elements ($|\mathcal{H}|$), the number of operator removals during the simplification process ($|\int|$), the index of the binary searched optimized mesh in the simplification sequence ($|\mathcal{I}|$), the average and minimal Scaled Jacobians (S. J.), and the number of components ($|\mathcal{B}_C|$) in the base-complex. The simplification rate for each model was provided in the R column in Table 6.1, which was computed as the ratio between $|\mathcal{B}_C| - |\mathcal{B}'_C|$ and $|\mathcal{B}_C|$. As shown in the $|\mathcal{B}_C|$ column in Table 6.1, I can see that our algorithm significantly simplifies the base-complex of each tested hex-mesh, while satisfying the local element quality requirements (the S. J. column in Table 6.1). Note that, I assume that the positive minimum-scaled Jacobian is acceptable in our experiments. If a higher quality hex-mesh is sought, our binary search will return a mesh with more complex structure than those shown in the results. Figure 6.14 provides visual results before and after simplification on some of the hex-meshes listed in Table 6.1. The hexahedral elements that belong to the same components of the base-complex are shown in the same colors. In our experiments, depending on the hex element number and topological complexity, the computational time varied from seconds (e.g., Ellipsoid model) to 30 minutes (e.g., Anc101 model).

All the timing information was recorded on a PC with an Intel Xeon (E5-1620) processor and 16 GB memory.

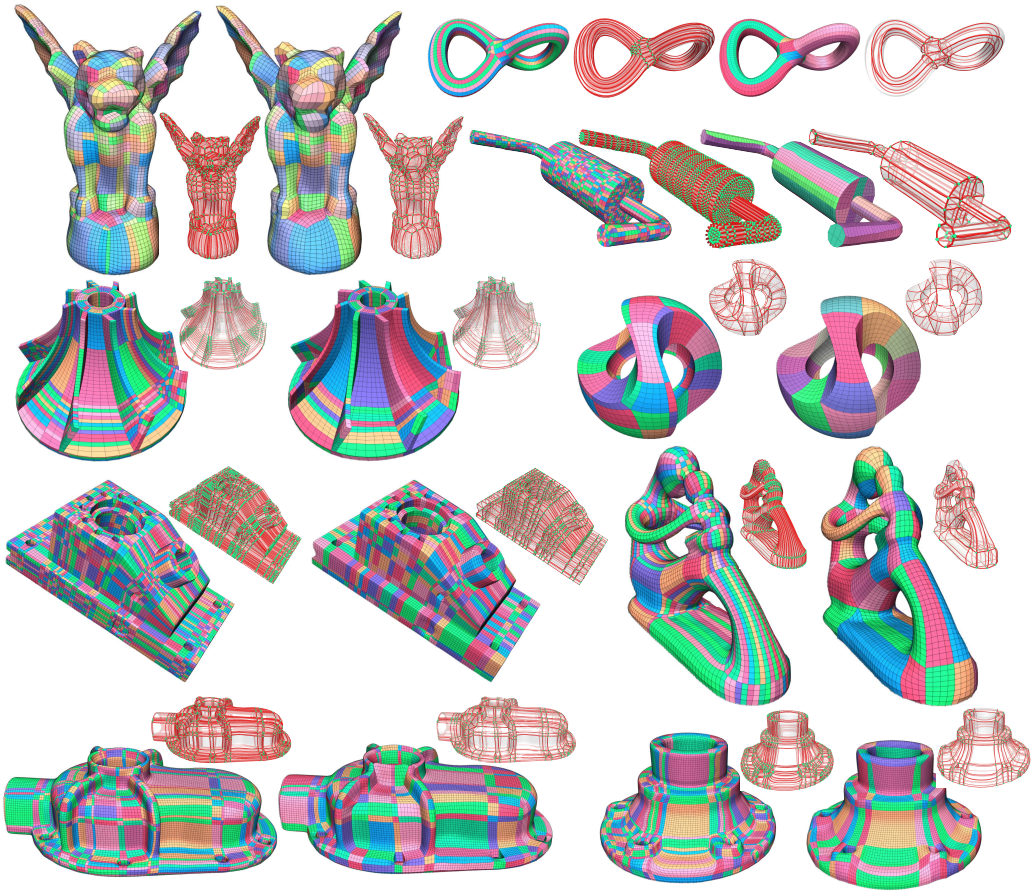


Figure 6.14: The hex-meshes of pipe [61], Sculpture, Double-torus, Impeller, Fertility [52], the Gargoyle [27], Anc101 [32], Cognit [38], and Carter [55] before (the left two of each model) and after simplification (the right two of each model).

6.3.1 Comparisons with the Alignment Approach

Figure 6.15 compares our approach with the alignment method introduced by Gao et al. [27]. Because of the constraint of singularity preservation during the alignment,

Models	$\|\mathcal{H}\|$	$\|f\|$	$\|\mathcal{I}\|$	S. J.	$\|\mathcal{B}_C\|$	Ratio
Anc101* (Fig. 6.14)	74k 70k	51	19	.952/.196 **.965/.076**	12336 **2172**	82.4%
Asm001*	18k 20k	5	2	.95/.131 **.937/.121**	122 **29**	72.2%
Bumpy-torus*	35k 38k	58	31	.891/.27 **.909/.396**	2518 **635**	74.8%
Bunny*	82k 81k	24	15	.930/.138 **.934/.292**	1324 **232**	82.5%
Casting*	21k 21k	41	21	.845/.195 **.845/.039**	2805 **1150**	58.6%
Fertility* (Fig. 6.14)	20k 22k	30	23	.911/.196 **.873/.261**	598 **240**	59.9%
Bone†	3k 3k	9	8	.93/.620 **.844/.169**	87 **7**	92.0%
Bunny†	134k 138k	10	5	.94/.293 **.929/.302**	259 **65**	74.9%
Double-torus† (Fig. 6.14)	4k 9k	8	5	.891/.717 **.918/.564**	185 **24**	87.0%
Ellipsoid† (Fig. 6.1, 6.2, 6.3, 6.4)	2k 3k	4	4	.95/.752 **.817/.07**	34 **1**	97.1%
Fertility† (Fig. 6.5, 6.6, 6.14)	14k 20k	32	29	.911/.351 **.877/.062**	1352 **121**	91.1%
Hanger†	5k 17k	8	4	.964/.599 **.920/.116**	60 **40**	33.3%
Impeller† (Fig. 6.14)	11k 15k	67	31	.924/.185 **.906/.021**	944 **361**	61.8%
Joint† (Fig. 6.15)	18k 20k	9	6	.984/.729 **.958/.583**	83 **17**	79.5%
Rod†	5k 4k	8	2	.947/.658 **.928/.253**	66 **51**	22.7%
Sculpture-A†	24k 7k	11	7	.961/.689 **.846/.057**	51 **12**	76.5%
Sculpture-B† (Fig. 6.14)	6k 9k	13	6	.892/.055 **.853/.047**	51 **30**	41.2%

Models	$\|\mathcal{H}\|$	$\|f\|$	$\|\mathcal{I}\|$	S. J.	$\|\mathcal{B}_C\|$	Ratio
Carter‡ (Fig. 6.14)	65k 72k	48	26	.895/.250 **.807/.192**	2500 **566**	77.4%
Fertility‡	54k 53k	29	18	.872/.259 **.872/.256**	693 **270**	61.0%
Anc101♯	63k 16k	58	28	.964/.150 **.943/.085**	5009 **1630**	67.5%
Angel♯	14k 11k	30	19	.923/.470 **.904/.338**	1284 **218**	83.0%
Bumpy-torus♯	39k 40k	37	33	.929/.335 **.913/.320**	2254 **502**	77.7%
Bunny♯	38k 37k	14	6	.926/.382 **.924/.355**	273 **132**	51.6%
Bustle♯	12k 11k	19	18	.920/.442 **.890/.464**	348 **7**	98.0%
Cognit♯ (Fig. 6.14)	78k 74k	55	29	.829/.270 **.812/.071**	5194 **1709**	65.2%
Kitty♯	7k 21k	9	3	.91/.424 **.905/.312**	121 **77**	36.4%
Rod♯	11k 3k	7	2	.929/.418 **.89/.056**	122 **29**	76.2%
Dancing-Children⋆	38k 8k	43	11	.876/.184 **.793/.007**	1458 **1014**	30.5%
Gagoyle⋆ (Fig. 6.14)	28k 29k	50	18	.911/.214 **.902/.143**	1920 **1265**	34.1%
3-torus⊺ (Fig. 6.15)	5k 5k	67	61	.87/.33 **.890/.499**	3306 **82**	97.5%
Pipe⊺ (Fig. 6.14)	10k 11k	55	49	.890/.094 **.928/.225**	6121 **69**	98.9%
Rabbit§ (Fig. 6.16)	33k 38k	138	137	.896/-.917 **.882/.232**	32898 **7**	99.98%
Rabit-twist§ (Fig. 6.16)	34k 34k	143	142	.905/-.980 **.867/.157**	33967 **7**	99.98%
Sphere♮ (Fig. 6.17)	320 5k	38	37	.500/.263 **.913/.452**	320 **7**	97.8%

Table 6.1: Topological and geometrical comparisons of hex-meshes before and after simplification. For each model, the original hex-mesh and optimized hex-mesh by our method are shown in the upper row and bottom row, respectively. Input hex-meshes are from [32]*, [52]†, [55]‡, [38]♯, [27]⋆, [61]⊺, Voxelization § and tet-mesh subdivision ♮, respectively.

much fewer components were removed when most of the base-complex edges in the base-complex were singularities (Figure 6.15 (c-d)). This constraint was mitigated by our algorithm, with which a much simpler structure can be obtained (Figure 6.15 (e-f)). In fact, this is expected from the operators employed during the simplification, where the set of candidates for alignment is only a sub-set of our solution space.

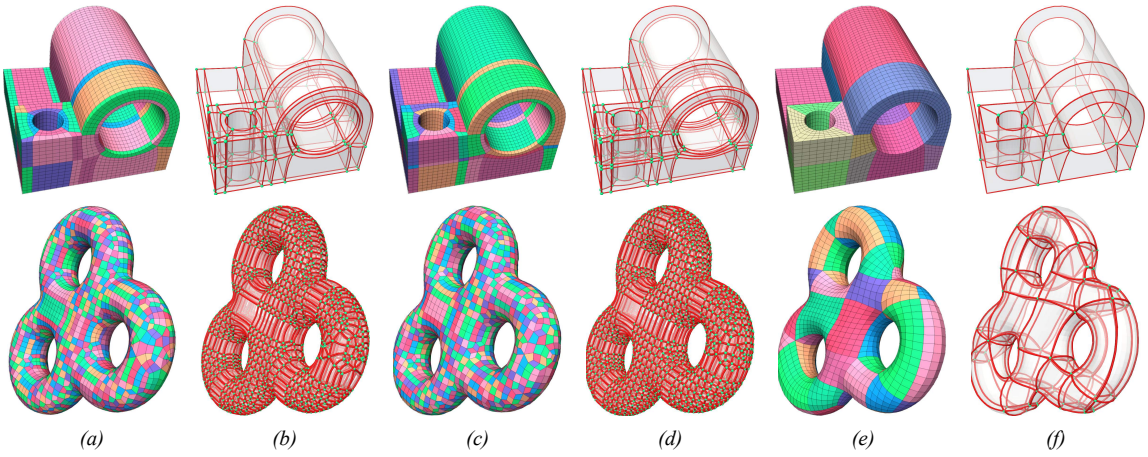


Figure 6.15: Compared to the results (c-d) optimized by [27], The proposed approach simplifies input meshes (a-b) with either sharp feature or complicated structures greatly while maintaining a high geometric quality (e-f).

6.3.2 Hex-meshes from Voxelization

A voxelized mesh can be considered as a complicated volumetric polycubes, with each vorxel as a polycube. A hex-mesh can be generated by first voxelizing the volume [62, 70] and then applying a padding process afterward. The hex-mesh obtained this way is all-hex yet with a complicated structure, which can hardly be employed by subsequent applications. Figure 6.16 (a,c) show two hex-meshes of a rabbit model with two different postures using voxelization approaches. Our technique successfully simplifies the structures of both meshes (Figure 6.16 (b,d)) while improving their geometric quality (see the corresponding entries in Table 6.1).

6.3.3 Hex-meshes from Tet-meshes

By subdividing each tetrahedral into four hexahedral elements, I can easily obtain an all-hex-mesh from a tet-mesh with quite a complicated structure, as shown in Figure 6.17(a). The simplified structure of this hex-mesh using our algorithm under default topology constraint settings is still too complex (Figure 6.17(b)). By relaxing the maximum valence constraint (e.g. five times of the maximum valence in the initial mesh), our algorithm can successfully simplify the mesh into its simplest form (i.e., a cube). Figure 6.17(c-e) demonstrates the last three steps of the simplification process for this mesh. Even though Figure 6.17 shows that the excessively complex base-complex of the hex-mesh converted from a tet-mesh of a sphere can be greatly simplified, more investigations will be needed for dealing with other more complex models.

6.4 Conclusion

In this chapter, I introduced a parameterization based approach to simplify the global structure of a hex-mesh. Our simplification was based on the collapsing of the base-complex sheets and chords extracted from the base-complex of the mesh. By optimizing the ordering of the removal of sheets and chords, our technique effectively reduced the complexity of the base-complex while maintaining the geometric fidelity of the input mesh. Given the user-input desired complexity, I employed a simple yet effective binary search strategy to obtain the optimal simplified structure of a hex-mesh without negative Jacobian from the obtained all-hex structure hierarchy. I

have applied our framework to numerous hex-meshes to demonstrate its effectiveness.

Limitations: While the proposed approach can simplify the structure of an input hex-mesh into its simplest form(s), to satisfy the constraint of local element quality (i.e., positive minimum-scaled Jacobian) for subsequent applications, the output hex-mesh of our pipeline usually has a more complex structure than its simplest representation. To address this, more effective optimization (especially, untangling) techniques for hex-meshes with simpler structure need to be investigated. In addition, despite successfully simplifying the hex-mesh of a sphere converted from its tet-mesh, the base-complex may become too complex for our algorithm to process (i.e., early termination) due to the more complicated configurations of sheets and chords, leading to topological degenerate configurations that are currently not well-understood. I plan to address these limitations in the future.

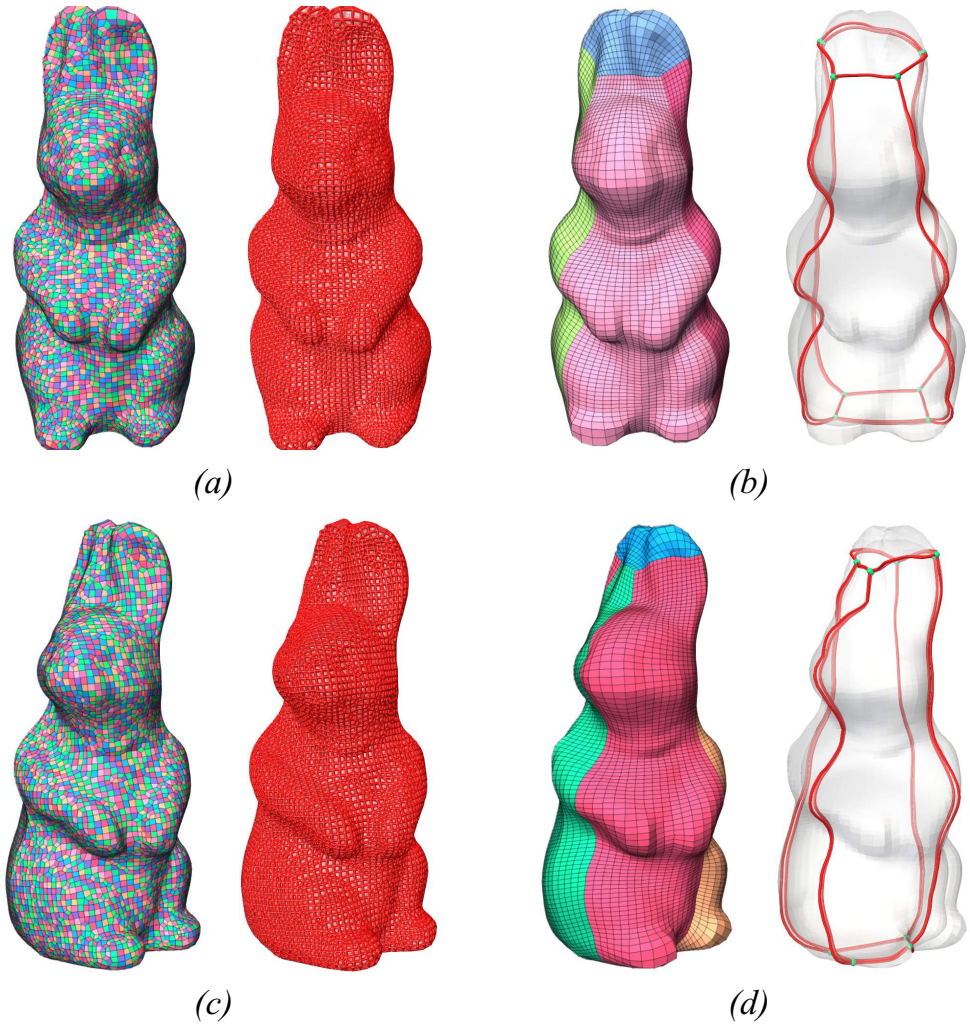


Figure 6.16: Our approach can greatly simplify hex-meshes (a) and (c) directly converted from voxelized meshes, while the surface patches of the resultant structures well capture the geometric characteristics of the model.

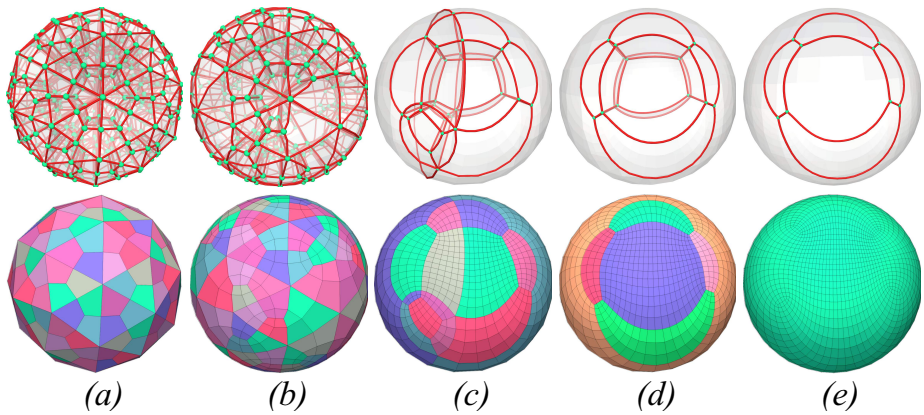


Figure 6.17: For a hex-mesh (a) converted from a tet-mesh, using the default maximum-edge valence constraint, our algorithm produces a hex-mesh with a structure that is still too complicated (b). By allowing larger maximum-edge valences, the input hex-mesh can be simplified to its simplest structure representation (e), where (c) and (d) are the simplified results of the third and second to the last steps, respectively.

Chapter 7

Evaluating Quality Metrics for Hex-meshes

As mentioned in Chapter 1.2, there does not exist a guideline for the selection of proper metrics for the effective measurement of hex-mesh quality in practice. To solve that, I conducted a first comprehensive study on the correlation among various hex-mesh quality metrics. Even though all existing metrics are defined and computed with analytic formulas [86], their relations cannot be simply formulated or predicted. Under this circumstance, studying the co-variant behaviors among different metrics is a natural quest, which is the first contribution of this work. Specifically, I compute the Pearson Product-Moment Correlation Coefficient (PPMCC) [85] for all pairs of metrics. Based on their pairwise covariance coefficients, I perform a bottom-up hierarchical-clustering of these metrics, in which two closest (or most similar) clusters are agglomerated each time. This gives rise to a binary tree, where the distance of

any pair of metrics in the tree measures their similarity. I demonstrated that this analysis framework can be effectively applied to reduce the number of quality metrics and identify the most reliable metric given a specific application.

Our second contribution lies in a strategy of generating a large set of hex-mesh samples with well-distributed metric values, which is required for an accurate correlation analysis. As it is known that the number of hexahedral elements has great influence on the simulation quality [64], it is necessary to maintain a constant number of elements during the generation of hex-mesh samples (for a 3D object) to minimize the effect of the element number on simulation quality. In the meantime, the metric values for these mesh samples should have a statistically sound distribution required for the co-variance study, which is difficult to achieve with existing hex-mesh generation tools. To address this challenge, I introduce a two-stage noise insertion framework to facilitate the generation of the test dataset. This is also the first time that tens of thousands of meshes are used for the evaluation of the effectiveness of quality metrics. To increase the generality of our study, the 3D objects I consider include both natural and man-made objects with various geometric and topological characteristics.

Third, I applied the proposed analysis techniques to three different applications – the linear-elasticity problem, Poisson's equation-solving and Stoke equation-solving, respectively, on which I conducted a first study to understand the effectiveness of all existing hex-mesh quality metrics. I observed that average metrics greatly affect the accuracy of those applications, while minimum and maximum metrics have more influences on the stability of these applications.

To the best of our knowledge, this work is the first step to quantitatively understand the correlation characteristics of a large number of existing hex-mesh quality metrics and their effectiveness to downstream applications. The encouraging results from our study can be used as practical guidelines for the development of effective hex-mesh generation and optimization techniques as well as the cornerstone for future research studies along this line.

7.1 Methodology

To understand the relationships among dozens of hex-mesh quality metrics, I compute the linear-correlation coefficients between metrics (Section 7.1.1), and use a hierarchical-clustering algorithm to classify similar metrics based on their correlation coefficients (Section 7.1.2). In Section 7.1.3, I present a two-level strategy to generate a large number of sampled meshes from benchmark hex-meshes with statistically sound distribution in the metric space.

7.1.1 Linear Correlation Analysis

As described earlier, I focused on how different metrics vary or co-vary. In addition, without a-priori on the exact relation among pair of metrics, linear-correlation study is the natural choice to understand their relations as the first step. In this work, I use the Pearson Product-Moment Correlation Coefficient (PPMCC) [85], $r_{x,y}$, to

measure the linear-correlation between any two metrics X and Y.

$$r_{x,y} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{7.1}$$

Here, $r_{x,y}$ is a real number in the range of [-1, 1]. The closer it is to 0, the weaker the correlation between X and Y. X and Y are positively correlated if $r_{x,y} > 0$; otherwise, X and Y are negatively correlated.

Consider a data matrix, $D[N, M]$ with each row corresponding to a quality metric (i.e., N metrics in total), and each column being the respective metric values of a hex-mesh (i.e., M hex-mesh samples in total). A *correlation matrix*, denoted by $C[N, N]$ (or simply C), is defined as:

$$C = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,N} \\ r_{2,1} & r_{2,1} & \cdots & r_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ r_{N,1} & r_{N,2} & \cdots & r_{N,N} \end{bmatrix} \tag{7.2}$$

Each entry $r_{i,j}$ of C is the correlation coefficient of the i^{th} and j^{th} metrics given the M hex-mesh samples using Eq.(7.1). C is a symmetric matrix, and its columns or rows will be used to measure the similarity among different metrics (Section 7.1.2).

7.1.2 Metric Reduction

After calculating the correlations among pairs of metrics, I further classified them into several groups based on their similarities defined according to the obtained

correlation matrix C. Our goal is to reduce the large number of metrics (as as seen in Table 3.1) into a small representative sub-set.

Many clustering algorithms [95] can be potentially used. Since the number of clusters for the metrics is unknown, I opt for the popular Ward's hierarchical agglomerative-clustering method [65]. This hierarchical-clustering is performed using a set of dissimilarities for n objects. In our case, the n objects are the metrics, each of which is characterized by a vector that is a row or a column in the matrix C. The process starts with assigning each metric as a cluster, and then proceeds iteratively, where at each step the two most similar clusters are merged into one. This process continues until only one single cluster remains. The similarities between any two clusters were measured by the Lance-Williams dissimilarity update formula that encodes the Ward's minimum variance criterion. Assume two clusters (containing one or more metrics), denoted by $\vec{i}$ and $\vec{j}$, are agglomerated into cluster $\vec{i} \cup \vec{j}$, the Lance-Williams dissimilarity update formula for computing the new dissimilarities between the new cluster and all other clusters is:

$$d(\vec{i} \cup \vec{j}, \vec{k}) = \alpha_{\vec{i}} d(\vec{i}, \vec{k}) + \alpha_{\vec{j}} d(\vec{j}, \vec{k}) + \beta d(\vec{i}, \vec{j}) + d_\gamma \tag{7.3}$$

where $d_\gamma = \gamma \left| d(\vec{i}, \vec{k}) - d(\vec{j}, \vec{k}) \right|$, and $d(\vec{i}, \vec{j})$ defines the function of Ward's minimum variance as:

$$d(\vec{i}, \vec{j}) = \frac{\left|\vec{i}\right| \left|\vec{j}\right|}{\left|\vec{i}\right| + \left|\vec{j}\right|} \left\| \vec{i} - \vec{j} \right\|^2. \tag{7.4}$$

$\left|\vec{i}\right|$ is the cardinality of cluster $\vec{i}$, and $\left|\vec{i}\right| = 1$ when $\vec{i}$ contains only one metric. For Ward's minimum variance metric, the corresponding agglomerative criteria $\alpha_{\vec{i}}$, $\alpha_{\vec{j}}$, β, and γ are $\frac{|\vec{i}|+|\vec{k}|}{|\vec{i}|+|\vec{j}|+|\vec{k}|}$, $\frac{|\vec{j}|+|\vec{k}|}{|\vec{i}|+|\vec{j}|+|\vec{k}|}$, $\frac{|\vec{k}|}{|\vec{i}|+|\vec{j}|+|\vec{k}|}$, and 0, respectively. For more details, please refer to [65].

Although one may simply use the correlation coefficient of two metrics to measure their similarity, their respective correlation coefficients to other metrics will affect their distance measurement in the metric space (a higher-dimensional space). To define a more accurate similarity metric, for each quality metric, I consider its corresponding row or column in the correlation matrix (Figure 7.4) with *absolute* entry values as its feature vector (i.e., $\vec{i}$ and $\vec{j}$ above). If two metrics have similar feature vectors measured by Eq.(7.4), then these two metrics should also have similar evaluation capacity to the hex-meshes. Based on the feature vectors of the individual metrics, I perform the above hierarchical-clustering to organize the metrics into a binary tree (see Figure 7.5 (a) for an example).

7.1.3 Data Preparation

To employ the aforementioned correlation analysis and clustering techniques, a large hex-mesh dataset that has well-distributed metric values is needed; otherwise, the generality of the analysis results will be questionable. However, it is non-trivial to generate hex-meshes that are well-distributed in the metric space while having the same or similar numbers of elements using existing techniques. This is because different metrics have unknown dependencies on each other, altering one metric value

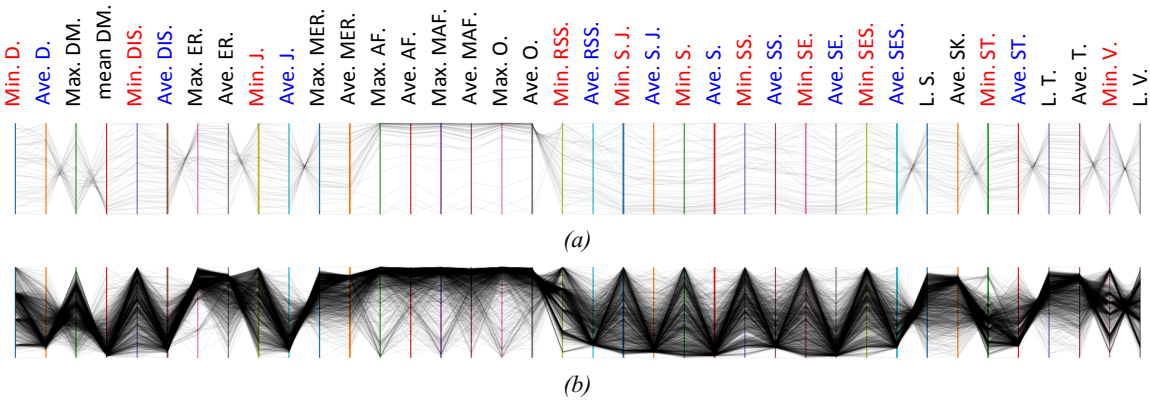


Figure 7.1: Parallel coordinate visualizations of all the metric values of (a) $\mathcal{H}_{min}$, 59 Cube hex-meshes by only controlling the minimum metrics with lower bounds and ↑ trends (red labels), and (b) $\mathcal{H}_{ave}$, 1677 Cube hex-meshes by additionally controlling the average metrics with lower bounds and ↑ trends (blue labels). I do not explicitly control metrics with labels in black.

may change the others in an unknown fashion.

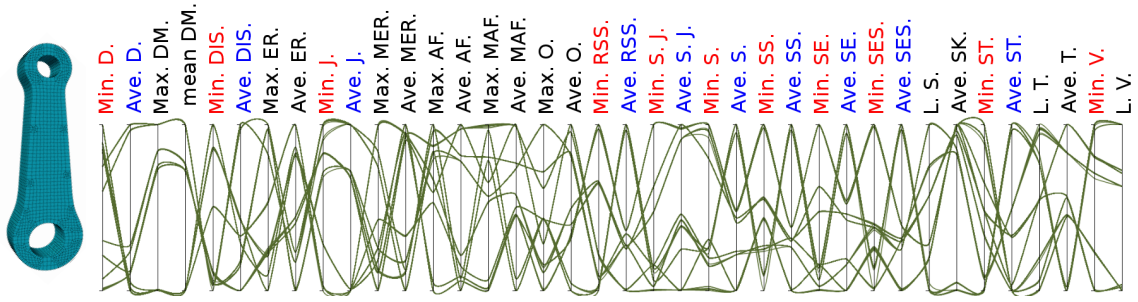


Figure 7.2: Parallel coordinate visualization of all the metric values of a set of hex-meshes of the crank model generated using MeshGem.

As a concrete example, I use MeshGem [61]–an octree-based tool to generate a set of hex-meshes from a selected object. Each of these meshes is generated with a slightly different element orientation from the others, while keeping the other parameters fixed. This allows us to produce a set of meshes with similar numbers of elements. The reason of adopting an octree-based method for this experiment is

because it is more robust and easier to use than other available techniques, such as the Polycubes-based [56, 38] and frame-field based [52, 43] approaches. Figure 7.2 shows a plot of the distribution of the individual metric values for the hex-meshes (i.e., 657 in total) of the crank model using the parallel coordinate technique [42]. The distribution of the metric values of these meshes is highly uneven. They can be clearly classified into no more than 30 clusters with distinct metric values. This means only at most 30 meshes are usable for the correlation analysis, making it inefficient to generate the test dataset with the octree-based method.

7.1.3.1 A Perturbation-based Data Generation Strategy

To address the above data generation challenge, I introduce a two-stage noise insertion strategy.

Our method was based on the following observation. That is, all the metrics in Table 3.1 can be divided into two groups based on their characteristics, i.e., a group with those metrics that have minimum/maximum values, and the other group consisting of the remaining metrics. According to their definitions, I know that minimum/maximum metrics can be greatly influenced by modifying a single vertex/hexahedron–much more effective than re-generating the entire mesh, while average metrics can only be affected by modifying the majority of the elements. Based on the above insights, given an input hex-mesh H_0, I generated its hex-mesh dataset using the following two steps (Algorithm 4).

First, I generate a series of hex-meshes $\mathcal{H}_{min}$ by explicitly controlling their distribution over the valid spaces of minimum metrics (i.e., those minimum metrics in Table 3.1 with lower bounds and $\uparrow$ trends). There are $N_{min} = 11$ such kind of minimum metrics. Specifically, for each of those metrics, M_i, with the value $\min_i$ of hex-mesh H_0, I evenly divide its valid range $\left[l_b^{M_i}, \min_i\right]$ into α sub-ranges, where $l_b^{M_i}$ is provided in Range* column of Table 3.1 that need not be the mathematical lower bound of the metric [86]. The lower bounds for all the metrics as seen in Table 3.1, especially the metrics related with Jacobian, were selected to warrant the generation of valid hex-meshes [47].

To generate a valid mesh falling in the range of $[R_{min}, R_{max}] \subset [l_b^{M_i}, \min_i]$ of the current metric M_i from H_0, I proceed as follows. I start with the perturbation of vertex positions of a randomly selected element h_t. The perturbation is done by adding a small random vector $\vec{v_r}$ to the vertex position, where $|\vec{v_r}| \leq s$ and s is a percentage of the average edge length of H_0 (e.g., 0.1%). After modifying the selected vertex positions, I measure the metric M_i 's value of this element h_t and its neighbors. If one of these values falls in $[R_{min}, R_{max}]$, I return the obtained mesh. Otherwise, if all values are larger than R_{max}, I double the amount of noise, i.e., $s \leftarrow 2s$. Similarly, if all the values are smaller than R_{min}, I decrease s by half. This trial is performed for I_{max} times and returns *null* if no valid mesh can be found. $I_{max} = 5$ in all our experiments. Note that during the above perturbation, I constrain the distortion of the hex-mesh within the lower bounds of all the metrics to avoid inverted hexahedral elements, and the location of a modified vertex to be within its bounding sphere to prevent the tangling issue of hexahedral elements.

The ideal number of the generated hex-meshes at this step is $\alpha \times N_{min}$. However, this ideal number is typically not achievable. This is mainly because that the above perturbation may still generate invalid hex-meshes or redundant meshes whose metric values fall in the same sub-ranges of the previous meshes. These meshes have to be discarded. The $\mathcal{H}_{min}$ row of Table 7.1 lists the numbers of the generated hex-meshes for all the tested models after performing this step. Figure 7.1(a) visualizes the 59 hex-meshes ($\mathcal{H}_{min}$) of the Cube model using the parallel coordinate technique [42], where each curve and each vertical coordinate represent a hex-mesh and a metric listed in Table 3.1, respectively. From Figure 7.1(a), I see that the generated hex-meshes have well distributed values across the ranges of those minimum metrics (whose labels are colored in red). To this end, the hex-meshes in H_{min} have varying values of the minimum metrics but approximately constant values of the average metrics, as only a few hex elements are altered. Although the average metrics in Figure 7.1(a) exhibit certain variations, the actual ranges of their distribution are very small.

Second, for each hex-mesh H_i in $\mathcal{H}_{min}$, I perturb the positions of a large portion of vertices of H_i in order to alter their average metric values. Similar to the first step, for each average metric A_k (assuming the total number of the average metrics is N_{ave}) with value ave_k of H_i that has its lower bound $l_b^{A_k}$ and $\uparrow$ trend, I divide its valid space $\left[l_b^{A_k}, ave_k\right]$ into β sub-ranges. Using a similar perturbation strategy described earlier to a random set of interior vertices, I aim to generate a hex-mesh with the value of metric A_k falling in the given sub-range. At this step, the maximal number of the generated hex-meshes is $\beta \times N_{ave}\mathcal{H}_{min}$, where N_{ave} is 10 corresponding

to the metrics with blue colored labels in Figure 7.1. However, similar to the first step, our perturbation does not guarantee to always generate a hex-mesh with the average metric value falling in a given sub-range, as modifying a large amount of vertices is harder to control than altering just a few. Therefore, the actual number of obtained valid meshes is much smaller than the ideal number. The $\mathcal{H}_{ave}$ row of Table 7.1 lists the numbers of the generated hex-meshes for all the tested models after performing this step. Figure 7.1(b) visualizes the 1677 hex-meshes ($\mathcal{H}_{ave}$) of the Cube model. From Figure 7.1(b), I see that the generated hex-meshes have reasonably well distributed values across the ranges of the average metrics that I control. The obtained meshes in $\mathcal{H}_{ave}$ will be used in our correlation analysis.

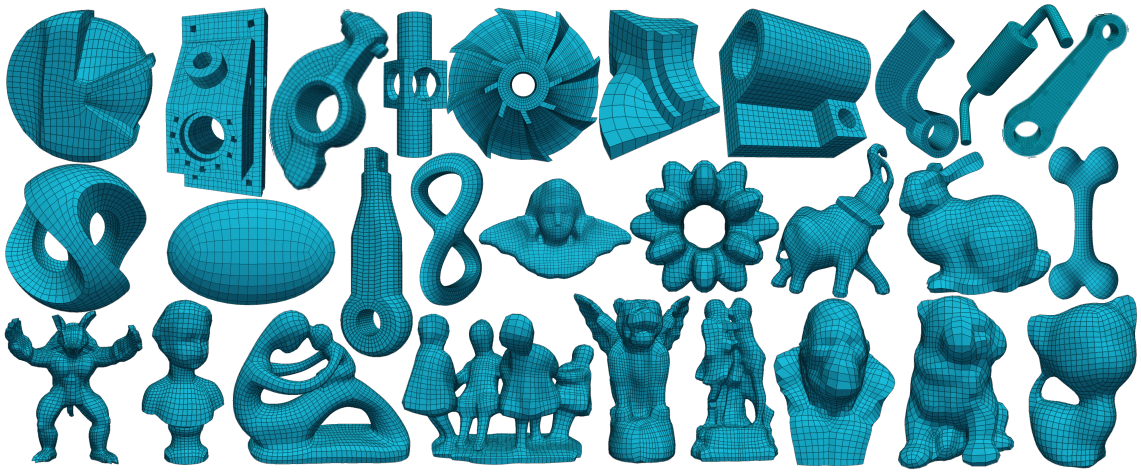


Figure 7.3: Models used in our experiments.

Note that the above perturbation process does not alter the structure of the hex-meshes[27] nor the number of hex elements. To cover a wide spectrum of shapes, I consider over 20 different 3D man-made or natural objects (Figure 7.3) with various geometric and topological complexities. For *each* model, I generate a large set of

Table 7.1: Datasets for six representative models (out of 20 models). Rows without * are for dataset generation, while Memory* is the memory cost of each mesh for solving the linear-elasticity problem and Timing* is for solving this problem of all the sampled meshes for each model.

Models	Cube	Sphere	Fandisk	Hanger	Bunny	Fertility
#H	216	135	357	939	17226	13584
α	8	8	8	8	8	8
$\mathcal{H}_{Min}$	61	61	58	58	63	58
β	12	8	15	25	8	8
$\mathcal{H}_{Ave}$	1676	1176	2389	3131	1698	1508
Timing	2h	1.5h	4h	8h	30h	25h
Memory*	8M	6M	10M	100M	200M	180M
Timing*	2h	1.5h	4h	7h	30h	25h

hex-meshes using the above described technique.

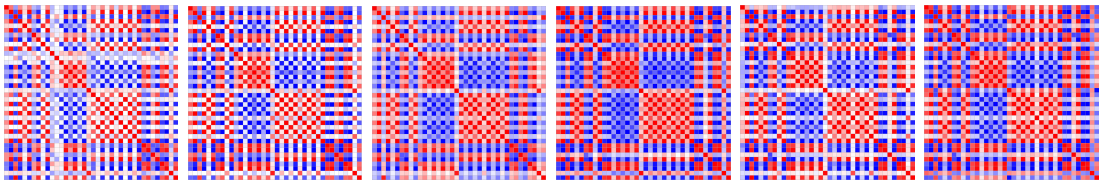
As as seen in Table 7.1, #H row lists the numbers of hexahedral elements for six representative models. The timing of the generation is also provided for each model. All the timing information was recorded on a PC with an Intel Xeon (E5-1620) processor and 16GB memory.

7.2 Evaluation

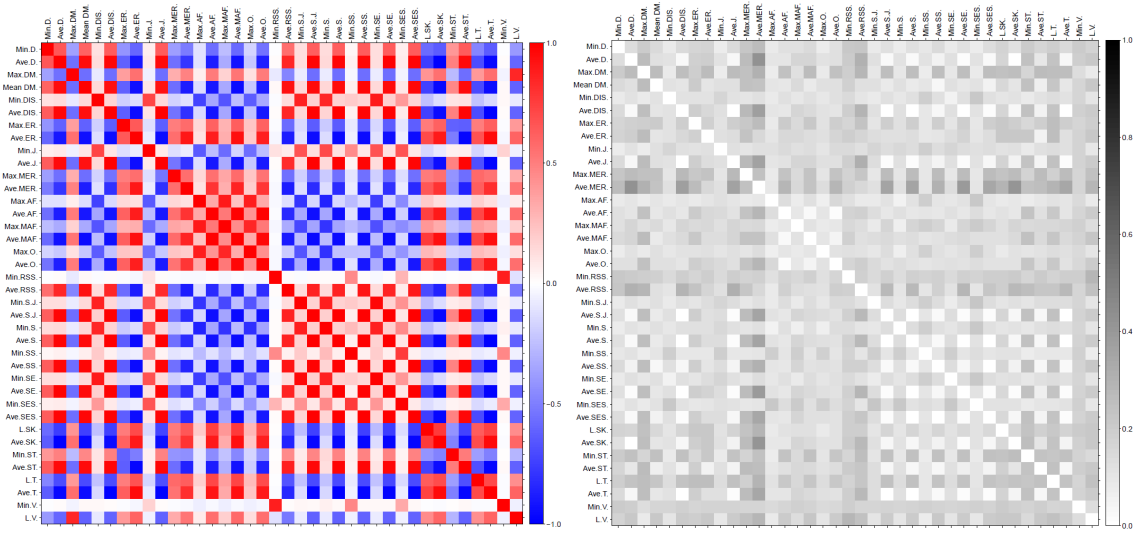
By applying the techniques described in Sections 7.1.1 and 7.1.2 to the generated hex-mesh datasets (Section 7.1.3), I evaluated the effectiveness of various quality metrics for hex-meshes and their dependencies. Our evaluation mainly contains two parts. I first investigated the relationships among the quality metrics for hex-meshes without considering subsequent applications in Section 7.2.1. The goal here was to comprehend the correlation among different metrics so that the used quality

Algorithm 4: Pseudo code of dataset generation for a model

Input : H_0, α, β
Output : $\mathcal{H}_{min}$, $\mathcal{H}_{ave}$
Constant: $\bar{e}$ is the average edge length of H_0
foreach *minimum metric* M_i *that has a lower bound* $l_b^{M_i}$ *and* $\uparrow$ *trends* **do**
    calculate the value min_i of metric M_i of H_0;
    evenly divide the range of metric M_i, $\left[l_b^{M_i}, \min_i\right]$, into α sub-ranges;
    $\sigma = (min_i - l_b^{M_i})/(\alpha - 1)$;
    for $j = 0 : \alpha - 1$ **do**
        compute the sub-range $[R_{min}, R_{max}]$, where $R_{min} = l_b^{M_i} + j\sigma$ and $R_{max} = l_b^{M_i} + (j+1)\sigma$;
        generate a valid hex-mesh and put it into set $\mathcal{H}_{min}$;
foreach *hex-mesh* H_i *in set* $\mathcal{H}_{min}$ **do**
    foreach *average metric* A_k *that has a lower bound* $l_b^{A_k}$ *and* $\uparrow$ *trends* **do**
        calculate the value ave_k of metric A_k of H_i;
        evenly divide the range of metric A_k, $\left[l_b^{A_k}, ave_k\right]$, into β sub-ranges;
        calculate the value ave_k of metric A_k of H_i;
        evenly divide the range of metric A_k, $\left[l_b^{A_k}, ave_k\right]$, into β sub-ranges;
        $\sigma = (ave_k - l_b^{A_k})/(\beta - 1)$;
        for $j = 0 : \beta - 1$ **do**
            compute the sub-range $[R_{min}, R_{max}]$, where $R_{min} = l_b^{A_k} + j\sigma$ and $R_{max} = l_b^{A_k} + (j+1)\sigma$;
            generated a valid hex-mesh and put it into set $\mathcal{H}_{ave}$;



(a) Block (b) Bone (c) Bust (d) Double-torus (e) Ellipsoid (f) Hanger



(g) Averaged correlation matrices of allmeshes (h) Deviation

Figure 7.4: For the Fandisk, Hanger, and Bunny models, the metric, Max. DM, has quite weak correlations with other metrics. This is mainly because of the Max. DM values of the corresponding hex-mesh datasets have a too narrow range.

metrics for different applications were reduced to a small representative set of metrics. Second, I performed the same analysis but at the same time considered three specific applications – linear-elasticity problem (Section 7.2.2.1), Poisson equation solving (Section 7.2.2.2), and Stoke equation solving (Section 7.2.2.3). The goal here is two-fold: 1) to validate the results of the preceding evaluation and 2) to identify the most effective set of metrics for the measurement of the quality of hex-meshes in response to the specific needs of these applications.

7.2.1 Application Independent Study

Our first evaluation measured the correlation among different metrics (Section 7.1.1) and their similarity (Section 7.1.2) without considering any specific applications. In particular, for each model I computed its correlation matrix C based on its dataset $D[N, M]$ (e.g. $D[N, M] = D[38, 1677]$ for the Cube model that has 1677 hex-mesh samples). Based on C, I classified these metrics into different groups (Section 7.1.2).

Figure 7.4(a-f) visualizes the correlation matrices of all the metrics listed in Table 3.1 for six representative models. Each cell of the correlation matrix was colored based on the correlation coefficient of the associated pair of metrics. It is either in red or blue, which means the two metrics have positive or negative correlation, respectively. The darker the color, the stronger the correlation.

To minimize the potential bias caused by a specific model to the relationships among metrics, I computed a simple averaged-correlation matrix $\bar{C}[N, N]$ (Figure 7.4(g)) of all the correlation matrices. From Figure 7.4, I observed that the correlation matrices for all the models as well as the averaged one have similar patterns, which indicates that the behaviors of the quality metrics may be independent from the various characteristics of the tested models. Based on $\bar{C}[N, N]$, I performed the Ward's hierarchical-clustering (Section 7.1.2). Figure 7.5 (a) shows the obtained binary tree describing the hierarchical-clustering of the metrics. Note that the distance among any pair of metrics is seen in the Y axis.

Although a priori does not exist on how many clusters these metrics should be classified into, in this work, I extracted 8 groups (highlighted by the red rectangles

in Figure 7.5), without loss of generality. The pairwise distance of metrics in the same clusters is less than 1.

From these clusters, I obtained the following observations.

(1) Different average metrics had closer relationships while minimum or maximum metrics are more akin to each other.

(2) Metrics related to operations on the element volume such as {Min. SS, Min. SES, Min. RSS, and Min. V.}, and {Ave. SS, Ave. SES, and Ave. RSS.} belong to the same clusters (i.e., cluster 1 and cluster 5, respectively), while Max. O., Max. AF, and Max. MAF. belong to cluster 2 that measures the deviations of the metric tensor from the identity matrix. By inspecting the largest cluster (i.e., cluster 5) that contains most of the average metrics, I saw that although L. SK. and L. T. metrics have names of the "largest", they were actually more related to other average metrics. This insight was not easily obtained by comparing their formulas.

(3) The small difference in their calculations of Min. J., Min. DIS., and Min. S. has been reflected in the clustering, where they belong to cluster 3. The close relations between metrics, such as Min. SE, Min. S. J. that measure the angle distortions have also been identified (see cluster 3) by the hierarchical-clustering.

(4) Max. ER., Min. ST., Min. D. (in cluster 6), and Max. MER. (in cluster 8) calculated based on the ratios of edge lengths or diagonal lengths were highly related and have been clustered together.

With this clustering result, researchers focusing on hex-mesh generation techniques can use a smaller number of metrics to evaluate the quality of the generated

hex-meshes, i.e., selecting one representative metric from each cluster. However, this only provides a general guideline when there is no specific application involved.

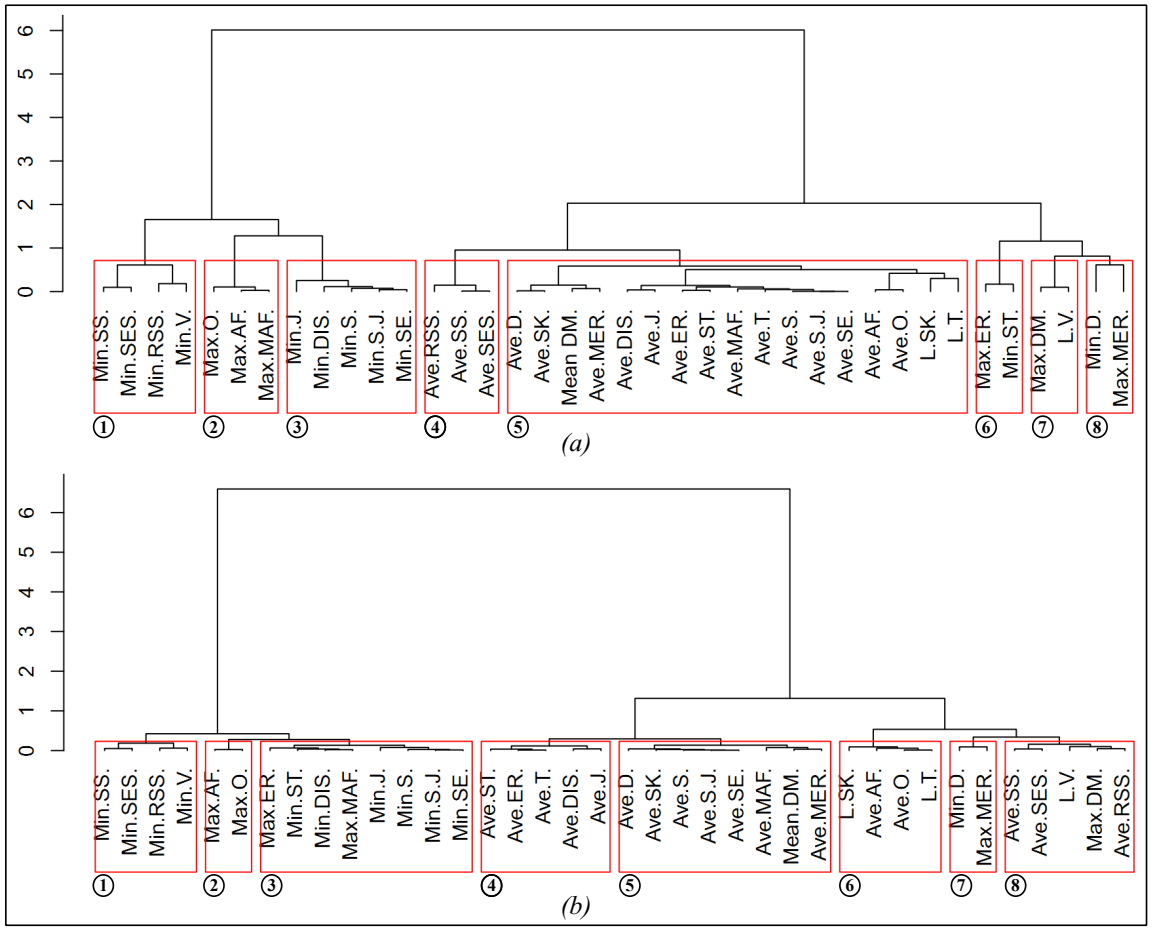


Figure 7.5: Hierarchical-clustering results of the metrics listed in Table 3.1 (a) without and (b) with linear elastic application considered.

7.2.2 Application Dependent Study

After studying the correlation among metrics under the application-independent assumption, to understand the impact of different hex-mesh quality metrics on FEA

based simulations, I further investigate their influences on a hex-mesh to three well-studied applications, i.e., linear-elasticity, the Poisson's equation and Stokes problems. Other applications can be analyzed in a similar way.

7.2.2.1 The Linear Elasticity

As our first application, linear-elasticity models an elastically deformable body under infinitesimal displacement. According to Bathe [2], isotropic linear elastic energy is an quadratic energy of the displacement field v, taking the following form:

$$E = \int_{\Omega} \mu\epsilon : \epsilon + \frac{\lambda}{2}\mathbf{tr}^2(\epsilon)dx,$$

where $\epsilon = (\nabla v + \nabla v^T)/2 - \mathbf{I}$ is the infinitesimal strain tensor and μ, λ are *Lamé*'s coefficients. For all three applications, I discretize these continuous energies using the FEM method with linear shape function and Gauss quadrature to approximate the per-hex integral. The induced Euler-Lagrangian equation is an elliptic PDE. So that I have a linear system in the discrete case whose left hand side K is the stiffness matrix. To extract the minimal and maximal non-trivial eigenpairs of this matrix, I optimize the Rayleigh quotient $x^T Ax/x^T x$ using nonlinear conjugate gradient method (NCG) [26] with approximate inverse preconditioner [5].

7.2.2.2 Poisson's Equation Related Application

Our second application, Poisson's equation, sees a lot of applications in the field of computer graphics. It can be used to reconstruct shape from differential domain [45], to find the potential component of a vector field [92], or to find smooth function

extensions from boundary values [36]. The solution to the Poisson's problem is the minimizer of the Dirichlet energy of an scalar function f:

$$E = \int_{\Omega} \|\nabla f\|^2 dx.$$

Again, the Euler-Lagrangian equation is an elliptic PDE and the resulting discrete system is linear, where the left hand size matrix A corresponds to the Laplace operator. The eigenpairs are extracted in the same way as Section 7.2.2.1 from A.

7.2.2.3 Stoke Equation Related Application

Finally, the stokes equation, which models fluid motion with high viscosity and low Reynolds number. They can also found applications in computational fabrication [99] and physics based animation [3]. Similarly, following FEM, it can be discretized by minimizing the component wise Dirichlet energy under the incompressible constraints:

$$E = \int_{\Omega} \|\nabla v\|^2 dx \quad \textbf{s.t. } \nabla \cdot v = 0,$$

where v is the velocity field. This system can still be considered as an elliptic PDE in the subspace specified by the constraints. So that the resulting system is a saddle point system with the following left hand side:

$$\begin{pmatrix} A_v & B^T \\ B & 0 \end{pmatrix},$$

where $A_v = A \otimes I^{3\times 3}$ is the component-wise Laplace operator and B is the divergence operator. The system can be solved only for the primal variables using the Schur

complement matrix: $A_v^{-1} - A_v^{-1}B^T(BA_v^{-1}B^T)^{-1}BA_v^{-1}$ as the left hand side. It is the non-trivial eigenpairs of this reduced system that I care about. To extract these eigenpairs while avoid evaluating this dense system directly, I instead use A_v as the left hand side and project each search direction d of NCG onto the constraint subspace by solving:

$$\mathbf{argmin}_{d^*}\|d^* - d\|^2 \quad \mathbf{s.t.}\ Bd^* = 0, \tag{7.5}$$

to get the equivalent result [46]. Equation 7.5 can be solved very efficiently by pre-factorizing the left hand side of the dual system: BB^T using Cholmod [15]. d^* was then found from: $d^* = (I - B^T(BB^T)^{-1}B)d$.

7.2.2.4 Application Dependent Analysis

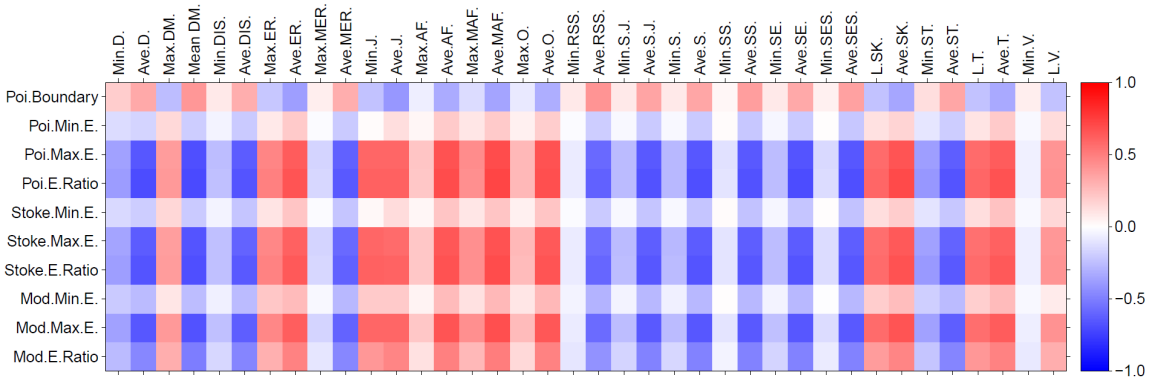


Figure 7.6: The illustration of the correlations of metrics listed in Table 3.1 with the minimum and maximum eigenvalues of the three applications after averaging the correlation matrices of all models.

To investigate the impact of the hex-mesh quality metrics on the above three applications, I include the maximal and minimal eigenvalues of the above individual systems, denoted by Max. E. and Min. E., respectively, to the correlation matrix,

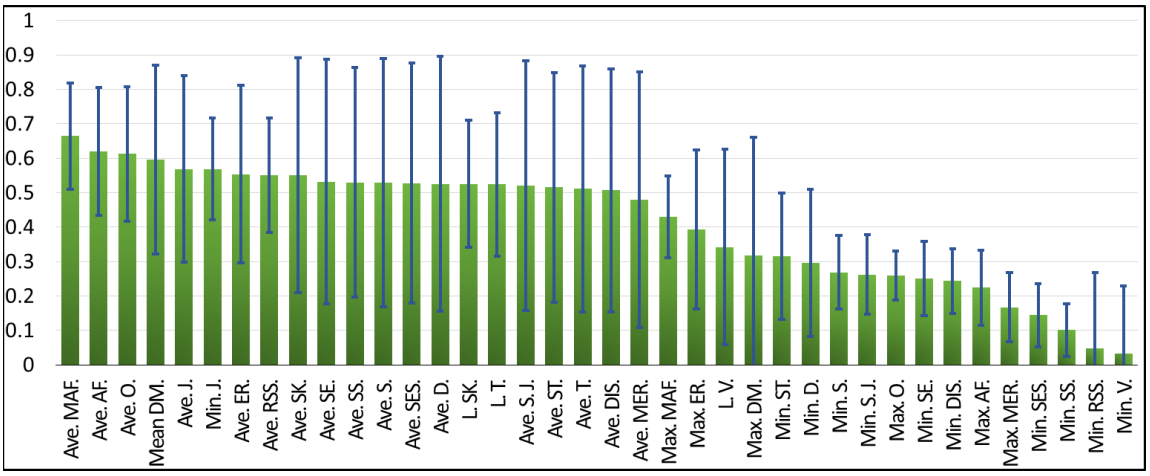


Figure 7.7: Metrics ranking based on their correlations with Max. E. of the linear-elasticity problem.

as shown in Figure 7.6. Based on the *absolute* values of their correlation coefficients with Min. E. and Max. E., I rank all the existing metrics as shown in Figures (7.7-7.12). From these results, I see that the correlation analysis results exhibit very similar patterns across all three different applications. I also observe that, for all three applications,

(1) Most average metrics have stronger correlation with both the maximal and minimal eigenvalues than those minimum/maximum metrics. Among those metrics, Ave. MAF. has the strongest correlation with the maximal eigenvalue for all three applications, while Ave. SES. and Ave. SE ranked top two with respect to the minimal eigenvalue.

(2) The correlations of the metrics (especially average-related metrics) with the maximal eigenvalues are typically much stronger than their correlations with the minimal eigenvalues. To explain such a behavior, I provide a qualitative analysis as

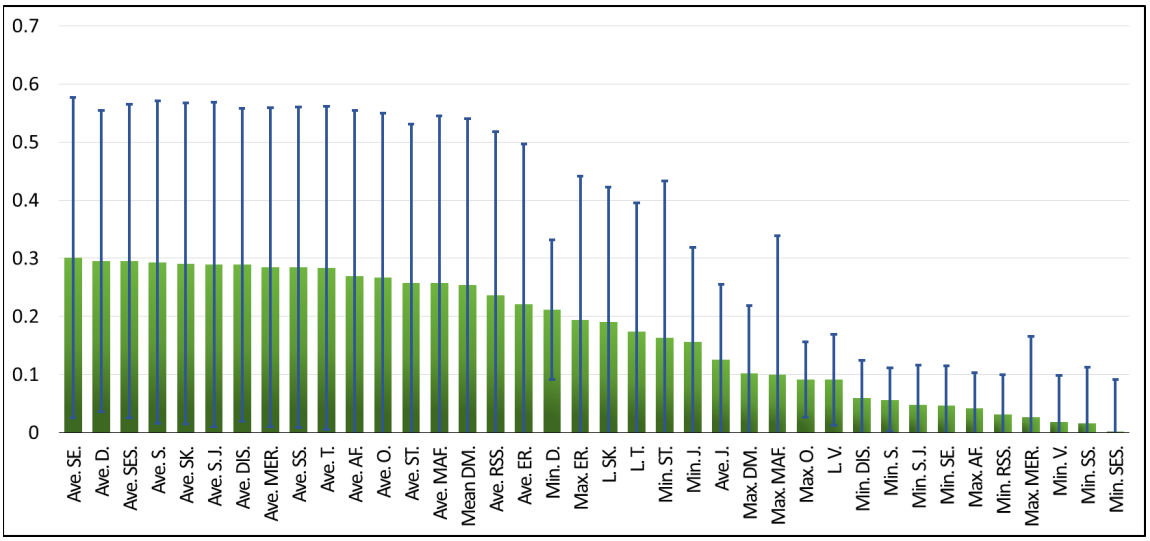


Figure 7.8: Metrics ranking based on their correlations with Min. E. of the linear-elasticity problem.

follows.

Indeed, the maximal eigenvalue is an effective indicator of the low quality elements in the mesh. It is tightly related to condition number, i.e., the stability and performance of solving the system, especially when applying iterative solvers. By a closer look, all three energies above take the following form:

$$E = \int_{\Omega} Q(\nabla f) dx = \sum_{\Omega_i} \int_{\Omega_i} Q(\nabla f) dx,$$

where Q is the energy density of quadratic form and f is a scalar or vector valued function over the hex domain. I can write this integral as a sum over all hex-elements Ω_i. In order for f to be the normalized maximal eigenvector, I need to assign f so that E is maximized. This can be accomplished by assigning $f \neq 0$ only in Ω_i with maximal $|\partial Q(\nabla f)/\partial f|$. But Q is same for all Ω_i, I are thus assigning $f \neq 0$ only in Ω_i with maximal $|\nabla f/\partial f|$. This last equation is exactly the inversed Jacobian

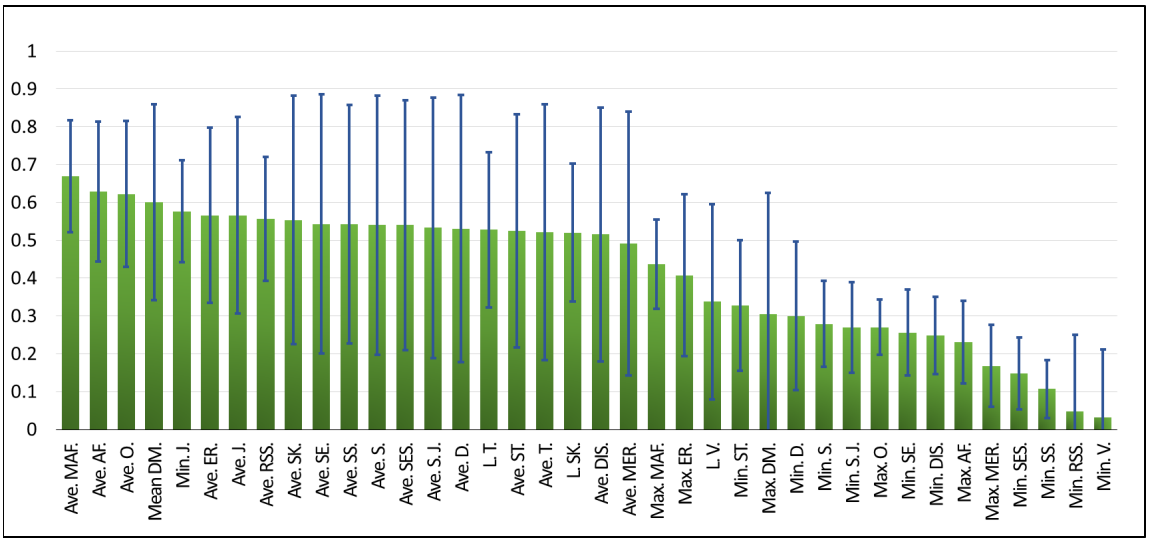


Figure 7.9: Metrics ranking based on their correlations with Max. E. of the Poisson problem.

of the hex-element Ω_i. Therefore, f indicates the hex-elements with low quality. One should notice that Q doesn't play an important role in this analysis, so that although the maximal eigenpair is computed in an application dependent way, it plays a similar role as typical application independent metrics that detect irregular sized or shaped hex-elements.

In order to get the true application dependent measure, the minimal non-trivial eigenpair should be used. Note that in order to discretize the energy functional $E : L^\infty \to R$, I are only looking at a finite dimensional subspace domain $\bar{L} \subset L^\infty$. And the minimal non-trivial eigenvalue is the minimal non-zero E available in $\bar{L}$, which must be larger than that available in L^∞. Thus, I can reach the conclusion that, unlike all the other existing metrics which measures the condition of an already discretized system, λ_{min} measures the accuracy of the discretization itself. Therefore, λ_{min} has low correlation with all the other metrics by revealing an orthogonal aspect

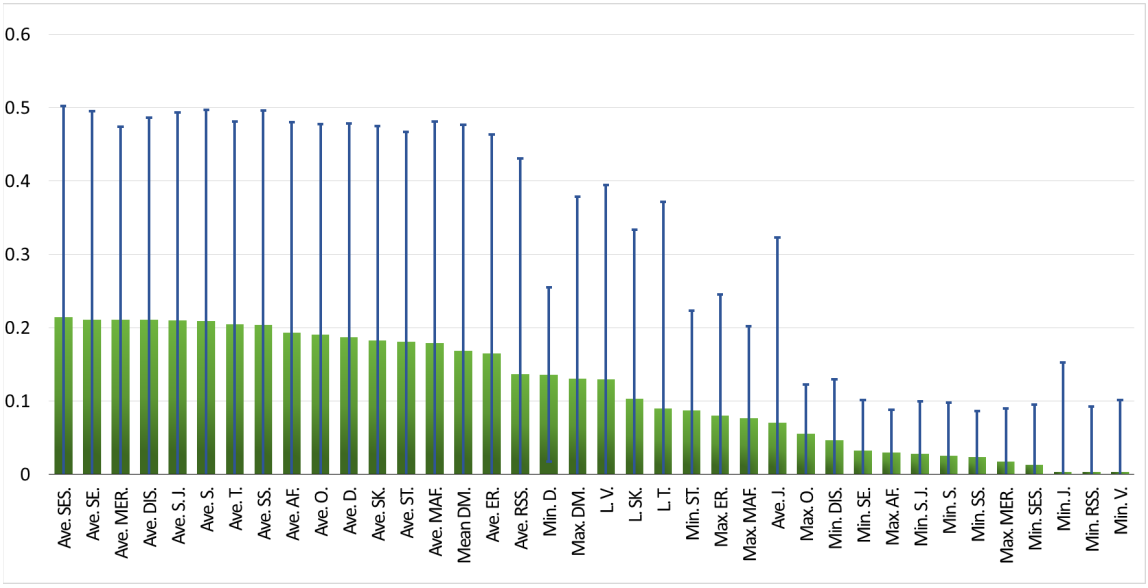


Figure 7.10: Metrics ranking based on their correlations with Max. E. of the Poisson problem.

of the discrete representation. A smaller λ_{min} indicates a better mesh for the specific application. The above analysis shows that higher quality meshes (measured by average quality metrics rather than by the minimum/maximum ones) will lead to smaller λ_{min}, which roughly indicates smaller L2 norm of error.

7.3 Conclusion

In this chapter, I investigate the correlations among various hex-mesh quality metrics using a linear-correlation analysis framework. Based on their pairwise covariance coefficients, I further calculate their similarities and agglomerate them in a hierarchical fashion. This helps us to classify similar metrics into a small number of groups, and thus reduces the number of necessary metrics for hex-mesh quality measurement

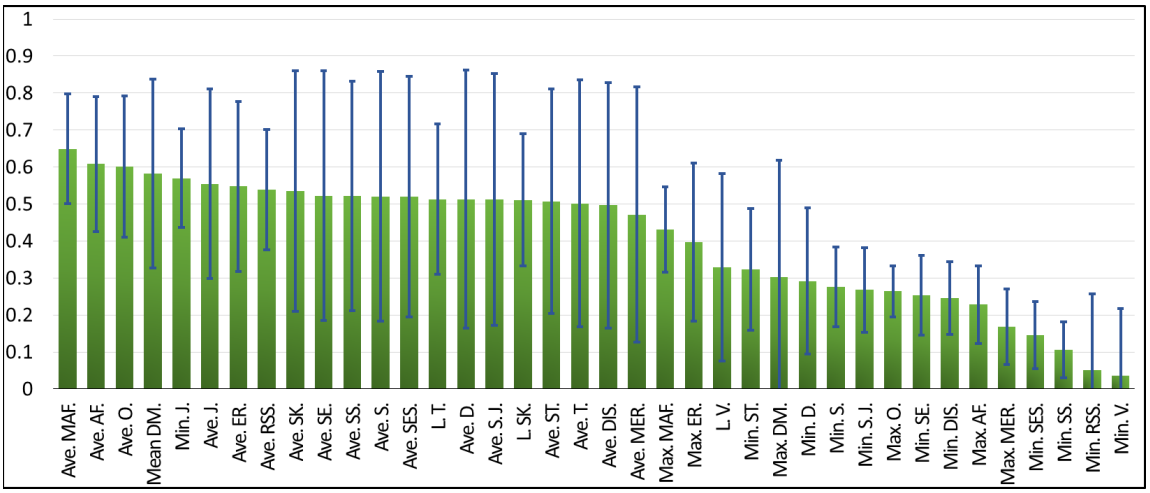
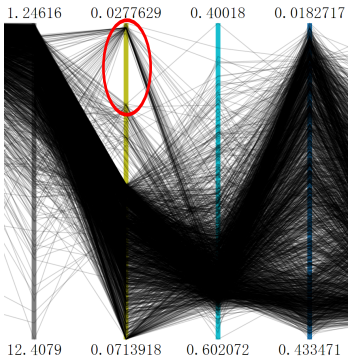


Figure 7.11: Metrics ranking based on their correlations with Max. E. of the Stoke problem.

in practice. Furthermore, with the proposed correlation analysis framework, I use linear-elasticity, Poisson's equation, and Stoke equation as example applications to evaluate the effectiveness of different metrics. In order to acquire the accurate outcome from the correlation analysis, I propose to generate a large set of hex-meshes for each model considered in a controllable fashion. The obtained sets of sample hex-meshes generally have well-distributed metric values.

Limitations and Future Work

The current work can be improved in the following ways. First, although I have obtained preliminary insights, due to the large set of metrics and their inter-dependencies, it is a challenging task to produce a hex-mesh dataset that uniformly samples the high dimensional metric space. Some artificial clusters can be observed and some of the data values will



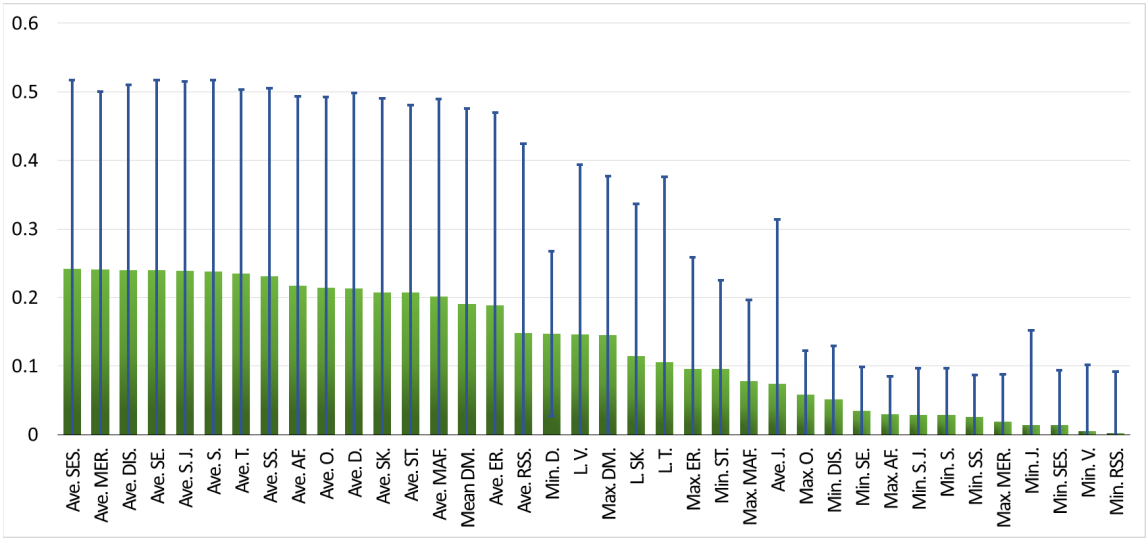


Figure 7.12: Metrics ranking based on their correlations with Min. E. of the Stoke problem.

never be sampled, see the inset figure for an example.

The current dataset generation could be improved by additionally controlling the maximum metrics and their corresponding average metrics. Another limitation of this work is that, I did not consider hex-meshes with inverted and tangled elements, which may lead to the underestimate of influences of Jacobian related metrics to three applications. Even though I have some interesting and potentially useful findings, the relations among metrics studied in this work are only linear, which may not be accurate enough. A more rigorous description of the relations among metrics will make our finding more consolidated. I would like to explore these directions in future work. Finally, based on the analytical results of our study, I will be able to design a more suitable objective function for the optimization of hex-meshes by considering the requirements of a given application.

Chapter 8

Conclusions

This chapter summaries the contributions of this dissertation and points out some major remaining problems in hex-meshing direction.

8.1 Summary

This dissertation attempts to address two major problems in hex-meshing: 1) obtain hex-meshes with simple global structures, and 2) identify critical metrics from the large amount of available quality metrics for future hex-meshing techniques. Specifically, I made the following contributions towards these two sub-directions:

Towards the direction of generating simple structured hex-meshes, I explicitly defined the singularity-structure composed of all the extraordinary edges in a hex-mesh, and the base-complex extracted from it guided by the singularity-structure.

base-complex contains a set of components that decompose a hex-mesh into different cube-like blocks. Within each component, user can fit high-order smoothness functions while only C^0 continuity can be achieved across the boundary of neighboring components. Based on the simple observation that fewer components will lead to larger smoothness volume throughout the whole volumetric-domain bounded by the surface of a hex-mesh, I proposed a series of techniques to produce hex-meshes with as few as possible components. First, given a 3D-volume domain bounded by a closed surface, I proposed to decompose a 3D-volume domain into a hex-mesh following a user designed skeletal structure, which is actually the singularity-structure, guided by a generalized sweeping strategy. For the first time, I achieved the simplicity of the skeletal structure, while having the generated hex-meshes follow the orientations specified by users. Second, given a hex-mesh as input, by assuming that the extracted singularity-structure of the given hex-mesh has been already well placed, I proposed to reduce the number of components in the base-complex of the input mesh by correcting the misalignments of singularities. Our proposed alignment technique first connects misaligned singularities in a zigzag way, and then improved the geometric fidelity through a structure-aware parameterization optimization. Third, I proposed a new structure simplification framework to solve a harder problem by making no assumption on the input structure. Since the misalignment correcting technique cannot be trivially extended, I employed a new pipeline by first constructing the simplified graph of the to-be-simplified structure without considering any geometric quality, and then parameterizing the involved patch domains to directly obtain the simplified structure while reducing the geometric distortion.

Towards identifying the critical quality metrics for hex-meshing (e.g., generation, optimization, and processing), I proposed a comprehensive yet complete pipeline to quantitatively evaluate the correlations among quality metrics, and their relationships with certain applications. Because of the non explicit relations among the metrics, I was the first to employ the linear-correlation analysis to obtain their approximate relationships. To obtain reliant analytical results, based on a hex-mesh input, I proposed a hex-mesh dataset generation algorithm to generate hex-mesh samples with statistically sound distributions over the non-orthogonal multi-dimensions, where each dimension represents a shape metric. Furthermore, I also investigated the correlations between shape metrics and linear-elasticity, Poisson equation, and naive Stoke problems, respectively. Our framework can also be employed to evaluate measurements for other representations, e.g., surfaces and images.

8.2 Future Work

Although I have made considerable contributions in directions of the simple global structure and quality measurements for hex-meshing, techniques and theories in structural hex-meshing and quality measurements for hex-meshing are still far away from being mature. Below I outline our thoughts on those unsolved and valuable problems in this field.

Quality of global structure. In this dissertation, I demonstrated that a smaller number of components in a base-complex is preferred for volumetric B-spline-fitting.

However, for models with complex shapes, coarse structures often lead to bad element quality and surface geometric deficits. This leads to a question, is the fewer components the better quality of the global structure? For other attributes of the structure, e.g., the number of singularities, the similarity of each component to a cube, volume changes of neighboring components within the same global structure, there still lack explicit quantitative measurements.

What is a good hex element? While I have studied the linear-correlations among the current available shape metrics, it would be beneficial to find a way to mathematically analyze the correlations between the shape of an element and the accuracy and convergency rate of numerical simulations. To address this, a valuable reference would be the "what is a good linear element?- Interpolation, Conditioning, Anisotropy, and Quality Measures" [81]. Moreover, how the analytical results performed on a hex element can be applied to a general hex-mesh that consists of a set of hex elements? These call for the design of new and more descriptive metrics.

Linkage between inversion-free and topology. Hex-mesh optimization is vitally important in improving the quality of a hex-mesh to a user-desired level. Many practical algorithms have been developed to untangle a non inversion-free hex-mesh input. However, there still lacks a theoretical answer of whether an inversion-free hex-mesh is guaranteed to be obtained given any topologically valid all-hex-mesh as the input.

To conclude, hex-meshing is still a critical and fruitful research area that will continue seeing many major breakthroughs in the years to come.

Bibliography

[1] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[2] K. J. Bathe. *Finite Element Procedures in Engineering Analysis*. Prentice Hall, 1995.

[3] C. Batty and R. Bridson. A simple finite difference method for time-dependent, variable coefficient stokes flow on irregular domains. *arXiv preprint arXiv:1010.2832*, 2010.

[4] Y. Bazilevs, L. Beirao da Veiga, J. Cottrell, T. Hughes, and G. Sangalli. Isogeometric analysis: approximation, stability and error estimates for h-refined meshes. *Mathematical Models and Methods in Applied Sciences*, 16(07):1031–1090, 2006.

[5] M. Benzi, C. D. Meyer, and M. Tuma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 17(5):1135–1149, 1996.

[6] S. E. Benzley, E. Perry, K. Merkley, B. Clark, and G. Sjaardema. A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *In Proceedings, 4th International Meshing Roundtable*, pages 179–191, 1995.

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

[8] D. Bommes, T. Lempfer, and L. Kobbelt. Global structure optimization of quadrilateral meshes. *Computer Graphics Forum*, 30(2):375–384, 2011.

[9] D. Bommes, H. Zimmer, and L. Kobbelt. Mixed-integer quadrangulation. *ACM Transactions on Graphics*, 28(3):77:1–77:10, 2009.

[10] M. J. Borden, S. E. Benzley, and J. F. Shepherd. Hexahedral sheet extraction. In *Proceedings of the 11th International Meshing Roundtable*, pages 147–152, 2002.

[11] A. Bozzo, D. Panozzo, E. Puppo, N. Pietroni, and L. Rocca. Adaptive quad mesh simplification. In *Eurographics Italian Chapter Conference 2010*, pages 095–102, 2010.

[12] M. Brewer, L. Diachin, P. Knupp, T. Leurent, and D. Melander. The mesquite mesh quality improvement toolkit. In *Proceedings of the 12th International Meshing Roundtable*, pages 239–250, 2003.

[13] M. Campen, D. Bommes, and L. Kobbelt. Dual loops meshing: quality quad layouts on manifolds. *ACM Transactions on Graphics*, 31(4):110:1–110:11, 2012.

[14] J. Chawner. Quality and control - two reasons why structured grids aren't going away. http://www.pointwise.com/theconnector/March-2013/Structured-Grids-in-Pointwise.shtml.

[15] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3):22, 2008.

[16] P. Cignoni, L. D. Floriani, C. Montani, E. Puppo, and R. Scopigno. Multiresolution modeling and visualization of volume data based on simplicial complexes. In *Proceedings of the 1994 symposium on Volume visualization*, pages 19–26, 1994.

[17] E. Cohen, R. F. Riesenfeld, and G. Elber. *Geometric modeling with splines: an introduction*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[18] J. Cottrell, T. Hughes, and A. Reali. Studies of refinement and continuity in isogeometric structural analysis. *Computer Methods in Applied Mechanics and Engineering*, 196(41-44):4160–4183, 2007.

[19] J. Daniels, C. T. Silva, and E. Cohen. Semi-regular quadrilateral-only remeshing from simplified base domains. 28(5):1427–1435, 2009.

[20] J. Daniels, C. T. Silva, J. Shepherd, and E. Cohen. Quadrilateral mesh simplification. *ACM Transactions on Graphics*, 27(5):148:1–148:9, 2008.

[21] J. Daniels, II, C. T. Silva, and E. Cohen. Localized quadrilateral coarsening. In *Proceedings of the Symposium on Geometry Processing*, SGP '09, pages 1437–1444, 2009.

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

[23] S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer Aided Geometric Design*, 22(5):392–423, 2005.

[24] M. Floater. Mean-value coordinates. *Computer Aided Design*, 20:19–27, 2003.

[25] M. S. Floater. Mean-value coordinates. *Computer aided geometric design*, 20(1):19–27, 2003.

[26] G. Gambolati, F. Sartoretto, and P. Florian. An orthogonal accelerated deflation technique for large symmetric eigenproblems. *Computer Methods in Applied Mechanics and Engineering*, 94(1):13–23, 1992.

[27] X. Gao, Z. Deng, and G. Chen. Hexahedral mesh re-parameterization from aligned base-complex. *ACM Transactions on Graphics*, 34(4):142:1–142:10, 2015.

[28] X. Gao, J. Huang, S. Li, Z. Deng, and G. Chen. An evaluation of the quality of hexahedral meshes via modal analysis. In *1st Workshop on Structured Meshing: Theory, Applications, and Evaluation*, 2014.

[29] X. Gao, J. Huang, K. Xu, Z. Pan, Z. Deng, and G. Chen. Evaluating hex-mesh quality metrics via correlation analysis. Under submission.

[30] X. Gao, T. Martin, S. Deng, E. Cohen, Z. Deng, and G. Chen. Structured volume decomposition via generalized sweeping. *IEEE Transactions on Visualization and Computer Graphics*, 2015.

[31] X. Gao, W. Wang, Z. Deng, and G. Chen. Practical structure simplification for hex re-meshing. Under submission.

[32] J. Gregson, A. Sheffer, and E. Zhang. All-hex mesh generation via volumetric polycube deformation. *Computer Graphics Forum*, 30(5):1407–1416, 2011.

[33] M. H. Gutknecht. A brief introduction to krylov space methods for solving linear systems. In *Frontiers of Computational Science*, pages 53–62. Springer, 2007.

[34] D. Hacon and C. Tomei. Tetrahedral decompositions of hexahedral meshes. *Eur. J. Comb.*, 10(5):435–443, 1989.

[35] Y. He, X. Xiao, and H.-S. Seah. Harmonic 1-form based skeleton-extraction from examples. *Graphical Models*, 71(2):49–62, 2009.

[36] K. Hormann, B. Lévy, and A. Sheffer. Mesh parameterization: Theory and practice. 2007.

[37] H. W. Hua Zhu, Jinming Chen and S. Gao. Direct editing on hexahedral mesh through dual operations. In *Proceedings of the 23th International Meshing Roundtable*, pages 467–484. Springer, 2014.

[38] J. Huang, T. Jiang, Z. Shi, Y. Tong, H. Bao, and M. Desbrun. L1-based construction of polycube maps from complex shapes. *ACM Transactions on Graphics*, 33(3):25:1–25:11, 2014.

[39] J. Huang, Y. Tong, H. Wei, and H. Bao. Boundary aligned smooth 3d cross-frame-field. *ACM Transactions on Graphics*, 30(6):143:1–143:8, 2011.

[40] T. J. R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Civil and Mechanical Engineering. Dover, 2000.

[41] B. Y. Hughes T.J., Cottrell J.A. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry, and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2005.

[42] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91, 1985.

[43] T. Jiang, J. Huang, Y. T. Yuanzhen Wang, and H. Bao. Frame field singularity correction for automatic hexahedralization. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1189–1199, 2014.

[44] T. Ju, S. Schaefer, and J. Warren. Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics*, 24(3):561–566, 2005.

[45] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 61–70, 2006.

[46] C. Keller, N. I. Gould, and A. J. Wathen. Constraint preconditioning for indefinite linear systems. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1300–1317, 2000.

[47] P. M. Knupp. Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part ii-a framework for volume mesh optimization and the condition number of the jacobian matrix. *International Journal for Numerical Methods in Engineering*, (8):1165–1185, 2000.

[48] P. M. Knupp. Hexahedral and tetrahedral mesh untangling. *Engineering with Computers*, 17(3):261–268, 2001.

[49] F. Ledoux and J. Shepherd. Topological modifications of hexahedral meshes via sheet operations: a theoretical study. *Engineering with Computers*, 26(4):433–447, 2010.

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

[51] B. Li, X. Li, K. Wang, and H. Qin. Surface mesh to volumetric spline conversion with generalized poly-cubes. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1539–1551, 2013.

[52] Y. Li, Y. Liu, W. Xu, W. Wang, and B. Guo. All-hex-meshing using singularity-restricted field. *ACM Transactions on Graphics*, 31(6):177:1–177:11, 2012.

[53] M. Livesu, A. Sheffer, N. Vining, and M. Tarini. Practical hex-mesh optimization via edge-cone rectification. *Transactions on Graphics (Proc. SIGGRAPH 2015)*, 34(4), 2015.

[54] M. Livesu, A. Sheffer, N. Vining, and M. Tarini. Practical hex-mesh optimization via edge-cone rectification. *Transactions on Graphics (Proc. SIGGRAPH 2015)*, 34(4):141:1–141:11, 2015.

[55] M. Livesu, N. Vining, A. Sheffer, J. Gregson, and R. Scateni. Polycut: monotone graph-cuts for polycube base-complex construction. *ACM Transactions on Graphics*, 32(6):171, 2013.

[56] M. Livesu, N. Vining, A. Sheffer, J. Gregson, and R. Scateni. Polycut: Monotone graph-cuts for polycube base-complex construction. *ACM Transactions on Graphics*, 32(6):171:1–171:12, 2013.

[57] L. Maréchal. Advances in octree-based all-hexahedral mesh generation: handling sharp features. In *proceedings of the 18th International Meshing Roundtable*, pages 65–84. Springer, 2009.

[58] T. Martin, G. Chen, S. Musuvathy, E. Cohen, and C. D. Hansen. Generalized swept mid-structure for polygonal models. *Computer Graphics Forum*, 31(2):805–814, 2012.

[59] T. Martin and E. Cohen. Volumetric parameterization of complex objects by respecting multiple materials. *Computers and Graphics*, 34(3):187–197, 2010.

[60] T. Martin, E. Cohen, and M. Kirby. Volumetric parameterization and trivariate B-spline-fitting using harmonic functions. In *Proceedings of the 2008 ACM symposium on Solid and Physical Modeling*, pages 269–280, 2008.

[61] MeshGems. Volume Meshing: MeshGems-Hexa. `http://meshgems.com/volume-meshing-meshgems-hexa.html`, 2015.

[62] P. Min. 3d mesh voxelizer. `http://www.google.com/search?q=binvox`, 2015.

[63] Y. Motooka, S. Noguchi, and H. Igarashi. Evaluation of hexahedral mesh quality for finite element method in electromagnetics. *Materials Science Forum*, 670:318–324, 2011.

[64] M. Muller-Hannemann, C. Kober, R. Sader, and H. florian Zeilhofer. *Anisotropic Validation of Hexahedral Meshes for Composite Materials in Biomechanics*. Freie Univ., Fachbereich Mathematik und Informatik, 2001.

[65] F. Murtagh and P. Legendre. Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion? *Journal of Classification*, 31(3):274–295, 2014.

[66] A. Myles, N. Pietroni, D. Kovacs, and D. Zorin. Feature-aligned t-meshes. *ACM Transactions on Graphics*, 29:117:1–117:11, 2010.

[67] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models in computer graphics. 25(4):809–836, 2006.

[68] X. Ni, M. Garland, and J. C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. *ACM Transactions on Graphics*, 23(3):613–622, Aug. 2004.

[69] M. Nieser, U. Reitebuch, and K. Polthier. Cubecover- parameterization of 3d volumes. *Computer Graphics Forum*, 30(5):1397–1406, 2011.

[70] F. S. Nooruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. *Visualization and Computer Graphics, IEEE Transactions on*, 9(2):191–205, 2003.

[71] S. J. Owen. A survey of unstructured mesh generation technology. In *Proceedings of the 7th International Meshing Roundtable*, pages 239–267, 1998.

[72] C.-H. Peng, E. Zhang, Y. Kobayashi, and P. Wonka. Connectivity editing for quadrilateral meshes. *ACM Transactions on Graphics*, 30(6):141:1–141:12, 2011.

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

[74] A. Ramos and J. Simões. Tetrahedral versus hexahedral finite elements in numerical modelling of the proximal femur. *Medical Engineering and Physics*, 28(9):916–924, 2006.

[75] M. D. Shapiro and M. B. Blaschko. On hausdorff distance measures. *Computer Vision Laboratory University of Massachusetts Amherst, MA*, 1003, 2004.

[76] A. Sheffer, M. Etzion, A. Rappoport, and M. Bercovier. Hexahedral mesh generation using the embedded voronoi graph. *Engineering with Computers*, 15(3):248–262, 1999.

[77] J. F. Shepherd. *Topologic and geometric constraint-based hexahedral mesh generation*. PhD thesis, The Scientific Computing and Imaging Institute at the University of Utah, 2007.

[78] J. F. Shepherd, M. W. Dewey, A. C. Woodbury, S. E. Benzley, M. L. Staten, and S. J. Owen. Adaptive mesh coarsening for quadrilateral and hexahedral meshes. *Finite Elem. Anal. Des.*, 46(1-2):17–32, 2010.

[79] J. F. Shepherd and C. R. Johnson. Hexahedral mesh generation constraints. *Engineering with Computers*, 24(3):195–213, 2008.

[80] J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1-3):21–74, 2002.

[81] J. R. Shewchuk. What is a good linear finite element? - interpolation, conditioning, anisotropy, and quality measures. 2002.

[82] H. Si. Tetgen: A quality tetrahedral mesh generator and three-dimensional delaunay triangulator. 2005.

[83] M. L. Staten, S. J. Owen, and T. D. Blacker. Unconstrained paving and plastering: A new idea for all hexahedral mesh generation. In *Proceedings of 14th International Meshing Roundtable*, pages 399–416, 2005.

[84] M. L. Staten, J. F. Shepherd, and K. Shimada. Mesh matching–creating conforming interfaces between hexahedral meshes. In *Proceedings of the 17th International Meshing Roundtable*, pages 467–484. Springer, 2008.

[85] S. M. Stigler. Francis galton's account of the invention of correlation. *Statistical Science*, 4(2):73–79, 1989.

[86] C. J. Stimpson, C. D. Ernst, P. Knupp, P. P. P. ebayand, and D. Thompson. The verdict geometric quality library. *SANDIA REPORT*, 2007.

[87] S. C. Tadepalli, A. Erdemir, and P. R. Cavanagh. A comparison of the performance of hexahedral and tetrahedral elements in finite element models of the foot. In *ASME 2010 Summer Bioengineering Conference, Parts A and B*, Naples, Florida, USA, 2010.

[88] M. Tarini, N. Pietroni, P. Cignoni, D. Panozzo, and E. Puppo. Practical quad mesh simplification. *Computer Graphics Forum (Special Issue of Eurographics 2010 Conference)*, 29(2):407–418, 2010.

[89] M. Tarini, E. Puppo, D. Panozzo, N. Pietroni, and P. Cignoni. Simple quad domains for field aligned mesh parametrization. *ACM Transactions on Graphics*, 30(6):142:1–142:12, 2011.

[90] T. J. Tautges. Moab-sd: integrated structured and unstructured mesh representation. *Engineering with Computers*, 20:286–293, 2004.

[91] T. J. Tautgesa and S. E. Knoopb. Topology modification of hexahedral meshes using atomic dual-based operations. *Algorithms*, 11:12, 2003.

[92] Y. Tong, S. Lombeyda, A. N. Hirani, and M. Desbrun. Discrete multiscale vector field decomposition. 22(3):445–452, 2003.

[93] M. ULTIGRID. An introduction to algebraic multigrid. *Computing in Science and Engineering*, 8:24, 2006.

[94] K. Wang, X. Li, B. Li, H. Xu, and H. Qin. Restricted trivariate polycube splines for volumetric data modeling. *IEEE Transactions on Visualization and Computer Graphics*, 18(5):703–716, 2012.

[95] R. Xu and I. Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Network*, 16(3):645, 2005.

[96] J. Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971.

[97] Y. Zhang, C. Bajaj, and G. Xu. Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. *Communications in Numerical Methods in Engineering*, 25(1):1–18, 2005.

[98] Y. Zhang, Y. Bazilevs, S. Goswami, C. L. Bajaj, and T. J. Hughes. Patient-specific vascular nurbs modeling for isogeometric analysis of blood flow. *Computer methods in applied mechanics and engineering*, 196(29):2943–2959, 2007.

[99] Y. Zhang, C. Yin, C. Zheng, and K. Zhou. Computational hydrographic printing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2015)*, 34(4):131:1–131:11, July 2015.