VARIATIONAL VOLUMETRIC MESHING

by

Saifeng Ni

APPROVED BY SUPERVISORY COMMITTEE:

Xiaohu Guo, Chair

Ovidiu Daescu

B. Prabhakaran

Sergey Bereg

Copyright © 2018

Saifeng Ni

All rights reserved

To my family.

VARIATIONAL VOLUMETRIC MESHING

by

SAIFENG NI, BS, MS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

December 2018

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my adviser, Dr. Xiaohu Guo, for his great guidance and support during my Ph.D. study. I really appreciate his time, knowledge and also his serious attitude on research yet easy-going personality.

Secondly, I would like to thank Dr. Wenping Wang at the University of Hong Kong, Dr. Yang Liu at Microsoft Research Asia, Dr. Jin Huang at Zhejiang University, Dr. Bruno Levy at Inria, Dr. Zhonggui Chen at Xiamen University, Dr. Zichun Zhong at Wayne State University for the professional opinions and assistance on my research.

Thirdly, I would like to show appreciation to all the other members of the supervisory committee: Dr. Balakrishman Prabhakaran, Dr. Ovidiu Daescu, Dr. Sergey Bereg, for your help on this dissertation.

Fourthly, I would like to give thanks to all my friends in Dallas, particularly my labmates in the UTD Computer Graphics & Animation (CGA) Lab: Yin Yang, Zichun Zhong, Liang Shuai, Yuan Tian, Yiqi Cai, Chao Wang, Chao Li, Baorong Yang, Yeqi Wang, Zheheng Zhao, Steven Hougue and Xiao Dong.

Last but not least, I am grateful to my family, for their love and support all the time. I want to say thank you to my husband, Xuming Zhai. Without his spiritual and technical support, it would be hard for me to accomplish this dissertation.

July 2018

VARIATIONAL VOLUMETRIC MESHING

Saifeng Ni, PhD
The University of Texas at Dallas, 2018

Supervising Professor: Xiaohu Guo, Chair

Domain discretization, also referred to as mesh generation, is one of the fundamental steps of many computation based applications. Although mesh generation techniques have evolved rapidly over the years, some volumetric meshing problems like sliver suppressing in tetrahedral meshing, field-aligned tetrahedral meshing, and hexahedral meshing are still not fully resolved. In this dissertation, we bring some insights to those problems.

This dissertation discusses variational-based methods to tackle mesh generation problems, i.e., we model these problems in the energy optimization framework. An energy which inhibits small heights is proposed to suppress almost all the badly-shaped elements in tetrahedral meshing. By iteratively optimizing vertex positions and mesh connectivity, slivers are harshly suppressed even in anisotropic tetrahedral meshing. Besides that, a particle-based field alignment framework is introduced. Specifically, a *Gaussian Hole Kernel* is constructed associated with each particle to constrain the formation of the desired one ring structure aligned with the frame field. The minimization of the sum of Gaussian hole kernels induces an inter-particle potential energy whose minimization encourages particles to have the desired layout. A cubic one ring structure leads to high quality hexahedral-dominant meshing. The one ring structures of the Body-Centered Cubic (BCC) and Face-Centered Cubic (FCC) lattice leads to high quality field-aligned tetrahedral meshing. This is the first time both Riemannian distance alignment and direction alignment problems have been considered in

tetrahedral meshing. Also, field-aligned tetrahedral meshing better preserves the rotation geometry and also creates better anisotropic meshes.

TABLE OF CONTENTS

ACKNOWLEDGMENTS v
ABSTRACT vi
LIST OF FIGURES x
LIST OF TABLES xv
CHAPTER 1 INTRODUCTION 1
CHAPTER 2 BACKGROUND AND RELATED WORK 4
  2.1 Backgrounds 4
    2.1.1 BCC and FCC Lattices 4
    2.1.2 Frame Field and Imagined Space 5
  2.2 Related Work on Tetrahedral Meshing 6
    2.2.1 Silver by Variational-method 8
  2.3 Hexahedral Meshing and Hexahedral-dominant Meshing 9
CHAPTER 3 SLIVER-SUPPRESSING TETRAHEDRAL MESHING 13
  3.1 Overview 13
  3.2 Shape Matching Triangulation Energy 14
    3.2.1 Edge-Based Shape Matching (ESM) 15
    3.2.2 Gradient-Based Shape Matching (GSM) 17
    3.2.3 The Importance of GSM Energy in Tetrahedral Meshing 20
  3.3 GSM Energy Optimization 21
    3.3.1 Vertex Smoothing 21
    3.3.2 Connectivity Update 23
    3.3.3 Boundary and Feature 24
  3.4 Experiment and Comparisons 26
    3.4.1 Quality Measurement 27
    3.4.2 Isotropic Tetrahedral Meshing 27
    3.4.3 With Sharp Features 28
    3.4.4 Adaptive and Anisotropic Tetrahedral Meshing 28
    3.4.5 Running Time and Convergence Analysis 32

3.4.6 Robustness 34

3.5 Conclusion 35

CHAPTER 4 FIELD ALIGNED FRAMEWORK 36

4.1 Overview 36

4.2 Particle-Based Optimization Method 36

4.2.1 Gaussian Hole Kernel 37

4.2.2 Lattice Optimization 39

4.3 Field Aligned Tetrahedral Meshing 46

4.3.1 Overview 46

4.3.2 Tetrahedral Mesh Generation 48

4.3.3 Experiments of Field Aligned Tetrahedral Meshing 48

4.3.4 Discussion and Future Work 61

4.4 Hexahedral-Dominant Meshing 62

4.4.1 Overview 62

4.4.2 Mesh Generation 65

4.4.3 Experiment and Comparison 70

4.4.4 Limitation and Future Work 82

CHAPTER 5 CONCLUSION AND FUTURE WORK 89

REFERENCES 90

BIOGRAPHICAL SKETCH 96

CURRICULUM VITAE

LIST OF FIGURES

2.1 BCC and FCC lattice 5

3.1 The illustration of edge-based shape matching for a 2-simplex. 16

3.2 The illustration of gradient-based shape matching for a 2-simplex. 18

3.3 Several typical badly-shaped tetrahedrons (Freitag and Knupp, 2002). 19

3.4 Comparison of ESM and GSM on the Bumpycube volume. The red ones are tetrahedrons with the smallest dihedral angles less than 18°, while the blue ones are tetrahedrons with the smallest dihedral angles less than 36°. 28

3.5 Duck volume meshing with 10,000 vertices. The red ones are tetrahedrons with the smallest dihedral angles less than 20, while the blue ones are tetrahedrons with smallest dihedral angles less than 40. 29

3.6 Fandisk volume meshing with 18,000 vertices. The red ones are tetrahedrons with the smallest dihedral angles less than 20, while the blue ones are tetrahedrons with the smallest dihedral angles less than 40. 30

3.7 Sphere volume meshing(10,000 vertices) with scaling field. The red ones are tetrahedrons with the smallest dihedral angles less than 15, while the blue ones are tetrahedrons with the smallest dihedral angles less than 30. 31

3.8 Sphere volume meshing (10,000 vertices) with sinusoidal variation of anisotropy. The red tetrahedrons are the ones with smallest dihedral angles less than 15, while the blue tetrahedrons are the ones with smallest dihedral angles less than 30. 31

3.9 Sphere volume meshing (10,000 vertices) with sinusoidal variation of anisotropy. The red tetrahedrons are the ones with smallest dihedral angles less than 15, while the blue tetrahedrons are the ones with smallest dihedral angles less than 30. 32

3.10 (a), (b), (c), (d) Dihedral angle and radius ratio quality changing along with computational time of the Duck model. (e) GSM energies changing along with the round numbers in different initializations. 34

3.11 Teddy volume meshes with different vertex numbers. The blue ones are tetrahedrons with the smallest dihedral angle less than 40°. Note: there's no tetrahedron with the smallest dihedral angle less than 20°. 35

4.1 Initial particle distribution, and particles before and after insertion and deletion. The gray ones are particles on the boundary. The red ones are particles inside the boundary. 40

4.2 The energy curve of Kitten Model about BCC optimization with 20 rounds of LBFGS optimization. The purple and green numbers are the number of being deleted and inserted at the end of each round respectively. The zoom-in view of the last 10 rounds is also provided. 46

4.3 Comparison with the traditional Gaussian kernel method proposed in Particle2013 (Zhong et al., 2013). The first row is the result on surface. The second shows the tetrahedra with $\theta_{min} < 40°$. The third row shows the clipping views. The following rows are histograms of dihedral angles, edge radius ratio, condition, and alignment error. 50

4.4 Comparisons with CVT and ODT on Fandisk model. The first row shows the clipping views. The second row shows the tetrahedra with $\theta_{min} < 40°$. The following rows are histograms of dihedral angles, edge radius ratio, condition, and alignment error. 53

4.5 Comparisons with CVT and ODT on Bimba model. The red tetrahedra shown in the second row have $\theta_{min} < 40°$. 54

4.6 BCC and FCC experiments on Torus with $\mathbf{I}$ and $\mathbf{R}_y$ rotation fields. The yellow ones are the clipping views. 56

4.7 BCC and FCC experiments on Fancyring with different rotation fields. The red ones are the tetrahedra with $\theta_{min} < 40°$. 57

4.8 Anisotropy variation along a single direction on Cube $[0.1, 1.1]^3$. The inverse of frame field is defined as $\mathbf{B} = diag\left(\left(1.0125 - e^{-|x-0.6|}\right)^{-1}, 1, 1\right)$. The second row shows the clipping views of the result tetrahedral meshes. The last four rows show the histograms of dihedral angle, edge radius ratio, condition, and alignment error. 59

4.9 Cylindrical anisotropy on Cube $[1, 11]^3$. The inverse of frame field is defined as $\mathbf{B} = \mathbf{S} * \mathbf{R}$, where $\mathbf{S} = diag\left(\left(1.05 - e^{-0.01|x^2+y^2-49|}\right)^{-1}, 1, 1\right)$, and the three columns of $\mathbf{R}$ are $\left(-x/\sqrt{x^2+y^2}, y/\sqrt{x^2+y^2}, 0\right)$, $\left(-y/\sqrt{x^2+y^2}, x/\sqrt{x^2+y^2}, 0\right)$, and $(0, 0, 1)$, respectively. The second row is the clipping view of the result tetrahedral meshes. The last four rows show the histograms of dihedral angle, edge radius ratio, condition, and alignment error. 60

4.10 The first two rows shows BCC and FCC results in different vertex numbers. The next two rows show the quality statistics and time consumption with different numbers of vertices on Teddy. 61

4.11 Field-aligned anisotropic triangular meshing on Cyclide, compared with Particle2013 (Zhong et al., 2013) and LCT (Fu et al., 2014). The three columns are the results of Particle2013 method, LCT method, and our method, respectively. The first two rows show the resulting surface meshes and the zoom-in views of the narrow part. r_6 shown in the first row is the ratio of vertices with degree 6 in the result mesh. The last two rows are the histograms of $\overline{\theta_{min}}$ and $\overline{\xi}$. 63

4.12 The particles and mesh results on Front_Upright_ASM005 Model. The three images on the left show the particle results in zoom-in views. The gray particles are on the domain boundary surface and the red ones inside the domain. The three images on the right show the corresponding hexahedral-dominant mesh in cut-out views. 64

4.13 The 20 rounds of L-BFGS optimization on the Fertility model. The optimization starts from a random initialization of 10,000 particles. The purple numbers are the numbers of particles being deleted after each round. The green numbers are the numbers of particles being inserted after each round. The blue numbers are the computation time in seconds. The second curve shows the zoom-in view of the last 10 rounds. The energy raise at the beginning of each round is caused by two reasons: one is particle insertion and deletion; the other one is particle projection. 65

4.14 Optimization of particles on a 2D plane with different weights w. 72

4.15 Hexahedral meshing with different frame fields on Cube and Warped Cube models. (a) $\mathbf{T} = \mathbf{I}$; (b) stretch along x-axis; (c) linear scale along x axis, first linearly decreasing then linearly increasing; (d) scale linearly increasing along x axis; (e) shear along y; (f) shear along both y and z. 74

4.16 Three different frame fields are tried on the Cylinder model. The first and second rows show our results with closed-form frame fields, with and without scale correction along the radius, respectively. The third and fourth rows show our Particle and Gao et al.'s results with a discrete frame field generated by their method. The last column shows the corresponding non-hex elements. Non-hex elements including prisms, pyramids, and tetrahedra are shown in green, blue, and red, respectively. In Gao et al.'s results, non-hex polyhedra with different number of faces are rendered in different colors, including 6-face polyhedra with triangle faces shown in green color. Our result in the first row does not have non-hex element. 75

4.17 A comparison of our particle (left) and L_p-CVT (right) methods on Anc101 model. The first row is the result mesh on domain boundary. The second row shows the non-hex elements including prisms, pyramids, and tetrahedra in green, blue, and red, respectively. The third rows is the clipping view. The fourth row shows the histograms of alignment errors measured in degrees. 77

4.18 Comparison between our method (left) and PGP3D (right) methods on Bunny model. The non-hex elements are shown in the second row, including prisms, pyramids, and tetrahedra rendered in green, blue, and red, respectively. The third row is the clipping view. The bar graphs in the fourth row shows the ratio of numbers of all elements including hexahedra (H), prisms (Pr), pyramids (Py) and tetrahedra (T). Note that the views of non-hex elements in the second row may give you the wrong impression of more non-hex elements in our result, which is caused by 3D projection. Non-hex elements in different depth are overlaid together. The following two rows are good proof of the better quality of our results. 78

4.19 Comparison between our method and PGP3D method on Cubo model. The non-hex elements including prisms, pyramids, and tetrahedra are rendered in green, blue, and red, respectively. The bar graph shows the ratio of numbers of all elements including hexahedra (H), prisms (Pr), pyramids (Py) and tetrahedra (T). 78

4.20 Comparison between our method and PGP3D method on Switchmec model. The non-hex elements shown in the second row including prisms, pyramids, and tetrahedra are rendered in green, blue, and red, respectively. 79

4.21 Comparison of computation time to generate point set between our method and PGP3D method on Venus model. Three input tetrahedral meshes of different densities are used. Their vertex numbers are given in the legend. For each input tetrahedral mesh, point sets of different sizes are generated. The experiments of PGP3D are done by two machines: Intel(R) Core(TM) i7-6820HQ CPU @ 2.70GHz (the experiments with the two smaller tetrahedral meshes) and Intel(R) Core(TM) i7-6800K CPU @ 3.40GHz (the experiments with the largest tetrehedral mesh). 79

4.22 A comparison between our method and Gao et al.'s method on ASM_Diff027, a CAD model with thin features. The first row shows the result on domain boundary. The second row shows the non-hex elements. In our result, non-hex elements including prisms, pyramids, and tetrahedra are shown in green, blue, and red, respectively. In Gao et al.'s results, non-hex polyhedra with different number of faces are rendered in different colors, including 6-face polyhedra with triangle faces shown in green color. The third row shows the histogram of alignment error measured in degrees. (a) and (b) are comparisons with the same Q_{SJ}. The relationship between Q_{SJ} and $H_{vol}\%$ is shown in (c). 84

4.23 A comparison of our method and Gao et al.'s method on the Elephant Model. The first and second row are the results of our particle method and Gao et al.'s method. The first column shows the result mesh on domain boundary. The second column shows the non-hex elements. In our result, non-hex elements including prisms, pyramids, and tetrahedra are shown in green, blue, and red, respectively. In Gao et al.'s results, non-hex polyhedra with same number of faces are rendered in one color, including 6-face polyhedra with triangle faces shown in green color. The clipping view is shown in the third column. The fourth column shows the histogram of alignment error measured in degrees. The first two rows are comparisons under the same Q_{SJ}. The relationship between Q_{SJ} and$H_{vol}\%$ is shown in the last row. 85

4.24 Comparison of our result (left) and Gao et al.'s result (right) on Front_Upright Model. In our result, non-hex elements including prisms, pyramids, and tetrahedra are shown in green, blue, and red, respectively. In Gao et al.'s results, non-hex polyhedra with different number of faces are rendered in different colors, including 6-face polyhedra with triangle faces shown in green color. 87

4.25 The meshing results before and after mesh refinement on the Skull model. The vertex number is changing from 12837 to 27597 after mesh refinement in the first column. The vertex number is changing from 96504 to 104741 after mesh refinement in the second column. The first row shows the results before mesh refinement. The second row shows the results after mesh refinement. The third row shows the clipping view after mesh refinement. 88

LIST OF TABLES

3.1 Quality statistics of all volume meshing models. Note: best results of each model are in bold font. 33

3.2 Quality statistics of Teddy volume meshes with different vertex numbers. 34

4.1 The mesh quality in comparison with Particle2013 (Zhong et al., 2013). #V and #T are the numbers of vertices and tetrahedra in the result meshes. $\#T_{<20^\circ}$ and $\#T_{<40^\circ}$ are the numbers of tetrahedra with $\theta_{min} < 20^\circ$ and $\theta_{min} < 40^\circ$, respectively. The minimum/maximum, mean, and standard deviation of smallest dihedral angle θ_{min}, largest dihedral angle θ_{max}, edge radius ratio ρ, condition κ are provided. The mean and standard deviation of alignment error ϵ are also listed. Note that the best values are highlighted in bold for each group. 52

4.2 Comparison with CVT and ODT. #V and #T are the numbers of vertices and tetrahedra in the output meshes. The mean value of smallest dihedral angle θ_{min}, largest dihedral angle θ_{max}, edge radius ratio ρ, condition κ, and alignment error ϵ are provided. The computation time is provided, which is the total time including particle optimization and mesh generation. Note that the best values are highlighted in bold for each group. 55

4.3 The quality statistics of rotation alignment experiments on Torus and Fancyring. #V and #T are the numbers of vertices and tetrahedra of the result meshes. $\#T_{<20^\circ}$ and $\#T_{<40^\circ}$ are the numbers of tetrahedra with $\theta_{min} < 20^\circ$ and $\theta_{min} < 40^\circ$, respectively. The minimum, mean, and standard deviation of smallest dihedral angle θ_{min}, edge radius ratio ρ, condition κ are provided. The mean and standard deviation of ϵ are also listed. Dist is the Hausdorff distance between the boundary surfaces of the result and input meshes. Note that the best values are highlighted in bold for each group. 56

4.4 Statistics of mesh quality and time consumption compared with LCT (Fu et al., 2014). #V and #T are the numbers of vertices and tetrahedra of the result meshes. $\#T_{<20^\circ}$ and $\#T_{<40^\circ}$ are the numbers of tetrahedra with $\theta_{min} < 20^\circ$ and $\theta_{min} < 40^\circ$, respectively. The mean of smallest dihedral angle θ_{min}, edge radius ratio ρ, condition κ, and computation time are provided. Note that the best values are highlighted in bold for each group. 58

4.5 Experimental parameters 73

4.6 Statistics and time consumption of models with our particle method and PGP3D method. #Vert is the number of particles in the resulting mesh. $\#Hex\%$ and $H_{vol}\%$ are used to measure the proportion of the hexahedral elements. Non-hex element proportions are measured by $\#Pri\%$, $\#Pyr\%$ and $\#Tet\%$. The shape quality of the hex element is measured by Scaled Jacobian Q_{SJ} in Eq. (4.5). The Hausdorff distance before and after refinement are listed in column dist1 and dist2 respectively. The number before "|" is the Hausdorff distance of the result mesh to the input boundary mesh. The number after "|" is the Hausdorff distance of the input boundary mesh to the result mesh. The time consumption for the particle optimization (Pointset), mesh refinement (Refine) and Tet. to Hex. mesh construction (Tet2Hex) are given. The total time is provided. The time items with * are the total time given in PGP3D paper, which include all the steps. All the times are measured in seconds. Note: n/a denotes that we do not have their results. 83

4.7 Statistics and time consumption of models with our particle method and Gao et al.'s method. Our results are in the gray rows, and Gao et al.'s results are in the white rows. #Vert is the number of particles in the resulting mesh. $H_{vol}\%$ is the volume percentage of hex elements. The quality of hex element is measured by Scaled Jacobian Q_{SJ}. To show the advantage of our method, we adjust Q_{bound} to get the same Q_{SJ} as Gao et al.'s result, while obtaining higher $H_{vol}\%$. $H^*_{vol}\%$ and Q^*_{SJ} are the hex quality of our method when setting $Q_{bound} = 0$. $\overline{\theta_a}$ is the mean alignment error. $\#p$ is the max number of faces in the result hexahedral-dominant mesh. The total computation time is provided in seconds. 86

4.8 The quality statistics and time consumption of meshing before and after mesh refinement on the Skull model. The gray and white rows are the results before and after mesh refinement respectively. dist is the Hausdorff distance. The number before |is the Hausdorff distance from the resulting mesh to the input boundary, the number after |is the Hausdorff distance in the reverse way. 87

CHAPTER 1

INTRODUCTION

Mesh Generation is the process of discretizing a physical domain into a set of connected but non-overlapped elements, in order to solve a numerical solutions for a particle differential or integral equation. Decomposing a geometric domain into high quality subregions, or elements, is one of the biggest challenges in scientific computing. The surface domain can be decomposed to triangles and quads. The volume domain can be subdivided into tetrahedrons, prisms, pyramids, and hexahdrons. Mesh quality is important for particle differential equation (PDE) based simulations. A poor quality element will cause ill-conditioned problems in numerical simulation which may destroy the efficiency and accuracy of the solution. To utilize meshes in numerical applications, some of the following requirements are usually expected.

- The number of the elements are less than a certain number to maintain the scale of the problem.
- The mesh is fine enough in some important regions to gain best approximation in numerical simulation.
- The qualities of the elements are above some threshold to avoid ill-conditioned matrices.
- The mesh well represents the boundary of the geometry.

In some applications, a triangular mesh / tetrahedral mesh may have lower performance than quadrilateral / hexahedral mesh. The later ones usually have less elements with the same number of vertices. Besides that, element alignment is important for some applications.

Based on the current status and requirement in volumetric meshing, this dissertation tackles some remaining issues (e.g., sliver exudation in tetrahedral meshing), raises the directional alignment problem in tetrahedral meshing (i.e. field-aligned tetrahedral meshing), and proposes a new method to optimize the vertices for hexahedral-dominant meshing. All mesh generation methods discussed in this dissertation are variational-based methods, which model the problem in energy optimization framework. The main contributions of this dissertation include the following:

- A novel shape matching energy is proposed to suppress slivers for tetrahedral mesh generation. Given a volumetric domain with a user-specified template (regular) simplex, the tetrahedral meshing problem is transformed into a shape matching formulation with a gradient-based energy, i.e., the gradient of a linear shape function. It effectively inhibits small heights and suppresses all the badly-shaped tetrahedrons in tetrahedral meshes. The proposed approach iteratively optimizes vertex positions and mesh connectivity, and makes the simplices in the computed mesh as close as possible to the template simplex.

- A particle-based frame field alignment framework is proposed to optimize a set of vertices displaying the desired pattern guided by frame field. A *Gaussian Hole Kernel* associated with each particle is constructed. Minimizing the sum of kernels of all particles encourages the particles to form the desired layout.

  - Based on above framework, we raise the frame field alignment problem in tetrahedral meshing and generate field-aligned tetrahedral meshes, guided by cubic lattices, including BCC (Body-Centered Cubic) and FCC (Face-Centered Cubic) lattices. Given a volumetric domain with an input frame field and a user-specified edge length for the cubic lattice, we optimize a set of particles to form the desired lattice pattern, e.g., field-aligned BCC and FCC. The resulting set of particles are connected to yield a high quality field-aligned tetrahedral mesh. As demonstrated by experiments and comparisons, the field-aligned and lattice-guided approach can produce higher quality isotropic and anisotropic tetrahedral meshes than state-of-the-art meshing methods.

- Besides that, we also apply the particle-based variational approach to the generation of hexahedral-dominant meshes. Given a volume domain with an input frame field and user-specified target edge lengths of hexahedral elements, our method generates a hexahedral-dominant mesh aligned with the input frame field. Specifically, the sum of Gaussian hole kernels induces an inter-particle potential energy whose minimization encourages the neighboring particles of each kernel to have a desired hexahedral layout. The definition of the Gaussian hole kernel also allows anisotropic meshing. Upon convergence, the particles form a distinct hexahedral pattern to yield a hexahedral-dominant mesh. Our method is more efficient and produces better-quality meshes than the state-of-the-art hexahedral-dominant meshing algorithms, as demonstrated by extensive experiments and comparisons.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Backgrounds

2.1.1 BCC and FCC Lattices

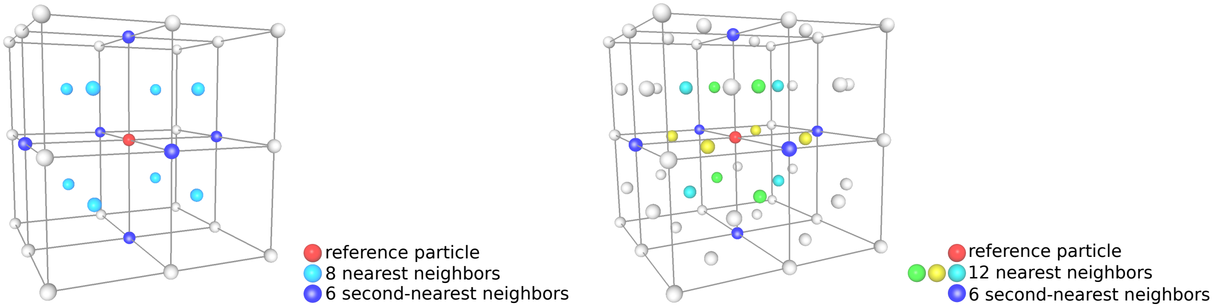
A Body-Centered Cubic (BCC) lattice is formed by vertices of cubic cells along with cell centers as shown in Figure 2.1a. Their Voronoi cells are truncated octahedra, and each dual Delaunay tetrahedral element has dihedral angles [60°(4), 90°(2)], called *BCC tetrahedron*. BCC lattice is the optimal lattice quantizer in terms of the mean squared error (Barnes and Sloane, 1983). Every point in the BCC lattice has identical one-ring neighbor structures, consisting of 8 nearest neighbors and 6 second-nearest neighbors. For a regular cubic lattice with unit edge length, the 14 one-ring neighbors of each vertex consist of the set:

$$\begin{aligned}\mathbf{Onering}_{BCC} =& \{\pm 0.5, \pm 0.5, \pm 0.5\} \cup \\ & \{\{\pm 1, 0, 0\}, \{0, \pm 1, 0\}, \{0, 0, \pm 1\}\}.\end{aligned} \tag{2.1}$$

A Face-Centered Cubic (FCC) lattice consists of vertices of cubic cells and their face centers as shown in Figure 2.1b. Their Voronoi cells are rhombic dodecahedra, and the dual Delaunay tetrahedral elements include two kinds of tetrahedra with dihedral angles [54.735°(4), 90°, 109.47°] and [70.528°(6)]. Comparing to BCC, FCC is preferred as a finite-element mesh generation in terms of better approximation error bounds (Radovitzky and Ortiz, 2000). Each inner vertex in FCC lattice has the same one-ring neighbor structures, denoted as:

$$\begin{aligned}\mathbf{Onering}_{FCC} =& \{\{\pm 0.5, \pm 0.5, 0\}, \{\pm 0.5, 0, \pm 0.5\}, \{0, \pm 0.5, \pm 0.5\}\} \\ & \cup \{\{\pm 1, 0, 0\}, \{0, \pm 1, 0\}, \{0, 0, \pm 1\}\}.\end{aligned} \tag{2.2}$$

The number of one-ring neighbors N_{or} for each vertex is: $N_{or} = 14$ for BCC and $N_{or} = 18$ for FCC.



(a) BCC lattice (b) FCC lattice

Figure 2.1: BCC and FCC lattice

2.1.2 Frame Field and Imagined Space

A frame field specifies the anisotropic behavior of a certain surface or volume domain. In 3D volume meshing, *a discrete frame field* is defined on a tetrahedral mesh by a smoothly-varying set of three vectors, i.e., $\mathbf{T}_i = [\mathbf{t}_{i_1}, \mathbf{t}_{i_2}, \mathbf{t}_{i_3}]$ at each vertex i or each tetrahedron i . These vectors can be non-orthogonal to each other and each vector can have non-unit length. If $\mathbf{t}_{i_1}, \mathbf{t}_{i_2}, \mathbf{t}_{i_3}$ are orthonormal for every i, the frame field reduces to a cross field. The formation of matrix $\mathbf{T}_i$ from these three vectors is not unique since the order of these three vectors can be different and the sign of each vector can be flipped.

The vectors of $\mathbf{T}_i$ define the relative length and alignment of edges starting from point i to form a hexahedral mesh. Suppose $\mathbf{e}_{i1}, \mathbf{e}_{i2}, \mathbf{e}_{i3}$ are the three edges of a hexahedron incident to vertex i. If the hexahedron is perfectly aligned with frame field $\mathbf{T}_i$, then $[\mathbf{e}_{i1}, \mathbf{e}_{i2}, \mathbf{e}_{i3}] = \mathbf{T}_i$. For each matrix $\mathbf{T}_i$, there is a corresponding 3×3 matrix $\mathbf{B}_i$ called *anisotropic metric*, defined by $\mathbf{B}_i = \mathbf{T}_i^{-1}$. If we transform edges $\mathbf{e}_{ij}$ with $\mathbf{B}_i$, then we get $\mathbf{B}_i[\mathbf{e}_{i1}, \mathbf{e}_{i2}, \mathbf{e}_{i3}] = \mathbf{I}$, i.e., the hexahedron aligned with frame field $\mathbf{T}_i$ is transformed to a hexahedron aligned with the axes of a 3D Cartesian coordinate system. Matrix $\mathbf{B}_i$ maps a vector from the original space into a new *imagined space*, in which the hexahedral elements are aligned with the three coordinate axes.

We suppose that the frame field is given as an input. Several existing state-of-the-art algorithms can be used to generate a high-quality cross field for a volumetric domain (Huang et al., 2011; Ray et al., 2016; Gao et al., 2017; Solomon et al., 2017). There are also user-designed frame fields for specific applications. We will present meshes using both user-designed frame fields and discrete cross fields. The smoothness and the number of singularities of the input frame field do affect the final output mesh quality, e.g. the quality of alignment to BCC and FCC lattice and the percentage of hexahedral elements. We focus on generating a particle pattern that locally aligns with the given frame field. Even if the input frame field is not sufficiently smooth, our method still robustly generates particles with desired pattern, which lead to high-quality meshes.

2.2 Related Work on Tetrahedral Meshing

Tetrahedral mesh generation has been studied for several decades (Owen, 1998) in both engineering and computer sciences fields.The algorithms can be categorized into four types including advancing front methods (Ito et al., 2004; Schöberl, 1997; Li et al., 2000), octree-based and lattice-based methods (Labelle and Shewchuk, 2007; Neil Molino and Fedkiw, 2003), Delaunay-based tetrahedral methods (Cheng et al., 2012), Poisson-disk sampling methods (Guo et al., 2016) and particle-based methods.

Advancing Front Methods (Möller and Hansbo, 1995) start from the domain boundary and gradually add vertices and tetrahedra until the domain is completely meshed. They preserve the domain boundary explicitly. However, the difficulty of this type of method is to resolve the intersected tetrahedra inside the domain.

Octree and Lattice Methods: Quadtree/Octree is a Cartesian grid structure in 2D/3D. Quadtree encoding an input curve in 2D was first introduced by Yerry and Shephard (Yerry and Shephard, 1983), then generalized to 3D (Yerry and Shephard, 1984) and refined by Shephard and Georges (Shephard and Georges, 1991). These methods start from encoding the surface as an adaptive grid structure, i.e., an octree, which converts the surface to a volumetric representation. The tetrahedral meshes are then constructed by local tetrahedralization of each octree cell, with special treatment for the cells intersecting the input surface (Shephard and Georges, 1991; Mitchell and Vavasis, 1992).

Similar to octree methods, lattice methods utilize a space-filling tetrahedral lattice instead of Cartesian grid as volume representation, which omits the local tetrahedralization for the interior cells. The boundary tetrahedra are deformed to preserve smooth boundary. For a better quality, finer cells are generated by Fredenthal subdivision of a grid (Velho et al., 1997) or adaptive BCC lattice (Molino et al., 2003) along the surface boundary. The isosurface stuffing method (Labelle and Shewchuk, 2007) proposed an option to cut the boundary tetrahedra to resolve the input surface. The cutting rules yield theoretical bounds for the smallest/largest dihedral angles. Doran et al. (Doran et al., 2013) extended the method with A15 lattice. However, these stuffing-based methods cannot generate either field-aligned isotropic tetrahedral meshes, or anisotropic tetrahedral meshes.

Delaunay-Based Methods (Cheng et al., 2012) can be further categorized into two groups: (1) Delaunay refinement-based methods (Chew, 1997; Jamin et al., 2015; Si, 2015) improve the mesh quality by inserting new vertices until certain user-specified conditions are met, e.g., the minimal dihedral angle. (2) Variational methods iteratively minimize an energy, e.g., CVT (Du and Wang, 2003; Alliez et al., 2005; Liu et al., 2009), ODT (Chen and Holst, 2011; Chen et al., 2014), by optimizing positions of vertices and their connectivities.

Particle-Based Methods use repulsive particles to resample surfaces or volumes. It was first introduced by Turk (Turk, 1992), and later extended by Witkin and Heckbert (Witkin and Heckbert, 2005) for implicit surface meshing. They introduced Gaussian kernel to model the interaction between particles which sample an implicit surface. Researchers have tried different choices of kernels, such as a modified cotangent function with finite support (Meyer et al., 2005), or a bounded cubic function (Yamakawa and Shimada, 2000), and packing ellipsoidal bubbles instead of spherical bubbles to get anisotropic tetrahedral meshes. Zhong et al. (Zhong et al., 2013) used the Gaussian kernel to model the inter-particle energy in an embedding space to solve anisotropic surface meshing. Note that the traditional Gaussian kernel is radially-symmetric. Even though it can be distorted to elliptically-symmetric under Riemannian metric, the interaction between particles still resembles packing of circles/spheres isotropically, or ellipses/ellipsoids anisotropically. This makes it impossible to explicitly control field-alignment of particles. We propose *Gaussian Hole Kernel* as potential energy between particles to guide their distribution into either BCC or FCC pattern, which will be introduced in the next section.

2.2.1 Silver by Variational-method

CVT-based methods (Du and Wang, 2003, 2005a) calculate the dual meshes of Voronoi cells, instead of computing the shape of tetrahedrons, which may generate a large number of slivers in 3D tetrahedral mesh (Yan et al., 2010). ODT-based methods (Alliez et al., 2005; Chen et al., 2014) perform better in terms of suppressing slivers comparing to CVT-based methods, However, it still cannot completely avoid slivers. Comparing to CVT and ODT methods, the recent Particle-based mesh optimization method (Zhong et al., 2013) does not require to optimize the mesh connectivity during the optimization. This property makes it easy and fast to converge, but the lack of considering mesh structure leads to many slivers in the final tetrahedral mesh. It is noted that all of three variational-based optimization methods cannot eliminate badly-shaped elements (e.g., slivers) completely, so a post process for improving the mesh quality is necessary, such as sliver exudation (Cheng et al., 2000), aggressive improvement (Klingner and Shewchuk, 2007), vertex perturbation (Tournois et al., 2009).

2.3 Hexahedral Meshing and Hexahedral-dominant Meshing

Hexahedral volume meshing has been studied for several decades and there are some surveys (Owen, 1998; Shimada, 2006; Shepherd and Johnson, 2008).

All-hex Remeshing Several semi-automatic strategies were proposed, such as multiple sweeping (Shepherd et al., 2000), paving and plastering (Staten et al., 2005), requiring the user to decompose the model for suitable mappings. There are some automatic methods, such as decomposition of the volume by Voronoi graph (Sheffer et al., 1999), constrained hexahedral meshing by using Geode-Template (Carbonera and Shepherd, 2010). Generally, these methods are very time-consuming, and complicated to generate high-quality and boundary-conforming hexahedral meshes for complex shapes, since the partition and sizing are crucial to the mesh quality.

Inspired by recent development of quadrangulation techniques, a lot of all-hex remeshing approaches (Nieser et al., 2011) are proposed using multi-chart parameterization. The chart layout of the manifold are derived from an automatically generated frame field (Huang et al., 2011; Sokolov et al., 2016). Unfortunately, such frame fields have singularities in most of the cases, which makes the parameterization degenerate and leads to missing elements even using a specially-designed robust hex element extractor (Lyon et al., 2016). Although the local conflicts can be resolved (Li et al., 2012; Jiang et al., 2014), addressing the global ones remains a big challenge.

As a remedy, one can simplify the topological structure by restricting the hex mesh to be grid-like (Maréchal, 2009). Methods using polycube structure is indeed a warped grid (Tarini et al., 2004; Han et al., 2010; Gregson et al., 2011; Xu et al., 2017), but still lacks enough flexibility to handle complex models. Although recent closed-form polycube method (Fang et al., 2016) extends such a topology into internal singularity-free structure, these methods may still introduce large distortion and mis-matched features on boundary surfaces.

Hex-dominant Remeshing Considering the difficulty of automatically generating high quality all-hex mesh, hex-dominant remeshing methods have been proposed (Yamakawa and Shimada, 2003). Recent techniques tried to utilize the development of parameterization-based methods. PGP3D (Sokolov et al., 2016) extends Periodic Global Parameterization (Ray et al., 2006) to generate the point set. The Eulerian style discretization just encodes the fractional parts of the parametrization coordinates at the vertices of the input tetrahedral mesh, which has weaker topological constraints between the hex nodes. The blindness of the integer part in PGP3D may introduce many inproper hex nodes according to the frame field. PGP3D follows the three-step pipeline as mentioned before, and adds two extra steps to further improve the mesh results. One step is curl-correction which is used to add some corrections to edge vector map which indirectly improves the cross field by reducing the number of singularities. The other step is mesh refinement, which inserts extra points to the surface to reduce the Hausdorff distance between the resulting domain and the input domain. PGP3D method may fail due to the singularities in the frame field (Li et al., 2012; Jiang et al., 2014). The major difference between PGP3D and our method is that we develop a different method to optimize vertex positions to locally align with the given frame field. Gao et al's method (Gao et al., 2017) is another state-of-the-art method in generating hexahedral-dominant meshing with high amount of isotropy. It proposes a robust and automatic field-guided polyhedral agglomeration algorithm to generate hexahedral dominant mesh with a few irregular polyhedra from the input tetrahedral mesh. Quaternion representation is introduced in the field generation phase, which leads to a more efficient orientation field optimization. Besides that, it proposes a robust extraction algorithm which works with both local parameterization and global parameterization. The non-hex elements in their result meshes are any polyhedra with k faces of either triangle or quadrilateral. We show in Sec. 4.4.3 that our new particle-based method produces better quality hexahedral-dominant meshes than both PGP3D method and Gao et al.'s method.

Another group of hexahedral-dominate meshing is the front-propagation-based approaches (Owen and Saigal, 2000; Baudouin et al., 2014; Botella et al., 2016). The H-Morph method (Owen and Saigal, 2000) starts with an initial tetrahedral mesh and systematically transforms and combines tetrahedra into hexahedra. It uses an advancing front technique where the initial front consists of a set of prescribed quadrilateral surface facets. (Baudouin et al., 2014) is similar to the advancing front method. The vertices are created layer-by-layer toward the center of the geometry, and the hexahedra are built at the very end. However, the limitation of the front-propagation-based methods is that they do not globally optimize the mesh vertex positions altogether.

Variational and Particle-based Method Variational methods define an objective function of vertex positions as well as connectivities to optimize the mesh. Variational methods usually perform effectively and robustly in mesh generation, since they optimize the mesh elements (vertex positions or connectivities) globally. Variational methods in both isotropic and anisotropic triangular / tetrahedral mesh generation were well investigated, e.g., Centroidal Voronoi Tessellation (CVT) (Du et al., 1999), Optimal Delaunay Triangulation (ODT) (Chen and Xu, 2004), and particle-based method (Zhong et al., 2013). All of them can effectively optimize the mesh through different objective functions. However, variational methods in hexahedral meshing are much more difficult to design. For instance, L_p-CVT (Lévy and Liu, 2010) is a variational method to generate hexahedral-dominant meshes based on the higher-order moment of the vertex coordinates on Voronoi cells. Particle-based variational method works well in both triangular and tetrahedral meshing (Zhong et al., 2013). Compared with L_p-CVT and other variational methods, one of the main advantages of particle-based method is that it only needs to iteratively update the vertex positions based on the defined inter-particle energy and force in their local neighborhoods, instead of computing Voronoi cells or connectivities, which is more efficient especially in anisotropic cases. The inter-particle energy function plays a vital role in particle-based method. Gaussian kernel has nice performance in triangular and tetrahedral meshing (Zhong et al., 2013). However, it is not suitable for the hexahedral meshing problem. We propose a new Gaussian hole kernel to simulate the inter-particle energy and force for the hexahedral meshing, and we show in Sec. 4.4.3 that our new method produces better quality hexahedral-dominant meshes than L_p-CVT.

CHAPTER 3

SLIVER-SUPPRESSING TETRAHEDRAL MESHING[1]

3.1 Overview

In mesh generation, mesh quality highly depends on the size and shape of each element. There are several measurements for tetrahedral mesh quality, and dihedral angle is one of the most important criteria (Alliez et al., 2005; Guo et al., 2016), since badly-shaped tetrahedrons with tiny dihedral angles (i.e., sliver) can severely affect numerical simulation (Shewchuk, 2002b).

Essentially, simplex meshes are used to form a piecewise linear approximation of function $u(\mathbf{x})$ to represent the given shapes. There are several ways to describe the approximation error. Optimal Delaunay Triangulation (ODT) (Chen and Xu, 2004) was proposed to minimize the L^p norm of the difference over the domain Ω between the target function $\hat{u}(\mathbf{x})$ and the interpolated function $u(\mathbf{x})$: $E(\mathbf{x}) = \int_\Omega \|\hat{u}(\mathbf{x}) - u(\mathbf{x})\|_{L^p}\,\mathrm{d}\mathbf{x}$. Interpolation error is an important mesh quality measurement. However, the definition of ODT determines that it cannot avoid sliver in tetrahedral mesh. A sliver with close to zero volume still has small interpolation error. So minimizing interpolation error cannot avoid sliver.

Consider the L^p norm of the difference between the gradient of the target function $\triangledown\hat{u}(\mathbf{x})$ and the gradient of the interpolated function $\triangledown u(\mathbf{x})$: $E(\mathbf{x}) = \int_\Omega \|\triangledown\hat{u}(\mathbf{x}) - \triangledown u(\mathbf{x})\|_{L^p}\,\mathrm{d}\mathbf{x}$, the gradient error can be strongly affected by the shape of the elements as well as their sizes. Once one dihedral angle approaches either 0° or 180° in the tetrahedral mesh, the gradient error will grow dramatically large. We propose a shape matching framework and design a gradient-based energy (i.e., the gradient of linear shape function), which heavily punish slivers in tetrahedral meshes. By specifying a template simplex, e.g. a regular simplex, the

---

[1] ©2017 Elsevier, Reprinted, with permission, from Saifeng Ni, Zichun Zhong, Yang Liu, Wenping Wang, Zhonggui Chen, “Sliver-suppressing Tetrahedral Mesh Optimization with Gradient-Based Shape Matching Energy”, in Computer Aided Geometric Design (Proceedings of GMP 2017), Vol.52-53, pp. 247-261, 2017.

idea of the proposed method is to make the shape of the to-be-optimized simplex as close as possible to the shape of the template simplex. The experiment results show that our proposed energy has high effectiveness in sliver suppression compared with state-of-the-art methods in the tetrahedral meshing.

3.2 Shape Matching Triangulation Energy

Our mesh optimization is illustrated in an algebraic framework, and we call it *shape matching*. Given a template d-simplex $\hat{\tau}_d$, our target is to make any d-simplex τ_d in the mesh as similar as possible to $\hat{\tau}_d$. Here "similar" means the same shape as well as the same sizing factor comparing with the template simplex. If the template is set as a regular d-simplex, any d-simplex in the mesh is expected to be a regular simplex with a constant sizing factor conforming to the defined template simplex, i.e., all simplices in the mesh are endowed with the same shape and the same size.

The shape matching idea is straightforward, but the tricky part is how to well represent the difference between the to-be-optimized simplex τ_d and the template simplex $\hat{\tau}_d$. As the difference is being minimized, the to-be-optimized simplex will become closer and closer to the template simplex. A good difference representation should be scale-sensitive, orientation-free, and also well encoding the shape information. Mathematically, a d-simplex embedded in d-dimension can be defined by a $d \times d$ matrix. Once the template simplex and the to-be-optimized simplex are both represented by $d \times d$ matrices, an affine mapping can be utilized to build the relationship between those two simplices with another $d \times d$ matrix, i.e., the Jacobian between those two simplices. If the affine mapping is an identity matrix, the to-be-optimized simplex is exactly equal to the template simplex. If the affine mapping is a rotation matrix, the to-be-optimized simplex is also equal to the template simplex. In this paper, the squared Frobenius norm of the affine transformation matrix is used to measure the difference between the to-be-optimized simplex and the template simplex, since

Frobenius norm is invariant under rotations and also it keeps the sizing information. Then by minimizing the summation of all differences over the entire mesh, the optimal solution of isotropic mesh will be reached when all to-be-optimized simplices are regular and of the same size.

The shape matching framework can be generalized for any d-simplex mesh. A 0-simplex is a vertex $\mathbf{v}_i$; a 1-simplex is an edge $\mathbf{e}_{ij} = \mathbf{v}_j - \mathbf{v}_i, 0 \leq i < j \leq d$; a 2-simplex is a triangle; and a 3-simplex is a tetrahedron, etc. In addition, the shape matching framework can be easily extended to solve adaptive and anisotropic meshing problems. By mapping the to-be-optimized simplex from Riemannian metric space to Euclidean space, then we still use the regular simplex as template. The computations after that are the same as the uniform isotropic case. The shape matching framework is extendable and flexible, and it provides freedom to design different mesh element shapes based on different applications. Both isotropic and anisotropic meshing are demonstrated in the experiment section.

In this paper, our target is to remove badly-shaped elements, e.g., slivers, in the isotropic tetrahedral mesh. We will interpret our energy under the proposed shape matching framework. Compared with the traditional *Edge-based Shape Matching (ESM)* in Sec. 3.2.1, the gradient of the linear shape function is used as basis to represent a simplex, and we called it *Gradient-based Shape Matching (GSM)* in Sec. 3.2.2.

3.2.1 Edge-Based Shape Matching (ESM)

The traditional methods (Knupp, 2001) used d edge vectors from one vertex as column vectors which form a $d \times d$ matrix $\mathbf{T}_d$ to represent a d-simplex τ_d. Given a matrix $\mathbf{T}_d$, the corresponding d-simplex is uniquely defined. Suppose $\hat{\mathbf{T}}_d$ represents the template d-simplex $\hat{\tau}_d$, then the affine transformation $\mathbf{J}_d$ satisfies $\mathbf{J}_d = \mathbf{T}_d\hat{\mathbf{T}}_d^{-1}$. Squared Frobenius norm of this affine transformation $\|\mathbf{J}_d\|_F^2$ is utilized to measure the difference between $\hat{\tau}_d$ and τ_d. This difference measurement is called shape matching energy E_{τ_d}. Summing up shape matching energies of all simplices, we obtain the energy for the entire d-simplex mesh $E_d = \sum_{\tau_d \in \mathscr{T}} E_{\tau_d}$, where $\mathscr{T}$ is the set of d-simplices in the mesh.

The shape matching framework works for any d-simplex. For the simplicity of illustration, we demonstrate the basic idea using a 2-simplex (i.e., a triangle) in Figure 3.1.

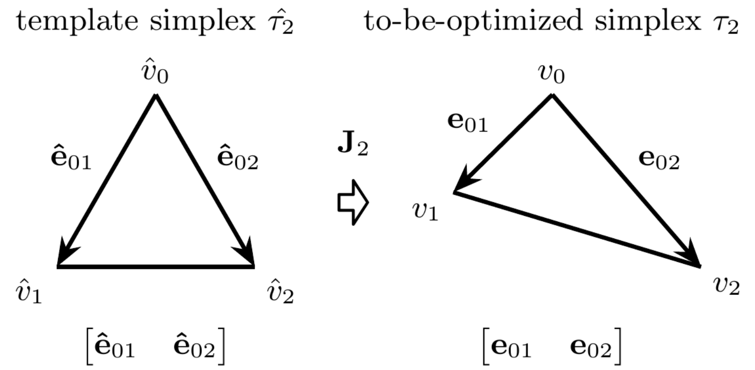

template simplex $\hat{\tau}_2$
to-be-optimized simplex τ_2
$\hat{v}_0$
$\hat{\mathbf{e}}_{01}$
$\hat{\mathbf{e}}_{02}$
$\hat{v}_1$
$\hat{v}_2$
$\mathbf{J}_2$
v_0
$\mathbf{e}_{01}$
$\mathbf{e}_{02}$
v_1
v_2
$\begin{bmatrix}\hat{\mathbf{e}}_{01} & \hat{\mathbf{e}}_{02}\end{bmatrix}$
$\begin{bmatrix}\mathbf{e}_{01} & \mathbf{e}_{02}\end{bmatrix}$

Figure 3.1: The illustration of edge-based shape matching for a 2-simplex.

Any 2-simplex τ_2 is represented as a matrix formed by two edge vectors $\mathbf{T}_2 = \begin{bmatrix}\mathbf{e}_{01} & \mathbf{e}_{02}\end{bmatrix}$, where $\mathbf{e}_{01} = \mathbf{v}_1 - \mathbf{v}_0$ and $\mathbf{e}_{02} = \mathbf{v}_2 - \mathbf{v}_0$. The determinant of $\mathbf{T}_2$ is proportional to the triangle area $|\tau_2|$. So matrix $\mathbf{T}_2$ encodes the shape as well as the size of the triangle.

The affine transformation $\mathbf{J}_2$ between $\hat{\tau}_2$ and τ_2 is expressed as:

$$\mathbf{J_2} \begin{bmatrix}\hat{\mathbf{e}}_{01} & \hat{\mathbf{e}}_{02}\end{bmatrix} = \begin{bmatrix}\mathbf{e}_{01} & \mathbf{e}_{02}\end{bmatrix}. \tag{3.1}$$

If the template $\hat{\tau}_2$ is a regular triangle with edge length $\hat{a}$, the shape matching energy $E^{esm}_{\tau_2}$ of a simplex τ_2 can be simplified as

$$E^{esm}_{\tau_2} = \|\mathbf{J}_2\|_F^2 = trace(\mathbf{J}_2^T \mathbf{J}_2) = \frac{2}{3\hat{a}^2} \sum_{0 \le i < j \le 2} \mathbf{e}_{ij}^T \mathbf{e}_{ij}. \tag{3.2}$$

When $d = 3$, the shape matching energy $E^{esm}_{\tau_3}$ from a simplex τ_3 to a regular tetrahedron $\hat{\tau}_3$ is represented by the affine transformation $\mathbf{J}_3$ as

$$E^{esm}_{\tau_3} = trace\left(\mathbf{J}_3^T \mathbf{J}_3\right) = \frac{1}{2\hat{a}^2} \sum_{0 \le i < j \le 3} \mathbf{e}_{ij}^T \mathbf{e}_{ij}, \tag{3.3}$$

where $\hat{a}$ is the edge length of $\hat{\tau}_3$.

The above definitions (2-simplex and 3-simplex) are based on edge vectors, so we call them Edge-based Shape Matching (ESM). The main disadvantage of ESM is that it cannot avoid slivers in tetrahedral mesh. The key reason is that slivers may have large edge lengths but close-to-zero heights. Based on this observation, Gradient-based Shape Matching (GSM) energy is proposed, which can be used to effectively suppress slivers.

3.2.2 Gradient-Based Shape Matching (GSM)

For a d-simplex τ_d, barycentric coordinate ω_i corresponding to each vertex i is used as the linear shape function. Any point v inside the simplex satisfies $\mathbf{v} = \sum_{i=0}^{d} \omega_i \mathbf{v}_i$, where $\sum_{i=0}^{d} \omega_i = 1$. If we use the mesh to approximate certain data function $u(\mathbf{x})$, then the data function value at the vertex is given. Suppose the data value at vertex $\mathbf{v}_i$ of τ_d is $u(\mathbf{v}_i)$, which is a constant. Then the data function value at $\mathbf{v}$ is $u(\mathbf{v}) = \sum_{i=0}^{d} \omega_i u(\mathbf{v}_i)$. The gradient of the data function can be written as $\bigtriangledown u(\mathbf{v}) = \sum_{i=0}^{d} \bigtriangledown \omega_i u(\mathbf{v}_i)$. $\bigtriangledown \omega_i$ corresponding to vertex $\mathbf{v}_i$ of a d-simplex is a constant vector. The direction of $\bigtriangledown \omega_i$ is pointing perpendicularly from the opposite $(d-1)$-simplex S_i (as the base) to the vertex v_i. The length of $\bigtriangledown \omega_i$ is equal to the inverse of height h_i, i.e., $|\bigtriangledown \omega_i| = \frac{1}{|h_i|}$ in 2-simplex case, and $|\bigtriangledown \omega_i| = \frac{|S_i|}{|\tau_d|}$ in 3-simplex case, where $|\tau_d|$ is the volume of the simplex τ_d and $|S_i|$ is the area of face S_i. Both its direction and length encode the shape information. Besides that $\bigtriangledown \omega_i, 0 \leq i < d$ are linearly independent for any non-degenerate simplex. So using the gradients of linear shape functions as the bases to represent simplex well describe the shape as well as size difference bewteen a badly-shaped simplex and a regular simplex.

For the simplicity of illustration, we still use a 2-simplex in Figure 3.2 to demonstrate the basic idea of GSM.

Any 2-simplex τ_2 is represented by its gradient matrix $\begin{bmatrix} \bigtriangledown \omega_0 & \bigtriangledown \omega_1 \end{bmatrix}$. Once $\bigtriangledown \omega_0$ and $\bigtriangledown \omega_1$ are given, $\mathbf{e}_{20}, \mathbf{e}_{21}$ is uniquely defined, so triangle is uniquely determined by $\bigtriangledown \omega_0$ and $\bigtriangledown \omega_1$.

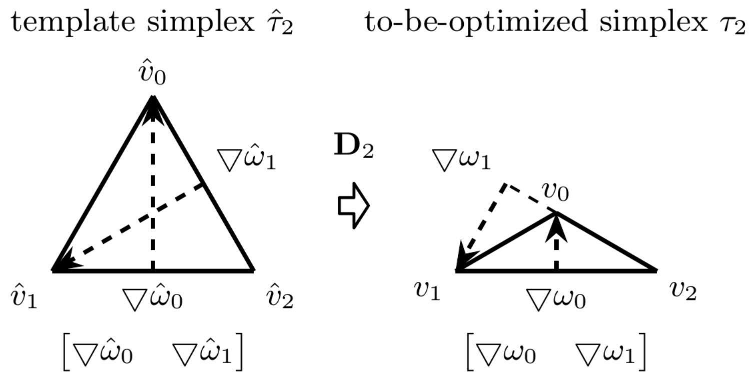

template simplex $\hat{\tau}_2$
to-be-optimized simplex τ_2
$\hat{v}_0$
$\triangledown\hat{\omega}_1$
$\mathbf{D}_2$
$\triangledown\omega_1$
v_0
$\hat{v}_1$
$\triangledown\hat{\omega}_0$
$\hat{v}_2$
v_1
$\triangledown\omega_0$
v_2
$\begin{bmatrix}\triangledown\hat{\omega}_0 & \triangledown\hat{\omega}_1\end{bmatrix}$
$\begin{bmatrix}\triangledown\omega_0 & \triangledown\omega_1\end{bmatrix}$

Figure 3.2: The illustration of gradient-based shape matching for a 2-simplex.

Affine transformation $\mathbf{D}_2$ between $\hat{\tau}_2$ and τ_2 can be expressed as:

$$\mathbf{D}_2 \begin{bmatrix}\triangledown\hat{\omega}_0 & \triangledown\hat{\omega}_1\end{bmatrix} = \begin{bmatrix}\triangledown\omega_0 & \triangledown\omega_1\end{bmatrix}. \tag{3.4}$$

Then, GSM energy between the template $\hat{\tau}_2$ and any 2-simplex τ_2 is defined as

$$E_{\tau_2}^{gsm} = trace(\mathbf{D}_2^T \mathbf{D}_2). \tag{3.5}$$

Inherited from Frobenius norm properties, GSM energy is also scale-sensitive and orientation-free. By expanding Eq. (3.5) with the height definition and Eq. (3.4), GSM energy is simplified as:

$$E_{\tau_2}^{gsm} = \frac{\hat{a}^2}{8} \frac{\sum_{0 \le i < j \le 2} \mathbf{e}_{ij}^T \mathbf{e}_{ij}}{|\tau_2|^2} = \frac{\hat{a}^2}{2} \sum_{i=0}^{2} \frac{1}{|h_i|^2}, \tag{3.6}$$

where $\hat{a}$ is the edge length of the regular template $\hat{\tau}_2$, $|\tau_2|$ is the area of simplex τ_2. For each 2-simplex, GSM energy is the summation of inverse of squared heights. When there is one internal dihedral angle approaching 0°or 180°, one or more heights will be also close to 0. Thus minimizing GSM energy inhibits small heights and also uneven height. So it suppresses all badly-shaped simplices. The minimal energy is reached when all heights are equal to each other, i.e., an equilateral 2-simplex. Besides that, the energy for each simplex also encodes its size information (height h_i). So when minimizing the total energy, the result mesh will converge to the optimal solution.

Tetrahedralization $d = 3$

Tetrahedral mesh optimization is more challenging than triangular mesh optimization, especially to remove slivers completely. Figure 3.3 shows several typical badly-shaped tetrahedrons (Freitag and Knupp, 2002). Most of them either have one or more dihedral angles approaching 0° / 180° or have uneven heights. Traditional ESM energy cannot avoid slivers as discussed in Sec. 3.2.1, while the proposed GSM energy significantly suppress all of them.

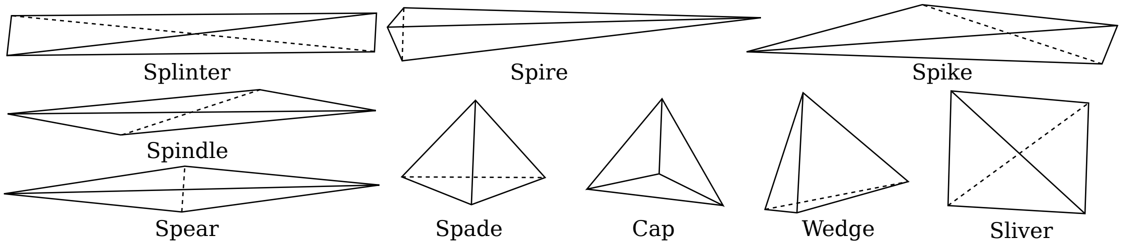


Figure 3.3: Several typical badly-shaped tetrahedrons (Freitag and Knupp, 2002).

Affine transformation $\mathbf{D}_3$ between $\hat{\tau}_3$ and τ_3 is represented as:

$$\mathbf{D}_3 \begin{bmatrix} \triangledown\hat{\omega}_0 & \triangledown\hat{\omega}_1 & \triangledown\hat{\omega}_2 \end{bmatrix} = \begin{bmatrix} \triangledown\omega_0 & \triangledown\omega_1 & \triangledown\omega_2 \end{bmatrix}. \tag{3.7}$$

For any 3-simplex, $\triangledown\omega_i$ has the same direction as the normal of face S_i, where S_i is the face opposite to vertex $\mathbf{v}_i$. If the template tetrahedron is regular and with edge length $\hat{a}$, then GSM energy $E^{gsm}_{\tau_3}$ between $\hat{\tau}_3$ and τ_3 can be simplified as:

$$E^{gsm}_{\tau_3} = trace\left(\mathbf{D}_3^T\mathbf{D}_3\right) = \frac{\hat{a}^2}{18}\frac{\sum_{i=0}^{3}|S_i|^2}{|\tau_3|^2} = \frac{\hat{a}^2}{2}\sum_{i=0}^{3}\frac{1}{|h_i|^2}, \tag{3.8}$$

where $|S_i|$ is the area of face S_i, $|\tau_3|$ is the volume of simplex τ_3.

GSM energy of a tetrahedron is the summation of the inverse of the squared heights. The optimal solution is to have the same heights, i.e. regular tetrahedron. The total GSM energy for the entire tetrahedral mesh is:

$$E^{gsm}_{total} = \frac{\hat{a}^2}{18}\frac{1}{|\mathscr{T}|}\sum_{\tau_3\in\mathscr{T}}\frac{\sum_{i=0}^{3}|S_i|^2}{|\tau_3|^2}, \tag{3.9}$$

where $\mathscr{T}$ is the set of tetrahedrons in the volume mesh and $|\mathscr{T}|$ is the total number of tetrahedrons in the volume mesh. When minimizing the GSM energy, the optimal solution is to have all the heights to be the same. GSM energy has great punishment on a small height. So minimizing the energy effectively suppressing all the badly-shaped tetrahedrons.

Curve Discretization $d = 1$

A curve can be discretized to be a set of 1-simplices. Inheriting from the idea of GSM, the mapping between the template $\hat{\tau}_1$ and the to-be-optimized τ_1 is the ratio between their lengths $\mathbf{D}_1 = \frac{|\hat{\tau}_1|}{|\tau_1|}$. So GSM energy function of the line segment τ_1 is:

$$E_{\tau_1}^{gsm} = \frac{|\hat{\tau}_1|^2}{|\tau_1|^2}. \tag{3.10}$$

GSM energy of the entire curve is:

$$E_{\tau_1} = \frac{|\hat{\tau}_1|^2}{|\mathscr{L}|} \sum_{\tau_1 \in \mathscr{L}} \frac{1}{|\tau_1|^2}, \tag{3.11}$$

where $\mathscr{L}$ is the set of 1-simplices and $|\mathscr{L}|$ is the total number of 1-simplices in the corresponding curve $\mathscr{L}$.

3.2.3 The Importance of GSM Energy in Tetrahedral Meshing

The difference between ESM and GSM lies in the representation of a simplex. ESM uses edge vectors, while GSM utilizes the gradients of linear shape functions. When $d = 3$, we have $\triangledown\omega_0 = \frac{\mathbf{e}_{12}\times\mathbf{e}_{13}}{6|\tau_3|}$, $\triangledown\omega_1 = \frac{\mathbf{e}_{02}\times\mathbf{e}_{03}}{6|\tau_3|}$, $\triangledown\omega_2 = \frac{\mathbf{e}_{03}\times\mathbf{e}_{01}}{6|\tau_3|}$, and $\triangledown\omega_1 \times \triangledown\omega_2 = \frac{\mathbf{e}_{01}}{6|\tau_3|}$. Via the cross product property in matrix transformation, the affine transformation $\mathbf{J}_3$ in ESM satisfies:

$$(\mathbf{J}_3\mathbf{e}_{ij}) \times (\mathbf{J}_3\mathbf{e}_{ik}) = |\mathbf{J}_3|\, \mathbf{J}_3^{-T}\mathbf{e}_{ij} \times \mathbf{e}_{ik}, \tag{3.12}$$

where $|\mathbf{J}_3|$ is the determinant of matrix $\mathbf{J}_3$.

Then the relation between affine transformations of ESM and GSM can be expressed as

$$\begin{aligned}\mathbf{D}_3 = &\begin{bmatrix} \bigtriangledown\omega_0 & \bigtriangledown\omega_1 & \bigtriangledown\omega_2 \end{bmatrix} \begin{bmatrix} \bigtriangledown\hat{\omega}_0 & \bigtriangledown\hat{\omega}_1 & \bigtriangledown\hat{\omega}_2 \end{bmatrix}^{-1} \\ =&\frac{|\mathbf{J}_3|\mathbf{J}_3^{-T}|\hat{\tau}_3|}{|\tau_3|} \begin{bmatrix} \hat{\mathbf{e}}_{12} \times \hat{\mathbf{e}}_{13} & \hat{\mathbf{e}}_{02} \times \hat{\mathbf{e}}_{03} & \hat{\mathbf{e}}_{03} \times \hat{\mathbf{e}}_{01} \end{bmatrix} \\ &\begin{bmatrix} \hat{\mathbf{e}}_{12} \times \hat{\mathbf{e}}_{13} & \hat{\mathbf{e}}_{02} \times \hat{\mathbf{e}}_{03} & \hat{\mathbf{e}}_{03} \times \hat{\mathbf{e}}_{01} \end{bmatrix}^{-1} \\ =&\mathbf{J}_3^{-T}. \end{aligned} \tag{3.13}$$

Suppose the eigenvalues of $\mathbf{J}_3$ are $\lambda_1, \lambda_2, \lambda_3$, then ESM energy of one tetrahedron is

$$E_{\tau_3}^{esm} = trace\left(\mathbf{J}_3^T\mathbf{J}_3\right) = \lambda_1^2 + \lambda_2^2 + \lambda_3^2, \tag{3.14}$$

while GSM energy of the tetrahedron is

$$E_{\tau_3}^{gsm} = trace\left(\mathbf{J}_3^{-1}\mathbf{J}_3^{-T}\right) = \frac{1}{\lambda_1^2} + \frac{1}{\lambda_2^2} + \frac{1}{\lambda_3^2}. \tag{3.15}$$

Although both ESM and GSM reach their minimums when $\lambda_1 = \lambda_2 = \lambda_3$, GSM is more sensitive to a small λ_i.

3.3 GSM Energy Optimization

The proposed GSM energy optimization is to build a high-quality regular d-simplex mesh in a domain Ω_d based on a given regular template and the user-specified number of vertices N. Both the positions of N vertices and their connectivities are required to be optimized. Our mesh optimization process involves two operations: the numerical nonlinear optimization for vertex smoothing and the combinatorial optimization for connectivity update. These two operations are carried out iteratively to minimize the proposed GSM energy.

3.3.1 Vertex Smoothing

With fixed mesh connectivity, the optimization computation reduces to a nonlinear numerical optimization problem. The connectivity is fixed and inverted simplices should be avoided during the vertex smoothing, so the possible solution of a vertex should be inside its one-ring domain. Newton's method converges quickly in the local region, so it is choosen to do vertex smoothing.

The vertex updating rule of Newton's method is:

$$\mathbf{v}^* = \mathbf{v} - \Delta\mathbf{v} = \mathbf{v} - \alpha\mathbf{h}^{-1}\mathbf{g}, \tag{3.16}$$

where $\mathbf{h}$ and $\mathbf{g}$ are the Hessian and gradient at vertex position $\mathbf{v}$, $\mathbf{v}^*$ is the updated position of vertex $\mathbf{v}$, and α is the step size. The backtracking line search is utilized to determine the step size.

In the following, we will discuss how to calculate the gradient and Hessian of GSM energy. GSM energy in any d-simplex has the similar formulation, which can be written as:

$$E_{\tau_d}^{gsm} = c_d \frac{p(\mathbf{v})}{q(\mathbf{v})}, \tag{3.17}$$

where c_d is a constant. When $d = 1$, $p(\mathbf{v})$ is 1, $q(\mathbf{v})$ is the squared length of the simplex τ_1; when $d = 2$, $p(\mathbf{v})$ is the summation of the squared edge lengths, $q(\mathbf{v})$ is the squared area of the simplex τ_2; when $d = 3$, $p(\mathbf{v})$ is the summation of the squared face areas, $q(\mathbf{v})$ is the squared volume of the simplex τ_3.

The gradient and Hessian of GSM energy in d-simplex are:

$$\mathbf{g}_{\tau_d} = c_d \frac{\mathbf{p}'(\mathbf{v})q(\mathbf{v}) - p(\mathbf{v})\mathbf{q}'(\mathbf{v})}{q(\mathbf{v})^2}, \tag{3.18}$$

$$\mathbf{h}_{\tau_d} = c_d \frac{\mathbf{h}_p q(\mathbf{v}) + \mathbf{p}'(\mathbf{v})\mathbf{q}'(\mathbf{v})^T - \mathbf{q}'(\mathbf{v})\mathbf{p}'(\mathbf{v})^T - \mathbf{h}_q p(\mathbf{v}) - 2q(\mathbf{v})\mathbf{g}_{\tau_d}\mathbf{q}'(\mathbf{v})^T}{q(\mathbf{v})^2}, \tag{3.19}$$

where $\mathbf{p}'(\mathbf{v}), \mathbf{q}'(\mathbf{v}), \mathbf{h}_p$ and $\mathbf{h}_q$ are the first order and second order derivatives of $p(\mathbf{v}), q(\mathbf{v})$. For arbitrary vertex $\mathbf{v}$, its one-ring simplices set $\mathscr{T}_v$ are used for gradient and Hessian computations.

Based on the gradient and Hessian of GSM energy at any vertex, the search direction $\mathbf{\Psi} = \mathbf{h}^{-1}\mathbf{g}$ is obtained. Since GSM is not convex in the one-ring domain, the backtracking line search is used in both $\mathbf{\Psi}$ and $-\mathbf{\Psi}$ directions to obtain the optimal step size α. The initial value of α is set as the maximal possible movement inside its one-ring domain, i.e., $\alpha_{init} = \frac{max_{\mathbf{v}_i \in N_v}|\mathbf{v}-\mathbf{v}_i|}{|\mathbf{h}^{-1}\mathbf{g}|}$, where the set of vertices in $\mathscr{T}_v$ is N_v.

It is noted that Newton's method is defined for each vertex. The vertices are updated one by one following the descending order of the norm of vertex update vector, i.e., $\|\alpha\mathbf{h}^{-1}\mathbf{g}\|$. The worst positioned vertex is optimized first. The backtracking line search and vertex update avoid the increase of GSM energy as well as inverted simplex.

3.3.2 Connectivity Update

Traditional CVT and ODT energy functions for meshing are based on Delaunay triangulation, while our GSM energy is not congruent with Delaunay triangulation. Given an initial connectivity, a set of flip operations are employed to optimize the connectivity. The details of connectivity updating in triangular and tetrahedral meshes are different.

For a 3D triangular mesh, the initial mesh is obtained using surface constrained mesh generation (Yan et al., 2009). The 2-2 edge-flipping operation is used to decrease the energy. A flip operation is performed if the energy will be decreased after the flip operation. Besides that, we also need to make sure the new triangles are still restricted to the original surface and avoid to generate non-manifold edges, i.e. one edge shared by more than two triangles. The edges along the sharp features and boundary edges are never flipped. During the connectivity optimization, we traverse through all the edges until the energy does not decrease anymore.

For a 3D tetrahedral volume mesh, the initial mesh is built by TetGen (Si, 2015). There are several flip operations available in tetrahedral meshing, including 2-3 flip, 3-2 flip, 4-4 flip. Besides that, edge removal and multi-face removal proposed in (Shewchuk, 2002a) are utilized to further improve the connectivity. Comparing to edge flip in triangle mesh, flip operation in tetrahedral mesh may change the number of tetrahedrons. The flip operation will be performed if average GSM energy to each simplex decreases after the operation. During the connectivity optimization, we will traverse through all the edge and faces until the energy cannot decrease anymore.

3.3.3 Boundary and Feature

Sharp features and boundary edges of 2-simplex meshes are usually defined by a set of curves (1-simplex meshes). The boundary of a 3-simplex mesh is a 2-simplex mesh.

In 2-simplex mesh optimization, we first extract the boundary and feature curves and estimate vertex numbers on those curves. Then the vertices along those curves are optimized. After that, we randomly sample the remaining vertices on surface domain and optimize them by fixing those boundary and feature points. The vertex numbers on one surface boundary curves or sharp features are estimated in the following way. Suppose A_Ω is surface area and L_Ω is the length of boundary curves, according to Euler's polyhedron formula, we have:

$$N_s - \frac{F}{2} - \frac{L_\Omega}{2\sqrt{4A_\Omega/\sqrt{3}F}} = 2 - 2g, \tag{3.20}$$

where N_s is vertex number on boundary, F is face number of boundary, g is the genus of surface. Only F is unknown in Eq. (3.20), so we can compute F. Then edge length l_{est} is estimated by $l_{est} = \sqrt{4A_\Omega/\sqrt{3}F}$. Finally, the vertex number on each curve is obtained by dividing its length by l_{est}.

In tetrahedral meshing, the vertex number on boundary is estimated at the beginning based on the domain boundary area A_Ω and the domain volume V_Ω. Body-Centered Cubic (BCC) lattice is used for the estimation. Voronoi cell of each vertex in BCC lattice is a truncated octahedron. By using truncated octahedron as Voronoi cell, the edge length l_{est} of an equilateral tetrahedron satisfies

$$V_\Omega = \frac{N}{\sqrt{2}} l_{est}^3 - \frac{A_\Omega}{\sqrt{6}} l_{est} + 2 - 2g, \tag{3.21}$$

where N is total vertex number. l_{est} can be calculated from Eq. (3.21). Then the boundary vertex number is $N_b = \frac{2A_\Omega}{\sqrt{3}l_{est}^2} + 2 - 2g$. The 2-simplex mesh optimization is applied to the surface boundary with N_b vertices. After that, we randomly sample the remaining vertices inside the volume and optimize them with fixed boundary surface vertices.

In the following subsections, the vertex update rules for 1-simplex curves and 2-simplex surfaces are introduced.

Feature Curve

1-simplex GSM optimizations are performed on the corresponding curves, which has been discussed in Sec. 3.2.2. During 1-simplex mesh optimizations, all the vertices are restricted on the corresponding curves. The update vector is projected along the tangent direction of the curve at $\mathbf{v}$. Suppose the normalized tangent direction of the curve l at vertex $\mathbf{v}$ is $\mathbf{d}_{l_\mathbf{v}}$, then the vertex update rule of Newton's method based on Eq. (3.16) is:

$$\mathbf{v}^\star = \mathbf{v} - \left(\alpha \left(\mathbf{h_v}^{-1}\mathbf{g_v}\right)^T \mathbf{d}_{l_\mathbf{v}}\right) \mathbf{d}_{l_\mathbf{v}}. \tag{3.22}$$

After that, $\mathbf{v}^\star$ is projected to the closest point on the corresponding curve.

Boundary Surface

2-simplex GSM optimization is performed on the corresponding 3D surface, which has been discussed at the beginning of Sec. 3.2.2. During 3D surface optimization, vertices should be restricted to the given surface. When updating the vertex positions, vertex movement is only allowed on its tangent plane. Suppose the normal of vertex $\mathbf{v}$ is $\mathbf{n_v}$, then:

$$\mathbf{v}^* = \mathbf{v} - \left(\alpha\mathbf{h_v}\mathbf{g_v} - \left(\left(\alpha\mathbf{h_v}\mathbf{g_v}\right)^T \mathbf{n_v}\right)\mathbf{n_v}\right). \tag{3.23}$$

After that, $\mathbf{v}^\star$ is projected to the closest point on the given surface.

The GSM optimization framework is given in Alg. 1.

Algorithm 1: GSM Optimization

```
Input: vertex number $N$, boundary domain $\Omega$, anisotropic metric $\mathbf{M}$
Output: tetrahedral mesh $(V, T)$ with $N$ vertices
estimate boundary vertex number $N_b$ and edge length $l_{est}$;
if $\Omega$ contains sharp features then
    estimate vertex number $N_s$ for sharp features;
    optimize vertices on sharp features $V_s$ ;
end
optimize vertices $V_b$ by fix $V_s$ ;
randomly sample $N - N_b - N_s$ vertices inside $\Omega$ as $V_f$;
build tetrahedral mesh $T$ of all $N$ vertices;
for $k \leftarrow 0$ to $50$ do
    calculate update vector $\Delta\mathbf{v}$ of all vertices;
    for $i \leftarrow 0$ to $(N - N_b - N_s)/4$ do
        $\mathbf{v}_t \leftarrow \arg\max_{\mathbf{v} \in V_f} \|\Delta\mathbf{v}\|$ with Newton's Method ;
        update position of $\mathbf{v}_t$;
    end
    optimize connectivity $T$;
end
```

3.4 Experiment and Comparisons

We implement the algorithms using C++. The experiments are done on a workstation with Intel(R) Xeon E5645 CPU 2.40GHz, and 32G DDR3 RAM. To demonstrate the performance of the proposed GSM method, we compare it with four mesh optimization approaches provided by The Computational Geometry Algorithms Library (CGAL) (Jamin et al., 2015). The optimizations of CGAL mesher have two categories. One is the local optimization, including vertex perturbation (Tournois et al., 2009) and sliver exudation (Cheng et al., 2000). The other one is the global optimization, including Lloyd smoother (Du et al., 1999; Du and Wang, 2003) and ODT smoother (Alliez et al., 2005; Chen and Xu, 2004). In the following, the tetrahedral mesh quality criteria are introduced in Sec. 3.4.1. The experimental results on isotropic meshing are presented in Sec. 3.4.2 (smooth surface) and Sec. 3.4.3 (surface with sharp features), and the experimental results of adaptive and anisotropic meshings are provided in Sec. 3.4.4. Finally, the running time and robustness analysis are given in Sec. 3.4.5

and 3.4.6. Due to the page limit, we only present seven models in the following, more meshing results are given in a supplementary document. Table 3.1 gives detailed quality statistics of all volume meshing models in the experiment. The best result in each model is shown in bold font.

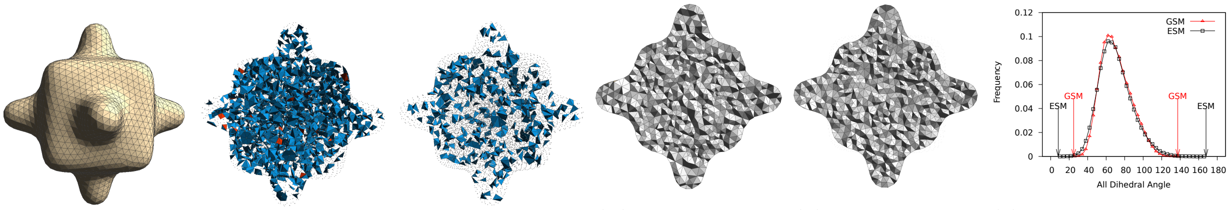
3.4.1 Quality Measurement

The quality criteria used in all our experiments for the isotropic meshing are dihedral angle θ and radius ratio $\gamma = 3\frac{r_{in}}{r_{circ}}$, where r_{in} is inradius and r_{circ} is circumradius. θ_{min} is the smallest dihedral angle. θ_{max} is the largest dihedral angles. $\bar{\theta}_{min}$ is the average value of the smallest dihedral angle of each tetrahedron. γ_{min} is the smallest radius ratios among all tetrahedrons. γ_{mean} is the average radius ratios of all tetrahedrons. The distribution of dihedral angles and radius ratios of all tetrahedrons are provided. Since the sliver is measured by the dihedral angles, we evaluate our experiments extensively by the number of tetrahedrons with different degrees of smallest dihedral angles 10°, 20°, 30°, and 40° as thresholds. For anisotropic tetrahedral meshes, each tetrahedron is transformed to the isotropic space, then its quality is measured based on the above isotropic criteria.

3.4.2 Isotropic Tetrahedral Meshing

Figure 3.4 shows the isotropic tetrahedral meshing results on the Bumpycube volume for comparison between GSM and ESM. GSM produces better θ_{min} and θ_{max} comparing to ESM. It is also observed that GSM has better performance on sizing control. The tetrahedral meshing result of ESM optimization has larger variance on tetrahedral volumes.

Figure 3.5 shows the isotropic tetrahedral meshing results on the Duck volume. With random initialization, GSM method produces meshes with better $\theta_{min}, \theta_{max}$, as well as radius ratios, which outperforms all other methods provided by CGAL (i.e., both local and global optimizations) as shown in Table 3.1. In order to generate high-quality tetrahedral meshes,



(a) Model (b) ESM (c) GSM (d) ESM Clipping (e) GSM Clipping (f) Dihedral Angle

Figure 3.4: Comparison of ESM and GSM on the Bumpycube volume. The red ones are tetrahedrons with the smallest dihedral angles less than 18°, while the blue ones are tetrahedrons with the smallest dihedral angles less than 36°.

we use Particle, Lloyd, and ODT results as the initializations, respectively, and then apply the proposed GSM, vertex perturbation, and sliver exudation to further suppress slivers. The results demonstrate that GSM method is an effective sliver remover. Figure 3.5 also provides the distributions of dihedral angles and radius ratios in Duck volume meshes of different methods.

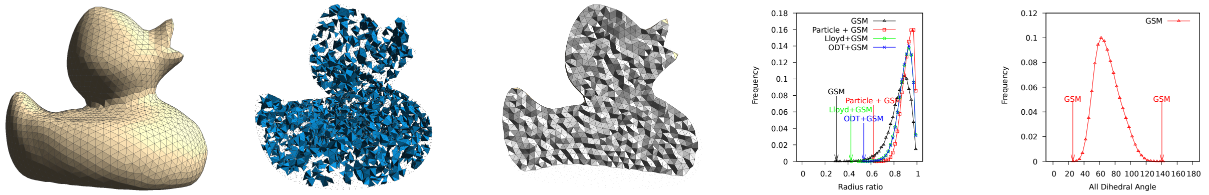
3.4.3 With Sharp Features

Figure 3.6 shows the isotropic tetrahedral meshing results as well as distributions of dihedral angles and radius ratios on the Fandisk volume with sharp features in different methods. We reach the same conclusion as the previous example that no matter with random initialization or initialized by Particle, Lloyd, and ODT results, our proposed GSM method obtains better $\theta_{min}, \theta_{max}$, as well as radius ratios, outperforming all other methods provided by CGAL. More detailed quality statistics comparison is given in Table 3.1.

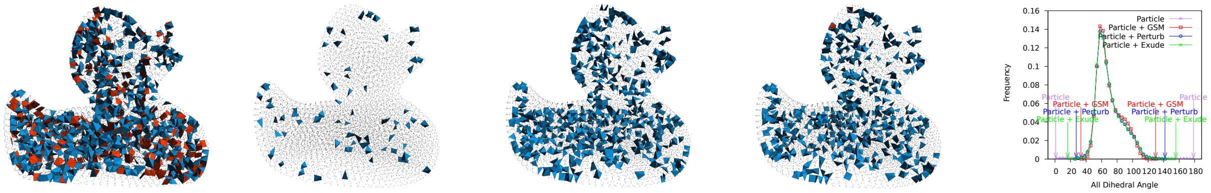
3.4.4 Adaptive and Anisotropic Tetrahedral Meshing

In order to show the generalization of the proposed GSM method, we also work on tetrahedral meshes in adaptive and anisotropic cases.

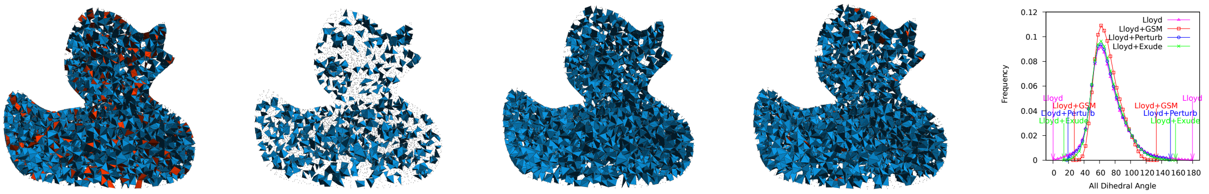
Figure 3.7 shows the adaptive tetrahedral meshing results on Sphere volume with scaling field, i.e. $\mathbf{M}(\mathbf{x}) = \mathbf{\Lambda}^2$, where $s = \left(0.025 + 0.2\left|\sqrt{x^2+y^2+z^2} - 0.5\right|\right)^{-1}$, $\mathbf{\Lambda} = diag\{s, s, s\}$, .



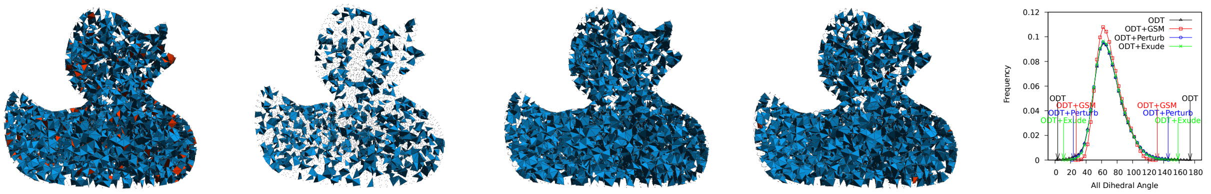
(a) Model (e) GSM (i) Particle+GSM Clipping (m) Radius Ratio (q) Dihedral Angle



(b) Particle (f) Particle+GSM (j) Particle+Perturb (n) Particle+Exude (r) Dihedral Angle



(c) Lloyd (g) Lloyd+GSM (k) Lloyd+Perturb (o) Lloyd+Exude (s) Dihedral Angle

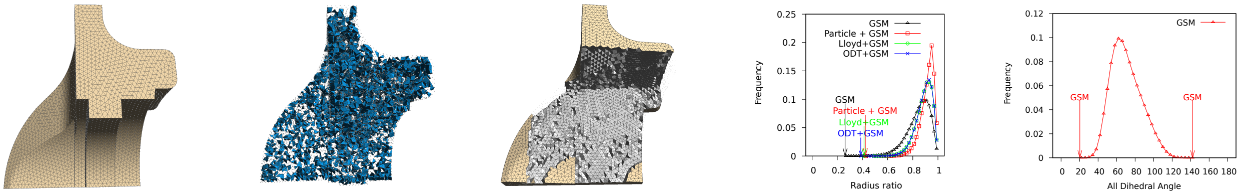


(d) ODT (h) ODT+GSM (l) ODT+Perturb (p) ODT+Exude (t) Dihedral Angle

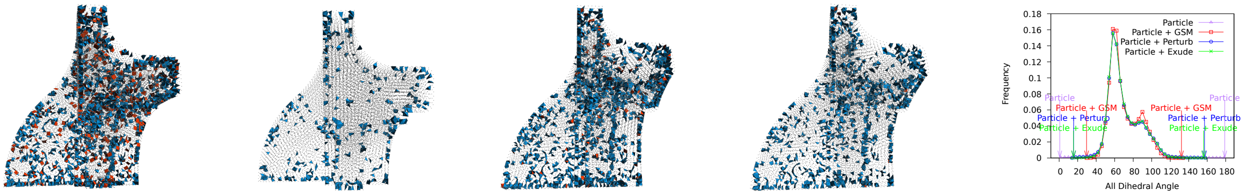
Figure 3.5: Duck volume meshing with 10,000 vertices. The red ones are tetrahedrons with the smallest dihedral angles less than 20, while the blue ones are tetrahedrons with smallest dihedral angles less than 40.

It is noted that the proposed GSM method leads to better dihedral angles and radius ratios among other methods provided by CGAL.

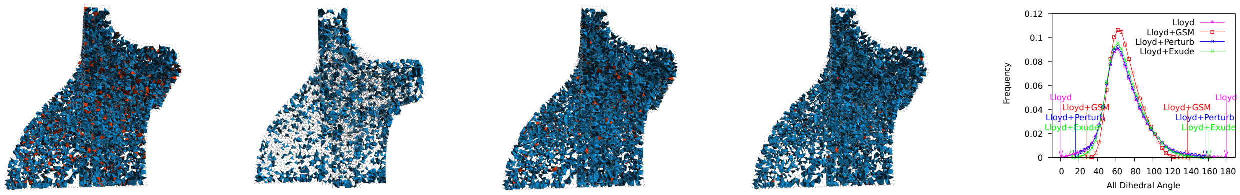
Figure 3.8 shows the anisotropic tetrahedral meshing results on the Sphere volume with metric field $\mathbf{M}(\mathbf{x}) = \mathbf{Q}^T(x)\mathbf{\Lambda}\mathbf{Q}(x)$, where $\mathbf{\Lambda} = diag(100, 10, 10)$, and $\mathbf{Q}$'s three columns are $(2\cos 6x, 1, 0)^T$ and two orthogonal unit vectors. Table 3.1 provides quality statistics of Sphere volume meshes in the specified anisotropic field with both random and Particle initializations. Here, only GSM and GSM with Particle initialization are compared, since



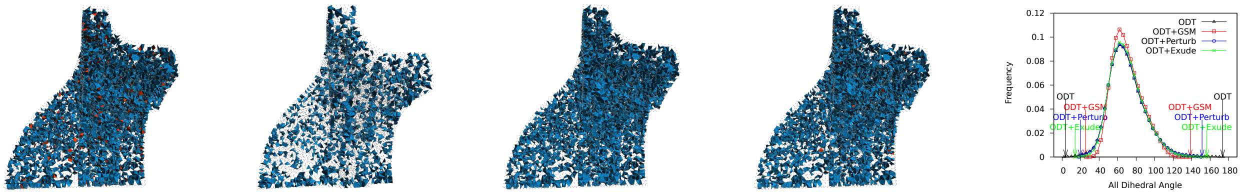
(a) Model (e) GSM (i) Particle+GSM Clipping (m) Radius Ratio (q) Dihedral Angle



(b) Particle (f) Particle+GSM (j) Particle+Perturb (n) Particle+Exude (r) Dihedral Angle



(c) Lloyd (g) Lloyd+GSM (k) Lloyd+Perturb (o) Lloyd+Exude (s) Dihedral Angle

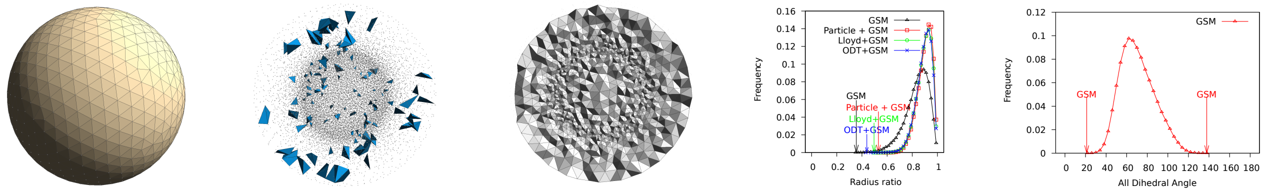


(d) ODT (h) ODT+GSM (l) ODT+Perturb (p) ODT+Exude (t) Dihedral Angle

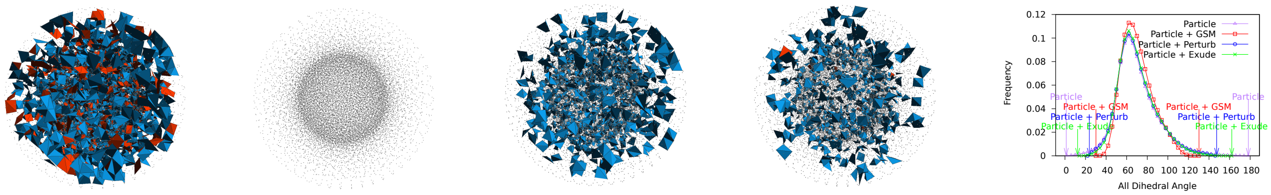
Figure 3.6: Fandisk volume meshing with 18,000 vertices. The red ones are tetrahedrons with the smallest dihedral angles less than 20, while the blue ones are tetrahedrons with the smallest dihedral angles less than 40.

according to our observations, the approach in GSM with Particle initialization can obtain best mesh quality among all other methods.

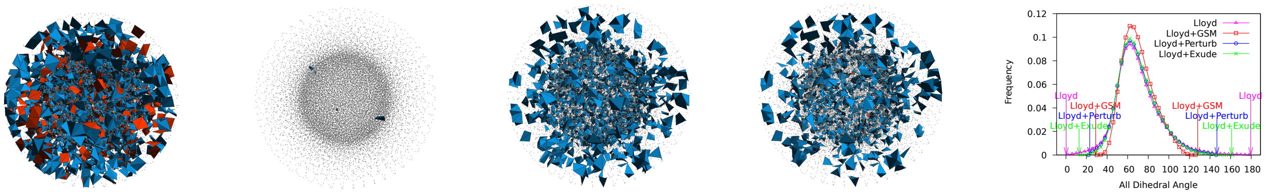
Figure 3.9 shows anisotropic tetrahedral meshing results inside a Cube with metric field defined as $\mathbf{M}(\mathbf{x}) = \mathbf{\Lambda}^2$, where $\mathbf{\Lambda} = diag\left((0.025 + 0.2(1 - e^{-x}))^{-1}, 5, 5\right)$. The proposed GSM method works well with both random and Particle initializations as shown in Table 3.1.



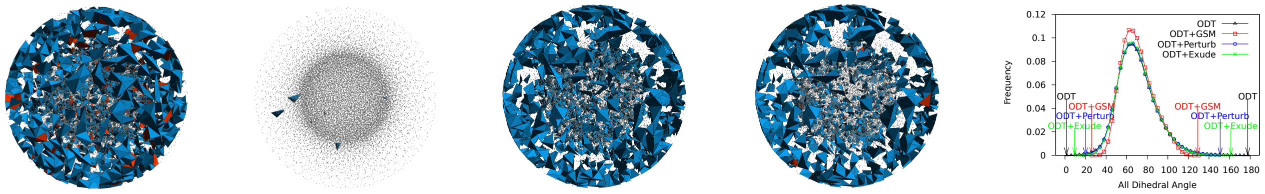
(a) Model (e) GSM (i) Particle+GSM Clipping (m) Radius Ratio (q) Dihedral Angle



(b) Particle (f) Particle+GSM (j) Particle+Perturb (n) Particle+Exude (r) Dihedral Angle

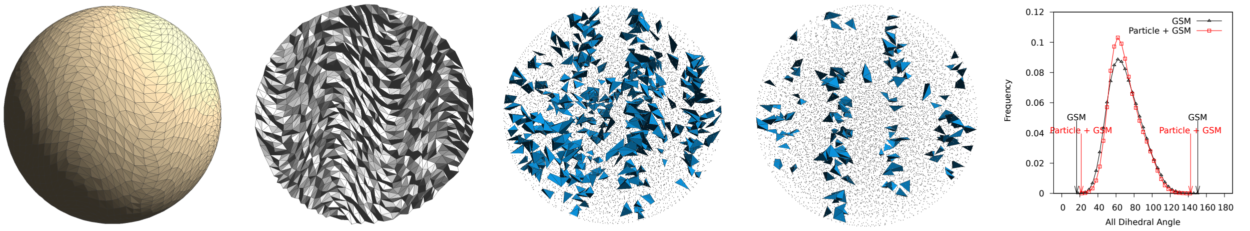


(c) Lloyd (g) Lloyd+GSM (k) Lloyd+Perturb (o) Lloyd+Exude (s) Dihedral Angle



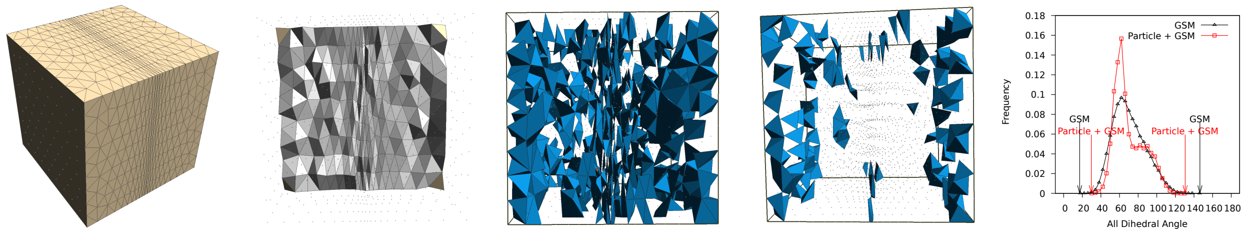
(d) ODT (h) ODT+GSM (l) ODT+Perturb (p) ODT+Exude (t) Dihedral Angle

Figure 3.7: Sphere volume meshing(10,000 vertices) with scaling field. The red ones are tetrahedrons with the smallest dihedral angles less than 15, while the blue ones are tetrahedrons with the smallest dihedral angles less than 30.



(a) Model (b) Particle+GSM Clipping (c) GSM (d) Particle + GSM (e) Dihedral Angle

Figure 3.8: Sphere volume meshing (10,000 vertices) with sinusoidal variation of anisotropy. The red tetrahedrons are the ones with smallest dihedral angles less than 15, while the blue tetrahedrons are the ones with smallest dihedral angles less than 30.



(a) Model (b) Particle+GSM Clipping (c) GSM (d) Particle + GSM (e) Dihedral Angle

Figure 3.9: Sphere volume meshing (10,000 vertices) with sinusoidal variation of anisotropy. The red tetrahedrons are the ones with smallest dihedral angles less than 15, while the blue tetrahedrons are the ones with smallest dihedral angles less than 30.

3.4.5 Running Time and Convergence Analysis

GSM is defined as a global optimization energy. Our GSM optimization iteratively updates vertex positions and vertex connectivities. There are several options to set the stop condition, i.e., maximum round number, energy decrease threshold, or any other mesh quality criteria. In our implementation, we set maximum round number as the criterion. In all of our experiments, maximum round number is set to be 50, which is large enough for all our experiments to converge. The time consumption is given in Table 3.1.

Duck volume is utilized as an example to show the relationship between θ_{min}, θ_{max}, $\bar{\theta}_{min}$, γ_{mean}, and computational time in Figure 3.10. The figure draws the first 40 rounds. The result shows that the optimizations converge fast so it provides good enough result after the first few rounds. θ_{min} is not strictly increasing along with the computational time and θ_{max} is not strictly decreasing along with the computational time, because we are not directly optimizing the minimal dihedral angle and the maximal dihedral angle. However, $\bar{\theta}_{min}$ and γ_{mean} keep increasing along with the computational time until reaching some local optimal solutions. Comparing to Perturb and Exude methods provided by CGAl, our implementation is not as efficient currently, but could be improved. For instance, if we compute vertex and connectivity updates smarter instead of a global strategy or use some parallel strategies, the implementation will be much more efficient. This will be one of our future work.

Table 3.1: Quality statistics of all volume meshing models. Note: best results of each model are in bold font.

Model	Method	$\theta_{min}/\theta_{max}$	$\gamma_{min}/\gamma_{mean}$	# < 10°	# < 20°	# < 30°	# < 40°	#tet	time (sec.)
Duck (Iso.)	Init	0.222/179	0.008/0.588	3422	12,830	25,445	38,473	57,898	n/a
	GSM	25.93/137.6	0.357/0.852	0	0	33	2211	49,782	1768.76
	Particle	1/179	0.02/0.898	190	413	737	1580	52,322	82.89
	Particle+GSM	**32/123.8**	**0.632/0.919**	0	0	**0**	**99**	51,522	1830.03
	Particle+Perturb	27.4/142	0.469/0.907	0	0	77	747	51,750	5.99
	Particle+Exude	16.3/156	0.305/0.906	0	3	85	625	51,736	3.66
	Lloyd	0.164/180	0.003/0.846	226	996	2592	6867	53,620	26.16
	Lloyd+GSM	30.05/132.7	0.411/0.893	0	0	**0**	923	50,800	1876.72
	Lloyd+Perturb	19.3/151	0.349/0.859	0	1	1480	5413	52,697	33.09
	Lloyd+Exude	14.1/158	0.273/0.866	0	52	704	4116	52,093	28.16
	ODT	4.18/174	0.084/0.875	36	325	1192	5036	52,571	17.28
	ODT+GSM	26.75/134.6	0.456/0.894	0	0	2	981	50,837	1089.84
	ODT+Perturb	23.6/146	0.413/0.881	0	0	603	4281	52,107	20.21
	ODT+Exude	11.1/159	0.26/0.883	0	41	470	3903	51,978	17.85
Fandisk (Feature)	Init	0.09656/179	0.0035/0.589	6039	21934	44258	67,110	102,135	n/a
	GSM	20.36/142.8	0.275/0.845	0	0	91	4550	88,015	4158.5
	Particle	0.6863/179	0.013/0.898	272	701	1324	3038	92,620	301.71
	Particle+GSM	**29.6/132.2**	**0.423/0.916**	**0**	**0**	**1**	**471**	91,241	4458.5
	Particle+Perturb	15.73/157.1	0.266/0.903	0	192	783	2379	91,545	6.79
	Particle+Exude	15.5/156.1	0.303/0.906	0	14	185	1290	90,442	6.59
	Lloyd	0.486/179.2	0.01/0.842	409	1865	4869	12,850	93,683	51.24
	Lloyd+GSM	27.51/136.9	0.413/0.889	0	0	6	1927	89,790	4315.6
	Lloyd+Perturb	16.15/156.3	0.292/0.852	0	535	3355	10,903	92,420	55.53
	Lloyd+Exude	12.56/160.5	0.219/0.863	0	84	1353	7802	90,329	52.4
	ODT	3.724/173.5	0.081/0.873	70	626	2218	9079	91,633	34.00
	ODT+GSM	25.06/138.4	0.386/0.889	0	0	12	2174	90,082	4345.86
	ODT+Perturb	19.59/151.2	0.369/0.877	0	3	1454	8209	90,705	35.71
	ODT+Exude	14/156.4	0.278/0.881	0	62	833	6899	89,663	32.32
Sphere (Adap.)	Init	0.23/179	0.005/0.558	7903	29,612	58,635	87,717	121,650	n/a
	GSM	21.1/138	0.355/0.837	0	0	118	5887	103,485	7840.78
	Particle	1.03/178	0.0203/0.856	372	1718	4655	12,282	111,505	410.37
	Particle+GSM	**30.2/130**	**0.531/0.900**	0	0	0	**1092**	106,106	8787.22
	Particle+Perturb	23.7/147	0.384/0.872	0	0	1801	8444	109,445	21.16
	Particle+Exude	12.3/162	0.210/0.874	0	53	1127	6794	108,789	6.79
	Lloyd	0.292/180	0.006/0.852	547	2234	6349	17,809	147,734	81.18
	Lloyd+GSM	28.9/128	0.496/0.895	0	0	4	2134	140,757	9636.53
	Lloyd+Perturb	23.2/147	0.368/0.867	0	0	2563	13,051	144,745	81.46
	Lloyd+Exude	12.8/161	0.245/0.869	0	108	1775	11,106	144,018	73.28
	ODT	1.47/178	0.0297/0.874	86	742	3284	15,068	144,176	54.9
	ODT+GSM	26.1/129	0.439/0.893	0	0	6	2854	140,776	8888.4
	ODT+Perturb	19.9/151	0.343/0.877	0	1	2484	14,134	143,547	47.29
	ODT+Exude	9.45/161	0.216/0.879	1	136	1755	12,616	142,922	47.19
Sphere (Aniso.)	GSM	16.5/150.5	0.147/0.812	0	7	500	6359	51,405	4659.15
	Particle+GSM	**21.5/142.5**	**0.289/0.864**	0	0	**174**	**3398**	52,206	4464.69
Cube (Aniso.)	GSM	17.2/146	0.271/0.832	0	2	33	737	8783	404.55
	Particle+GSM	**26.5/136**	**0.373/0.903**	0	0	**4**	**155**	25,253	453.9

The right figure in the second row in Figure 3.10 shows GSM energy changing along with the increasing of optimization round number in different initializations. In our GSM optimization, monotonic decrease of energy is guaranteed, so our optimization always converge with different initializations, including random, Particle, Lloyd, and ODT initializations. Although different initializations may lead to different final energies, Particle + GSM scheme

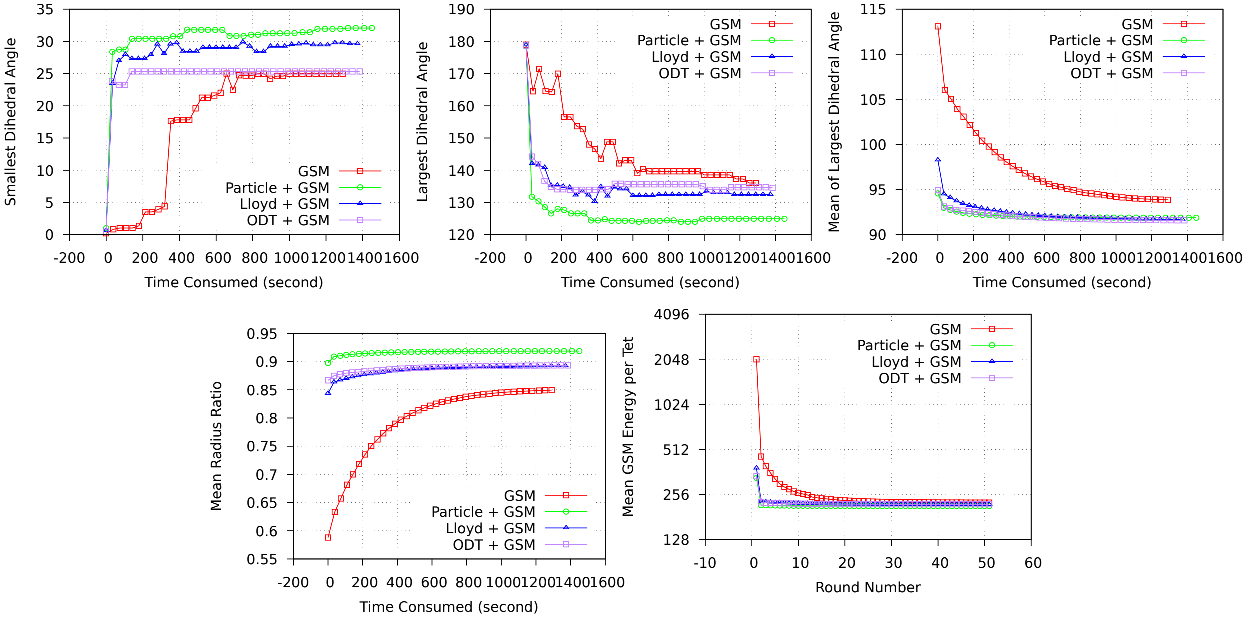


Figure 3.10: (a), (b), (c), (d) Dihedral angle and radius ratio quality changing along with computational time of the Duck model. (e) GSM energies changing along with the round numbers in different initializations.

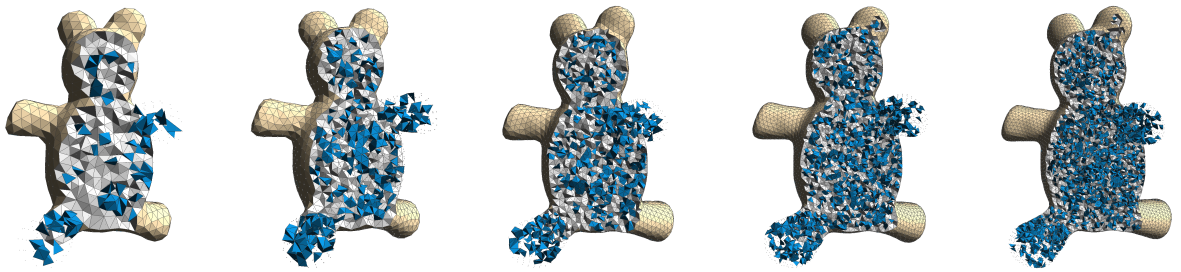
reaches the lowest convergence energy comparing to other initializations in this example (i.e., Duck volume model).

3.4.6 Robustness

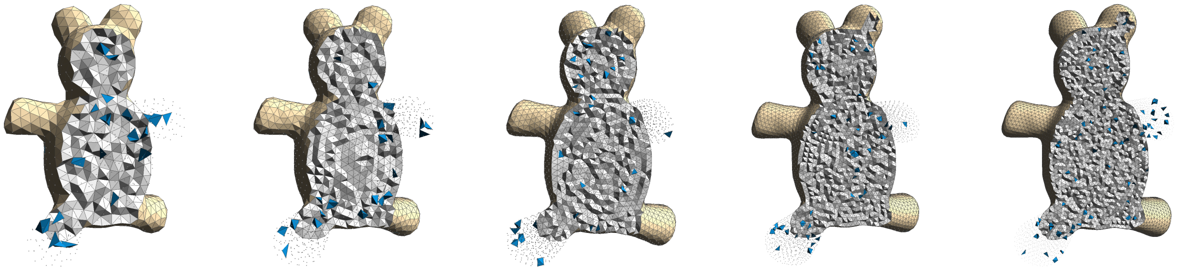
To show the robustness of our GSM energy, different vertex numbers on the Teddy volume are demonstrated. The volume clipping results are shown in Figure 3.11 and quality statistics e.g. $\theta_{min}, \theta_{max}, \gamma_{min}, \gamma_{mean}$ of GSM and Particle+GSM methods are shown in Table 3.2. The result shows that GSM has stable performance under different vertex numbers both as a standalone optimization method and as post-processing after Particle optimization.

Table 3.2: Quality statistics of Teddy volume meshes with different vertex numbers.

Method	Vertex Number	2000	5000	10,000	20,000	30,000
GSM	$\theta_{min}/\theta_{max}$	25.6/138.9	24.09/139.1	23/136.9	23.74/136.6	23.39/140.2
Particle+GSM	$\theta_{min}/\theta_{max}$	32.9/126.4	30.88/132.4	34.28/124	33.63/128.9	34.54/127.1
GSM	$\gamma_{min}/\gamma_{mean}$	0.309/0.842	0.347/0.848	0.402/0.853	0.36/0.853	0.353/0.853
Particle+GSM	$\gamma_{min}/\gamma_{mean}$	0.623/0.907	0.595/0.914	0.581/0.918	0.556/0.92	0.571/0.921



(a) GSM 2000 (b) GSM 5000 (c) GSM 10000 (d) GSM 20000 (e) GSM 30000



(f) Particle+GSM 2000 (g) Particle+GSM 5000 (h) Particle+GSM 10000 (i) Particle+GSM 20000 (j) Particle+GSM 30000

Figure 3.11: Teddy volume meshes with different vertex numbers. The blue ones are tetrahedrons with the smallest dihedral angle less than 40°. Note: there's no tetrahedron with the smallest dihedral angle less than 20°.

3.5 Conclusion

In this chapter, we introduce an effective sliver suppression method based on shape matching idea. It generates high-quality tetrahedral meshes in isotropic, adaptive, and anisotropic cases. The proposed GSM method is evaluated on extensive volume models and compared with state-of-the-art approaches. The results of proposed GSM method show much better performance than all other current methods. In the future, we would like to improve the computation speed by using GPU parallel techniques.

CHAPTER 4

FIELD ALIGNED FRAMEWORK[1]

4.1 Overview

Meshes are composed of a set of connected and non-overlapped simplex, e.g., triangles for the triangular meshes, quads for the quadrilateral meshes, tetrahedrons for the tetrahedral meshes, and hexahedrons for the hexahedral meshes. The main tasks of mesh generation include vertex positions optimization and vertex connectivity optimization. The field alignment framework focuses on the vertex positions. The idea of the framework comes from the symmetric one-ring structures of those meshes. In triangular mesh, the one-ring structure forms a hexagon. In quadrilateral mesh, the one-ring structure is in the form of a cross. The meaning of field alignment is to align the one-ring structure with a given frame field, so that the one-ring structure is uniquely determined according to the given frame field. Our field alignment problem is modeled as particle optimization problem.

4.2 Particle-Based Optimization Method

In the particle-based framework, each vertex is modeled as a particle with certain inter-particle potential energy, the derivative of which determines the inter-particle forces. The position of particles are optimized according to the forces from their neighbors until they reach the equilibrium. In the following subsection, we introduce how to design the potential energy which can guide particles to form the desired one ring structure, when they reach the equilibrium.

---

[1] ©2018 Eurographics Association, Revised, with permission, from Saifeng Ni, Zichun Zhong, Jin Huang, Wenping Wang, Xiaohu Guo,“Field-Aligned and Lattice-Guided Tetrahedral meshing”, in Computer Graphics Forum (Proceeding of SGP), Vol.35, pp. 161-172, 2018.

4.2.1 Gaussian Hole Kernel

In the particle framework, in order to form the expected pattern at equilibrium state, it is important to define a suitable inter-particle force derived from potential energy that leads to the expected pattern. The entire particle system's equilibrium is got by minimizing the sum of potential energies. Gaussian kernel is radially-symmetric, thus defining the inter-particle energy using it resembles packing of circles/spheres in 2D/3D, as demonstrated for anisotropic triangular meshing of surfaces (Zhong et al., 2013). Suppose two neighboring particles i and j are located at $\mathbf{p}_i$ and $\mathbf{p}_j$, respectively, their radially-symmetric energy can be defined as: $e^{-\frac{\|\mathbf{v}_{ij}\|^2}{2\sigma^2}}$, where $\mathbf{v}_{ij} = \mathbf{p}_i - \mathbf{p}_j$, and σ is the standard deviation of the Gaussian kernel. However, such radial-symmetry means that this potential energy does not have directional alignment property. In other words, given two different cross fields (with rotation only), their particle optimization results will be the same. The desired pattern may also equilibrium states for the Gaussian kernel, for example quadrilateral / hexahedral patterns are equilibrium states for the Gaussian kernel in 2D / 3D cases, but they are not stable. Any small perturbation from the regular quadrilateral / hexahedral grid will break the equilibrium. So they are unreachable states when minimizing the energy.

We need to construct a more specific potential energy to get the desired one-ring structure. Once the frame field and the target edge length are given, the one-ring neighbors of a particle are fixed accordingly. Radial-symmetry is not enough to form the particular one-ring structures locally. Our goal is to force neighbor particles to fall into each others' desired one-ring neighbor positions exactly, by minimizing the potential energy. To achieve such property, we place negative Gaussian kernels right at the desired one-ring neighbor positions, which is like digging a *hole* at those positions in the energy field: $-e^{-\frac{\|\mathbf{v}_{ij}-\mathbf{Onering}(k)\|^2}{2\sigma^2}}$, where $\mathbf{Onering}(k)$ is the k-th one-ring neighbor position, e.g. BCC in Eq. (2.1) and FCC in Eq. (2.2), $k = 1...N_{or}$. When we minimize such a potential energy, the neighboring particles will be pushed exactly to those *holes*. Besides that, we also include a positive Gaussian kernel at the position of the particle itself which will push its neighbors away to avoid particles being optimized to the same positions. We call this potential energy as *Gaussian Hole Kernel* (GHK):

$$E_{ij} = e^{-\frac{\|\mathbf{v}_{ij}\|^2}{2\sigma^2}} - \frac{1}{N_{or}} \sum_{k=1}^{N_{or}} e^{-\frac{\|\mathbf{v}_{ij} - \mathbf{Onering}(k)\|^2}{2\sigma^2}}. \tag{4.1}$$

To generate anisotropic field-aligned pattern, we will transform the anisotropic alignment problem to an isotropic one locally based on the given frame field. When particles form a regular pattern aligned with the axes of Cartesian coordinate system, particles in the anisotropic space will exhibit the desired one-ring pattern aligned with the desired frame field. Each particle i is associated with a matrix $\mathbf{T}_i$ expanded by three vectors $\{\mathbf{t}_{i1}, \mathbf{t}_{i2}, \mathbf{t}_{i3}\}$. Those three vectors define the local alignment of cubic lattice. Suppose there is no degenerate case, i.e., $|\mathbf{T}_i| \neq 0$, then the corresponding matrix $\mathbf{B}_i = \mathbf{T}_i^{-1}$ transforms the anisotropic space to an isotropic one locally: $\mathbf{B}_i\mathbf{T}_i = \mathbf{B}_i\{\mathbf{t}_{i1}, \mathbf{t}_{i2}, \mathbf{t}_{i3}\} = \mathbf{I}$. In other words, B_i transforms an anisotropic structure to an isotropic one locally. If we take $\{\mathbf{t}_{i1}, \mathbf{t}_{i2}, \mathbf{t}_{i3}\}$ as basis of the anisotropic space, then any vector $\mathbf{v} = k_1\mathbf{t}_{i1} + k_2\mathbf{t}_{i2} + k_3\mathbf{t}_{i3}$ in the anisotropic space has a corresponding vector $\mathbf{v}' = \mathbf{B}_i\mathbf{v} = diag\,(k_1, k_2, k_3)\,\mathbf{I}$ in the mapped isotropic space, where the one-ring neighbors of each vertex are well defined, e.g., BCC in Eq. (2.1) and FCC in Eq. (2.2).

Suppose there are N particles $\mathbb{V} = \{\mathbf{p}_i | i = 1...N\}$. For two neighboring particles i and j, we use $\mathbf{T}_{ij}$ as the frame field evaluated at the middle of two particles, i.e., $\mathbf{T}_{ij} = \mathbf{T}\left(\frac{\mathbf{p}_i + \mathbf{p}_j}{2}\right)$, and correspondingly the matrix $\mathbf{B}_{ij} = \mathbf{T}_{ij}^{-1}$ for transforming $\mathbf{v}_{ij}$ from its anisotropic space to the isotropic one. The energy of Eq. (4.1) between two neighboring particles i and j can be modified as:

$$E'_{ij} = e^{-\frac{\|\mathbf{B}_{ij}\mathbf{v}_{ij}\|^2}{2\sigma^2}} - \frac{1}{N_{or}} \sum_{k=1}^{N_{or}} e^{-\frac{\|\mathbf{B}_{ij}\mathbf{v}_{ij} - \mathbf{Onering}(k)\|^2}{2\sigma^2}}. \tag{4.2}$$

Here σ should be proportional to the expected edge length l^*. We discuss the choice of a proper value for σ in Sec. 4.3.3.

Note that we denote all the symbols in the isotropic space with a prime symbol ($'$). The energy E'_{ij} is defined in the isotropic space. The negative of first-order derivative of E'_{ij} with respect to $\mathbf{p}'_i$ is the force defined in the isotropic space: $\mathbf{f}'_{ij} = -\frac{\partial E'_{ij}}{\partial \mathbf{p}'_i}$. Since the particle positions $\mathbf{p}_i$ are optimized in the anisotropic space, we transform the force back to the anisotropic space: $\mathbf{f}_{ij} = \mathbf{T}_{ij}\mathbf{f}'_{ij}$, which is:

$$\mathbf{f}_{ij} = \frac{\mathbf{v}_{ij}}{\sigma^2} e^{-\frac{\|\mathbf{B}_{ij}\mathbf{v}_{ij}\|^2}{2\sigma^2}} - \frac{\mathbf{T}_{ij}}{N_{or}} \sum_{k=1}^{N_{or}} \frac{\mathbf{B}_{ij}\mathbf{v}_{ij} - \mathbf{Onering}(k)}{\sigma^2} e^{-\frac{\|\mathbf{B}_{ij}\mathbf{v}_{ij} - \mathbf{Onering}(k)\|^2}{2\sigma^2}}. \tag{4.3}$$

Our energy definition in Eq. (4.2) satisfies $E_{ij} = E_{ji}$, and the force definition in Eq. (4.3) satisfies $\mathbf{f}_{ij} = -\mathbf{f}_{ji}$.

4.2.2 Lattice Optimization

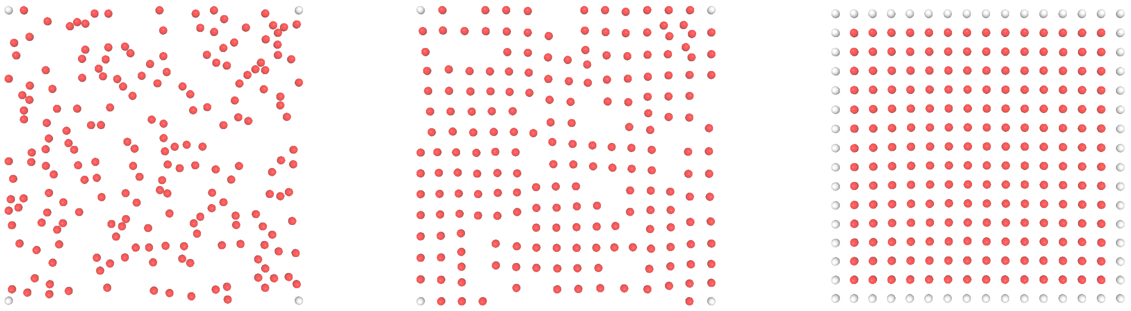
Once the inter-particle energy is defined, the particle optimization problem is modeled as an energy minimization problem. The variables are the particle positions $\mathbb{V} = \{\mathbf{p}_i | i = 1...N\}$, which are constrained in domain Ω. The problem is formulated as follows:

$$\begin{aligned} \min \quad & E(\mathbb{V}) = \sum_i \sum_{j \neq i} E_{ij} \approx \sum_i \sum_{j \in \mathbb{N}(i)} E_{ij} \\ s.t. \quad & \mathbf{p}_i \in \Omega, \quad \forall i = 1...N \end{aligned} \tag{4.4}$$

where $\mathbb{N}(i)$ is the set of neighbors of particle i within distance R. Instead of considering the inter-particle energy between every pair of particles, we only consider the energy of two particles within distance R. We call R the neighbor radius. Gaussian energy is close to 0 when $R >= 5\sigma$. This approximation affect very little to the total energy while significantly reducing the number of items in the energy summation from $O(N^2)$ to $O(N)$. We use k-d tree to query the neighbors for each particle. When the frame field has large stretching ratio, it is also necessary to adjust the query radius accordingly since k-d tree is built based on Euclidean distance. The energy and force related to particle i is E_i and $\mathbf{f}_i$, which is the sum of inter-particle energies and forces from its neighbors $\mathbb{N}(i)$.

It is not easy to add the constraint condition in Eq. (4.4) to the energy explicitly. When L-BFGS (Liu and Nocedal, 1989) is used to solve the energy minimization, we loosen the constraint condition a little bit. To preserve the domain boundary as well as sharp features of the domain, particles are classified into four types: *fixed particles*, *sharp edge particles*, *boundary particles*, and *free particles*. Fixed particles are corner points of the domain boundary. In our implementation, we simply calculate the dihedral angles between neighboring triangles to detect all the sharp edges in the input surface mesh. A corner is identified if it is shared by more than two sharp edges. During the optimization, the gradient of the sharp edge particle will be projected onto the direction of its underlying sharp edge, and the gradient of boundary particles will be projected onto the tangent plane of its boundary surface. After each round of L-BFGS optimization, we will project particles to the domain boundary if it is either outside the domain or inside but close to the boundary. If a boundary particle is close to a sharp edge, then it is projected to the sharp edge and labeled as a sharp edge particle. This is used to maintain the constraint in Eq.(4.4).

The details of our L-BFGS particle optimization algorithm are illustrated in Alg. 2. Stopping criteria are discussed in the end of Sec. 4.2.2.



(a) Initial Particles (b) Before Ins and Del (c) After Ins and Del

Figure 4.1: Initial particle distribution, and particles before and after insertion and deletion. The gray ones are particles on the boundary. The red ones are particles inside the boundary.

Algorithm 2: L-BFGS Particle Optimization Algorithm

```
Input: $l^*$, $\Omega$, $\mathbf{T}$, $\mathbb{V}$
Output: Optimized $\mathbb{V}$
while stopping criteria not satisfied do
    Build k-d tree for $\mathbb{V}$ ;
    $E \leftarrow 0$;
    foreach $\mathbf{p}_i \in \{\mathbb{V} - fixedparticles\}$ do
        Query the neighbors $\mathbb{N}(i)$ from k-d tree;
        Calculate $E_i$ and $\mathbf{f}_i$;
        if $\mathbf{p}_i$ is a sharp edge particle or a boundary particle then
            Update $\mathbf{f}_i$ ;
        end
        $E \leftarrow E + E_i$;
    end
    Run L-BFGS with $E$ and $\{\mathbf{f}_i | i = 1...N\}$ to update $\mathbb{V}$;
end
foreach $\mathbf{p}_i \in \mathbb{V}$ do
    if $\mathbf{p}_i$ is outside of domain or its distance to boundary $\leq 0.3l^*$ then
        Project and mark it as a boundary particle;
    end
    if $\mathbf{p}_i$ is a boundary particle and its distance to sharp edge is $\leq 0.3l^*$ then
        Project and mark it as a sharp edge particle;
    end
end
```

Particle Insertion and Deletion

Minimizing the GHK energy encourages each particle to fall into a nearby hole. If there is no initial particles near a hole, then that hole will be left empty. If more than one particles are close to a hole, then those particles will compete for that hole. So the random initialization of particles (Figure 4.1a) results in some regions missing particles and some regions packing with extra particles as shown in Figure 4.1b. Hence we need a *particle insertion and deletion* algorithm to overcome this problem and obtain the desired patterns, e.g., Figure 4.1c.

The existing mesh refinement schemes are designed based on the mesh structure, e.g., inserting a vertex at the center of an edge or the centroid of a face. In the particle optimization stage, we do not build the mesh, which provides efficency especially for anisotropic cases. Inspired by the existing mesh refinement scheme, we design the following "mesh-free" insertion and deletion schemes.

Particle Deletion Scheme: Without connectivity, each particle does not have a well-defined one-ring neighbor. But we can query the neighbors $\mathbb{N}(i)$ using k-d tree for any particle i. After that we calculate the anisotropic distance to its neighbors and sort the distance in ascending order. Suppose we store the sorted distance in array D_i, the particle i is deleted if any of the following condition holds:

- $D_i[0] < 0.5 * l_{closest}$;
- $\frac{1}{2}\sum_{k=0}^{1} D_i[k] < 0.75 l_{closest}$;
- $\frac{1}{4}\sum_{k=0}^{3} D_i[k] < 0.85 l_{closest}$;
- $\frac{1}{6}\sum_{k=0}^{5} D_i[k] < 0.9 l_{closest}$;
- $\frac{1}{8}\sum_{k=0}^{7} D_i[k] < 0.95 l_{closest}$;

where l_{cloest} is the closest one-ring neighbor distance, e.g., $l_{closest} = l^*$ for hexahedral structure, $l_{closest} = \sqrt{3}/2l^*$ for BCC, and $l_{closest} = \sqrt{2}/2l^*$ for FCC. We denote the set of particles to be deleted as $\mathbb{S}_D$. The general rule of setting the coefficients is stricter constrain for the average distance of more neighbors. The coefficients given above are based on our observation in the experiments. All of our experiments are conducted with the same empirical values.

Particle Insertion Scheme: Unlike the particle deletion scheme, the first step of particle insertion is to get the insertion candidates. Inspired by one-ring structure of the desired lattice, we collect the candidate set by going through each particle and add all expected positions of its one-ring neighbors to the candidate set $\mathbb{S}_I = \cup_{i=1}^{N} \cup_{k=1}^{N_{or}} \{p_i + \mathbf{T}_i * \mathbf{Onering}(k)\}$, where $\mathbf{T}_i$ is the frame field at particle i. If a candidate is outside the domain, we will project it to the domain boundary. The coinciding duplicates will be removed from the set $\mathbb{S}_I$ and

Algorithm 3: Particle Deletion Scheme

```
Input: input vertex set $V$, neighbor radius $R$
Output: a set of vertices to delete $S_D$
initialize an empty set $S_D$;
foreach $\mathbf{v} \in V$ do
    initialize an empty array $D$;
    foreach vertex $\mathbf{v}_n$ within distance $R$ of $\mathbf{v}$ do
        if $\mathbf{v}_n \notin S_D$ then
            append distance between $\mathbf{v}$ and $\mathbf{v}_n$ to array $D$;
        end
    end
    sort $D$ in ascending order;
    if $D[0] < 0.5l_{closest}$ or $\sum_{i=0}^{1} D[i]/2 < 0.75l_{closest}$ or $\sum_{i=0}^{3} D[i]/4 < 0.85l_{closest}$ or
      $\sum_{i=0}^{5} D[i]/6 < 0.9l_{closest}$ or $\sum_{i=0}^{7} D[i]/8 < 0.95l_{closest}$ then
        $S_D \leftarrow S_D \cup \{\mathbf{v}\}$;
    end
end
```

also candidates coinciding with any particle will also be removed. We define two positions as coincidence if their distance is less than $0.1l^*$. After filtering the candidate set based on coincidence, we will calculate GHK energy for each remaining candidate and sort them in ascending order. Then we examine candidates one by one and pick a candidate if the nearest particle, including previously picked candidates, is at least $0.75l_{cloest}$ away. Too small insertion threshold will cause unnecessary vertices being inserted and may slow the convergence. Too large insertion threshold will insert less vertices than required. The coefficient 0.75 is set according to our observation in the experiments.

The particle deletion and insertion schemes are performed after each round of L-BFGS optimization. In an L-BFGS optimization round, the particle number is fixed. After particle deletion and insertion, another round of L-BFGS optimization is performed. As the optimization processed, less particles are deleted and inserted. The overall lattice optimization process is given in Alg. 5.

Algorithm 4: Particle Insertion Scheme

```
Input: input vertex set $V$ and two thresholds $d_{ins}, e_{ins}$
Output: a set of vertices to insert $S_I$
initialize an empty candidate set $C$;
foreach $\mathbf{v} \in V$ do
    get the frame field of $\mathbf{v}$ as $\begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \mathbf{t}_3 \end{bmatrix}$;
    initialize an empty set $P$;
    for $i \leftarrow 1$ to 3 do
        $P \leftarrow P \cup \{\mathbf{v} + l^* \cdot \mathbf{t}_i, \mathbf{v} - l^* \cdot \mathbf{t}_i\}$;
    end
    foreach $\mathbf{p} \in P$ do
        if $\mathbf{p}$ is outside of the domain then
            project it onto domain boundary;
        end
        $d_v \leftarrow$ the closest distance of $\mathbf{p}$ to $V$;
        if $d_v > d_{ins}$ then
            $C \leftarrow C \cup \{\mathbf{p}\}$;
        end
    end
end
initialize an empty set $S_I$;
foreach $\mathbf{c} \in C$ in ascending order of their Gaussian hole energy $e_c$ do
    if $e_c < e_{ins}$ then
        $d_h \leftarrow$ closest distance of $\mathbf{c}$ to $S_I$ ;
        if $d_h > d_{ins}$ then
            $S_I \leftarrow S_I \cup \{\mathbf{c}\}$;
        end
    end
end
```

The complexity of our particle optimization algorithm is related to the number of particles. A small target edge length indicates large number of particles, which is time-consuming. To speed up the optimization, we can start with the particle optimization (Alg. 5) by setting the target edge length as $2l^*$. After the optimization is completed, we get a particle set $\mathbb{V}$ with edge length $2l^*$. Using a similar strategy as the particle insertion scheme, we collect the candidates to refine the particle set. For each particle, the point which is l^* distance away along the frame field vectors is added to the refined candidates $\mathbb{V}_r$. After removing

Algorithm 5: Particle-Based Lattice Optimization Algorithm

```
Input: $l^*$, $\Omega$, $\mathbf{T}$
Output: particle set $\mathbb{V}$
Estimate vertex number $N$;
Randomly initialize $\mathbb{V}$; Optimize $\mathbb{V}$ by Alg. 2;
for $i \leftarrow 0$ to MaxRoundNum do
    Apply particle deletion scheme, $\mathbb{S}_D \leftarrow$ deleted particles;
    $\mathbb{V} \leftarrow \mathbb{V} \setminus \mathbb{S}_D$;
    Apply particle insertion scheme, $\mathbb{S}_I \leftarrow$ inserted particles;
    $\mathbb{V} \leftarrow \mathbb{V} \cup \mathbb{S}_I$;
    if $|\mathbb{S}_D| + |\mathbb{S}_I| == 0$ then
        break;
    end
    Optimize $\mathbb{V}$ by Alg. 2;
end
Optimize $\mathbb{V}$ by Alg. 2;
```

the coincidence candidates in $\mathbb{V}_r$, we take the particles in $\mathbb{V}$ and $\mathbb{V}_r$ as initial particles, and start another round of optimization by setting the target edge length as l^*. With a better initialization, the optimization converges much faster. If l^* is too small, we can start with $2^k l^*$ and do the above trick for k iterations. Such refinement strategy not only accelerates the optimization, but also helps converge to a better result. Both gradient norm $EpsG$ and max iteration number $MaxIts$ are set as the stopping criteria in Alg. 2. $EpsG$ is set to 1e−2 for each round, except the last round, where $EpsG = 1\text{e}{-4}$. If $k = 0$, $MaxRoundNum$ is set to 16. If $k = 1$, $MaxRoundNum = 8$. If round number is less than 5, $MaxIts$ is set tp 13, otherwise it is set to 8. The last round is without the $MaxIts$ as stopping criteria. We use the kitten Model as an example to show the energy decreasing with respect to computation time as shown in Fig. 4.2 with 20 rounds of LBFGS optimization. The energy decreases the most at the first few rounds. The increase of energy at the beginning of the round is due to the particle deletion and insertion scheme. The number of vertices being inserted and deleted are given in Fig. 4.2. At the end of each round, the energy is optimized to be smaller than the previous round.

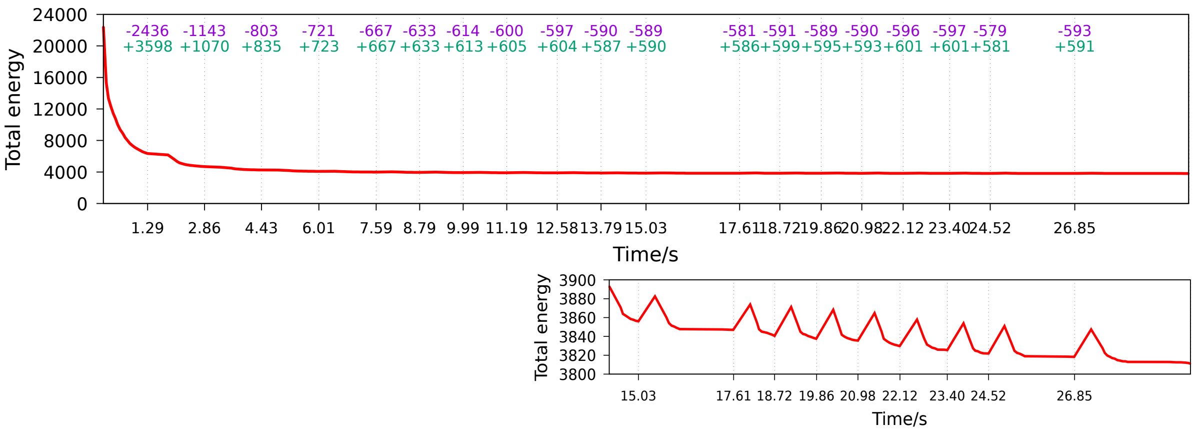


Figure 4.2: The energy curve of Kitten Model about BCC optimization with 20 rounds of LBFGS optimization. The purple and green numbers are the number of being deleted and inserted at the end of each round respectively. The zoom-in view of the last 10 rounds is also provided.

4.3 Field Aligned Tetrahedral Meshing

4.3.1 Overview

Field-aligned quadrilateral and hexahedral meshing are active research topics in recent years (Panozzo et al., 2014; Sokolov et al., 2016; Gao et al., 2017). For quadrilateral and hexahedral meshes, field alignment is very natural because the edges of those meshes are expected to agree with the vectors defining the underlying frame fields. Field-alignment includes the alignments of both Riemannian *distances* and *directions*. In triangular meshing, there has been researches focusing on the alignments of either Riemannian distances only (Zhong et al., 2013; Fu et al., 2014; Nieser et al., 2012), or rotational directions only (Jakob et al., 2015; Du et al., 2018). However, there is no consideration of both factors. Similarly, for field-aligned tetrahedral meshing, only the Riemannian distance has been considered (Labelle and Shewchuk, 2003; Du and Wang, 2005a; Fu et al., 2014; Boissonnat et al., 2015). So far we have not found any tetrahedral meshing work that takes alignments of both Riemannian distance and direction into consideration.

The triangle and the tetrahedron are the simplest elements in 2D and 3D, respectively. The dihedral angle of a regular tetrahedron is 70.53°. Unlike tiling regular triangles for 2D Euclidean space, it is impossible to tile regular tetrahedra for 3D Euclidean space. For most of the existing variational tetrahedral meshing algorithms, e.g., either Centroidal Voronoi Tessellation (CVT) based (Du and Wang, 2003; Alliez et al., 2005; Liu et al., 2009), or Optimal Delaunay Triangulation (ODT) based methods (Chen and Holst, 2011; Chen et al., 2014), the majority of their outputs are close to regular tetrahedra, accompanied by some badly shaped tetrahedra. This is one of the reasons that slivers are notoriously hard to remove in tetrahedral meshing (Klingner and Shewchuk, 2007; Tournois et al., 2009), and also one of the reasons that direction-aligned tetrahedral meshing has not been discussed.

Our motivation is to generate tetrahedral meshes with high quality elements, instead of regular tetrahedra, which can pack the 3D Euclidean space. Body-Centered Cubic (BCC) and Face-Centered Cubic (FCC) lattices are two close packing scheme of spheres in 3D. The corresponding tetrahedra formed by BCC and FCC lattices have high quality (Du and Wang, 2005b), which has been confirmed and used in mesh generation (Labelle and Shewchuk, 2007) and applications (Ando et al., 2013). Besides that, the symmetric cubic structures of BCC and FCC also allow us to build field-aligned anisotropic tetrahedral meshes.

We propose a particle-based variational method to generate field-aligned cubic lattice, which leads to anisotropic tetrahedral meshes. We design a *Gaussian Hole Kernel* as potential energy of the particle system, which effectively and efficiently optimize a set of particles to display the desired lattice patterns in their equilibrium status. To the best of our knowledge, this is the first approach that can generate field-aligned isotropic and anisotropic tetrahedral meshes, achieved by our particle-based cubic lattice (BCC and FCC) optimization method. As illustrated by our experiments, the field-aligned and lattice-guided tetrahedral meshing provides two benefits: (1) for isotropic tetrahedral meshing, having a direction field to guide the mesh could potentially improve mesh quality, especially for models with rotational features; (2) for anisotropic tetrahedral meshing, having BCC/FCC to guide the mesh can generate higher quality meshes as compared to other state-of-the-art methods.

4.3.2 Tetrahedral Mesh Generation

After particle optimization, we will connect the particles to build a tetrahedral mesh. Restricted Voronoi Diagram (RVD) (Yan et al., 2009) is used to build the surface boundary using the boundary particles. We use the RVD class provided by GEOGRAM (Lévy, 2015). Once we get the boundary triangle mesh, we perform the restricted Delaunay tetrahedralization by TetGen(Si, 2015), which does not consider the anisotropic frame field. To get a tetrahedral mesh with respect to the frame field, we perform a set of topological operations (Shewchuk, 2002a), by using the Gradient-Based Shape Matching Energy (Ni et al., 2017) as guidance to flip the tetrahedral mesh.

4.3.3 Experiments of Field Aligned Tetrahedral Meshing

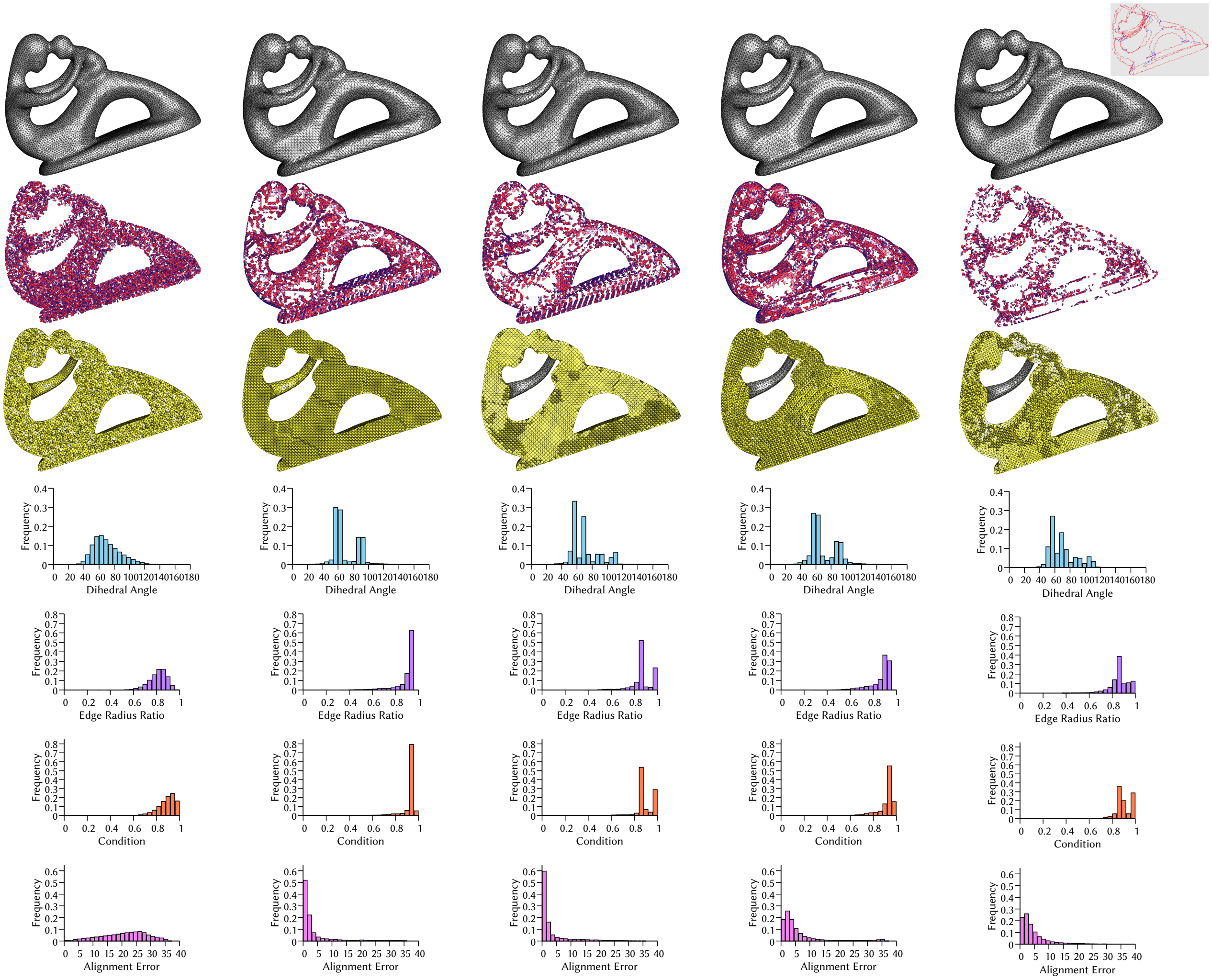
We compare our methods with the state-of-the-art methods (Jamin et al., 2015; Fu et al., 2014; Zhong et al., 2013). The implementation of our algorithms are based in C++. The experiments are conducted on a workstation with Intel(R) Xeon E5645 2.40GHz CPU, and 32GB DDR3 RAM. The input of our program includes a volume domain, its associated frame field, and also the target edge length l^* of the cubic lattice defined in the isotropic space.

Frame Field: Frame fields are given as an input. Several existing state-of-the-art algorithms can be used to generate a high-quality cross field for any arbitrary volumetric domain (Huang et al., 2011; Ray et al., 2016; Gao et al., 2017; Solomon et al., 2017). For the convenience, we denote the discrete cross field as $\mathbf{D}$. We also test our methods with some user-designed frame fields, e.g., rotation along y-axis on torus and highly anisotropic frame fields on cubes.

Quality Metrics: To calculate the quality of anisotropic tetrahedral meshes, we first transform the elements τ from anisotropic space to isotropic space τ'. Many anisotropic mesh quality metrics are discussed in (Shewchuk, 2002b). We measure the quality by *dihedral angles* θ, *edge-radius ratio* $\rho = \sqrt{6}e_{min}/4r_{circ}$, and *condition* $\kappa = 3\sqrt{6}v_{\tau'}/\left(2l_{rms} * A_{rms}\right)$, where r_{circ} is the circumradius, e_{min} is the shortest edge length, $v_{\tau'}$ is the volume, l_{rms} and A_{rms} are the root mean square of edge lengths and face areas of a tetrahedron. When τ' is a regular tetrahedron, $\rho_{opt} = 1$, $\kappa_{opt} = 1$. We report histograms and minimum, average and standard deviation for θ_{min}, ρ and κ, denoted as θ_{min}, $\overline{\theta_{min}}$, $\sigma(\theta min)$, ρ_{min}, $\overline{\rho}$, $\sigma(\rho)$, κ_{min}, $\overline{\kappa}$, and $\sigma(\kappa)$. θ_{min} and θ_{max} are the smallest and the largest dihedral angles of a tetrahedron, respectively.

Alignment Error: The alignment quality ϵ is evaluated on the resulting meshes. For each edge $\mathbf{v}_{ij}$ of the resulting tetrahedral mesh, we first transform it to the isotropic space $\mathbf{v}'_{ij} = \mathbf{B}_{ij}\mathbf{v}_{ij}$, then the smallest angle between $\mathbf{v}'_{ij}$ and vectors in **Onering** is used to measure the alignment error of edge $\mathbf{v}_{ij}$, i.e., $\epsilon = \min_{k=1}^{N_{or}} \arccos\left(\frac{\mathbf{v}'^{\top}_{ij}\cdot\mathbf{Onering}(k)}{\|\mathbf{v}'_{ij}\|\cdot\|\mathbf{Onering}(k)\|}\right)$. Histograms, mean $\overline{\epsilon}$ and $\sigma(\epsilon)$ are reported for the result meshes.

Experiment Parameters: From the experiments, we find that setting σ in the range $[0.25l_{closest}, 0.35l_{closest}]$ has the similar performance, so we use $0.3l_{closest}$ for all the experiments. The neighbor radius R is set as $1.3l^{*}$, which includes all the one-ring neighbors. The Gaussian Hole Kernel definition in Eq. (4.2) takes the inverse of N_{or} as the weight for the negative Gaussian kernels. This is to balance the force used to push particles away and the forces to drag particles to holes. Larger weight will result in more coinciding particles, while with smaller weight, particles are more evenly pushed away but may be more off the desired one-ring structure. However, after a few rounds of optimization with deletion and insertion scheme, the performance of different weights are similar. The other parameters for deletion, insertion and projection of particles are given in the Sec. 4.2.2. All the experiments are conducted with the same parameters.



(a) Particle2013 (b) BCC $\mathbf{T} = \mathbf{I}$ (c) FCC $\mathbf{T} = \mathbf{I}$ (d) BCC $\mathbf{T} = \mathbf{D}$ (e) FCC $\mathbf{T} = \mathbf{D}$

Figure 4.3: Comparison with the traditional Gaussian kernel method proposed in Particle2013 (Zhong et al., 2013). The first row is the result on surface. The second shows the tetrahedra with $\theta_{min} < 40°$. The third row shows the clipping views. The following rows are histograms of dihedral angles, edge radius ratio, condition, and alignment error.

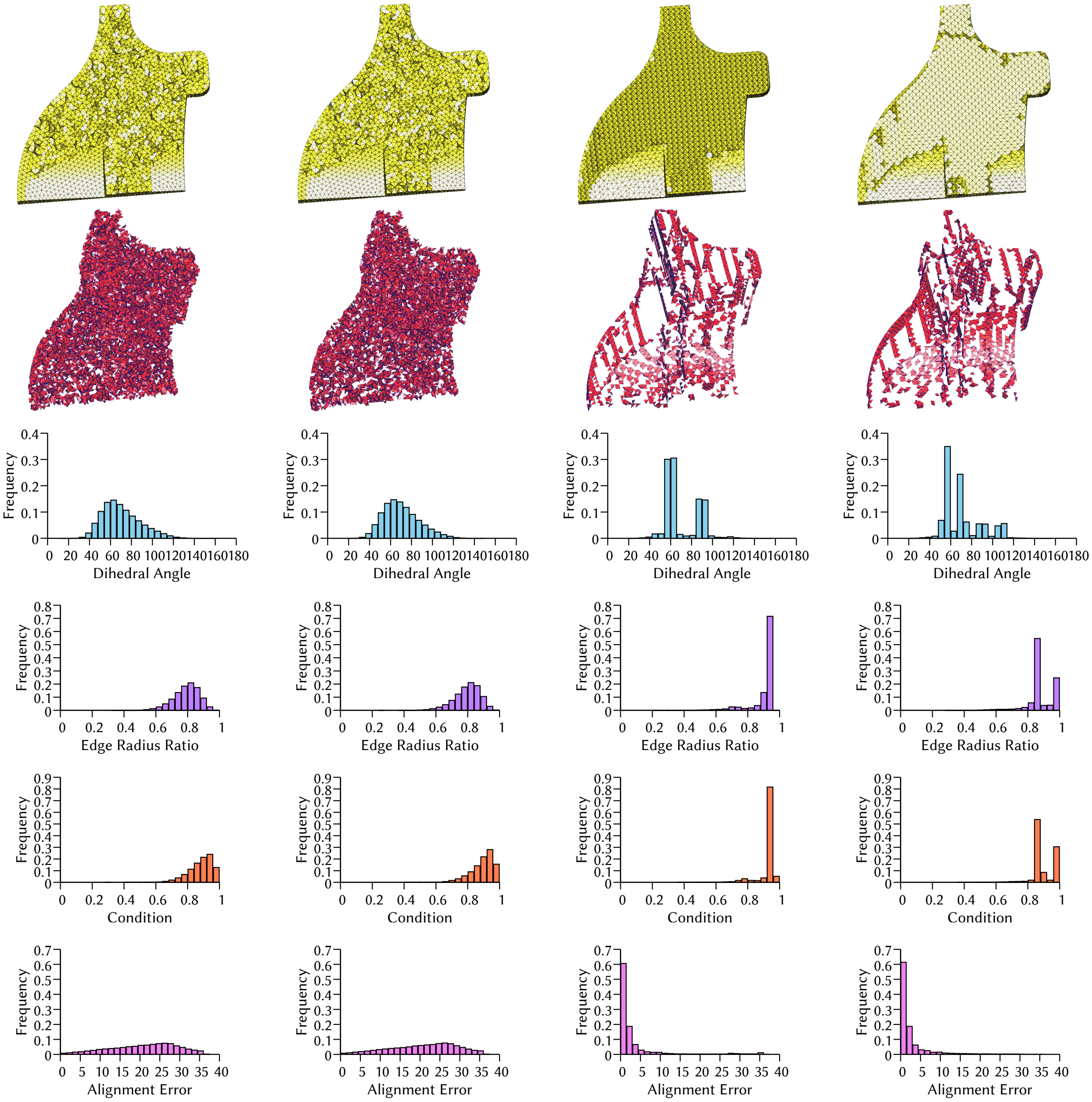
Comparison with Particle2013 (Zhong et al., 2013): Zhong et al.'s method (Zhong et al., 2013) used the traditional Gaussian kernel for generating anisotropic triangular mesh, and can be trivially extended for tetrahedral mesh generation. We compare our Gaussian hole kernel methods, named *BCC* and *FCC*, with their method, named *Particle2013*, on three models as shown in Figure 4.3 and Table 4.1. Particle2013 cannot achieve field-aligned meshing results for discrete cross fields. The experiments of our method use both $\mathbf{T} = \mathbf{I}$ and $\mathbf{T} = \mathbf{D}$ (discrete cross fields). Under either rotation field, our method achieves higher quality, e.g., about 3°to 5°growth on $\overline{\theta_{min}}$, 0.05 to 0.11 gain on $\overline{rho}$ and thousands less of tetrahedra in $\#T_{<40^\circ}$. Those gains are coming from the lattice-guided alignment, producing high quality BCC and FCC tetrahedra.

BCC alignment provides higher $\overline{\rho}$ and $\overline{\kappa}$ than FCC alignment. FCC alignment creates higher $\overline{\theta_{min}}$ but also higher $\overline{\theta_{max}}$ than BCC alignment. FCC alignment has smaller $\#T_{<40^\circ}$ and it also has smaller alignment error. Unlike identity field $\mathbf{I}$, a discrete cross field usually contains singularities inside the volume domain. The upper right corner of Figure 4.3 shows the singularities of $\mathbf{D}$ inside the Fertility model. Alignment to $\mathbf{D}$ is harder than alignment to $\mathbf{I}$, which explains the slight decline in quality. The advantage of rotation alignment will be illustrated in the later part.

Comparisons with CVT and ODT: We also compare with the CVT and ODT methods to see whether BCC and FCC alignment will improve the mesh qualities. We use their implementations in CGAL (CGAL, 2017; Jamin et al., 2015) for the comparison. Since both CVT and ODT energies do not support field-alignment in tetrahedral meshing, we set the frame field as $\mathbf{I}$ for our method. Our particle-based optimization well generates the BCC and FCC patterns as shown in Figure 4.4 and Figure 4.5. The dihedral angle histograms of BCC results have two peeks around 60°and 90°, and one peak in the histograms of ρ and κ. In FCC results, the histograms of ρ and κ have two peeks: the right ones are caused by regular tetrahedra in the meshes and the left ones are the other tetrahedra with dihedral htbp

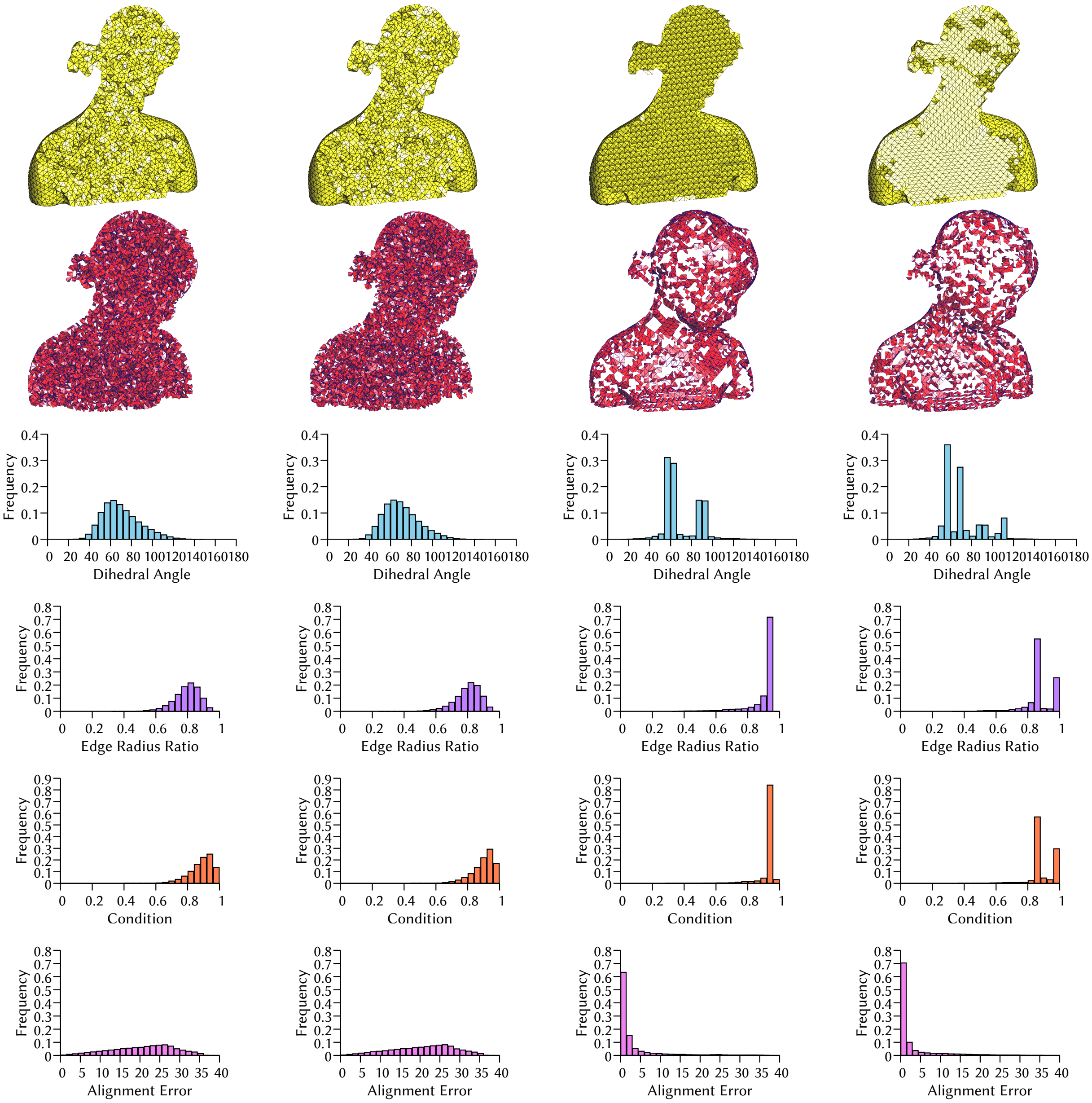
Table 4.1: The mesh quality in comparison with Particle2013 (Zhong et al., 2013). #V and #T are the numbers of vertices and tetrahedra in the result meshes. $\#T_{<20^\circ}$ and $\#T_{<40^\circ}$ are the numbers of tetrahedra with $\theta_{min} < 20^\circ$ and $\theta_{min} < 40^\circ$, respectively. The minimum/maximum, mean, and standard deviation of smallest dihedral angle θ_{min}, largest dihedral angle θ_{max}, edge radius ratio ρ, condition κ are provided. The mean and standard deviation of alignment error ϵ are also listed. Note that the best values are highlighted in bold for each group.

Model	**T**	Alg.	#V	$\theta_{min}/\overline{\theta_{min}}/\sigma(\theta_{min})$	$\theta_{max}/\overline{\theta_{max}}/\sigma(\theta_{max})$	$\rho_{min}/\overline{\rho}/\sigma(\rho)$	$\kappa_{min}/\overline{\kappa}/\sigma(\kappa)$	$\overline{\varepsilon}/\sigma(\varepsilon)$	$\#T_{<20^\circ}$	$\#T_{<40^\circ}$	#T	Time(s)
Bunny	n/a	Particle2013	60,000	15.73/52.93/7.00	156.62/94.90/11.65	**0.39**/0.82/0.07	0.31/0.89/0.07	20.88/7.77	3	13,359	319,562	143.60
Bunny	**I**	BCC	59,504	16.29/57.55/4.61	159.50/**91.93**/6.35	0.30/**0.91**/0.07	0.30/**0.94**/0.05	2.61/4.86	18	5,944	331,845	**55.52**
Bunny	**I**	FCC	57,533	17.98/**58.06**/7.83	157.45/96.64/16.68	0.35/0.89/0.08	0.33/0.90/0.07	**2.25**/4.35	2	5,143	318,470	94.84
Bunny	**D**	BCC	61,836	**18.72**/55.83/5.21	**148.07**/93.32/7.79	0.33/0.88/0.08	**0.42**/0.92/0.06	5.04/6.45	**1**	7,933	348,522	65.70
Bunny	**D**	FCC	60,324	17.50/56.36/7.19	153.16/96.27/14.75	0.31/0.87/0.07	0.35/0.91/0.06	4.20/4.43	2	**4,510**	336,753	137.79
Fertility	n/a	Particle2013	40,002	18.88/52.58/7.12	152.62/95.20/11.71	0.31/0.81/0.08	0.38/0.89/0.07	20.94/7.71	1	10,020	202,173	**33.77**
Fertility	**I**	BCC	41,188	16.86/56.59/5.88	158.44/**93.02**/8.25	0.35/**0.89**/0.09	0.32/**0.93**/0.07	3.70/6.07	26	7,286	218,319	40.60
Fertility	**I**	FCC	39,594	15.30/**57.09**/8.24	157.75/96.94/16.37	0.37/0.87/0.08	0.32/0.90/0.08	**3.29**/5.25	5	6,241	207,638	61.02
Fertility	**D**	BCC	42,594	17.91/55.23/5.97	154.92/94.30/9.18	0.30/0.87/0.09	0.35/0.92/0.07	6.05/7.52	6	7,664	229,902	51.88
Fertility	**D**	FCC	42,015	**22.17**/56.13/7.28	**144.17**/96.27/14.52	**0.38**/0.86/0.08	**0.45**/0.90/0.07	4.72/4.95	**0**	**3,697**	223,298	75.16
Kitten	n/a	Particle2013	60,000	15.63/52.47/7.90	158.25/95.15/11.65	0.32/0.80/0.10	0.31/0.88/0.08	20.83/7.78	28	23,988	305,286	92.65
Kitten	**I**	BCC	59,913	14.53/57.31/4.75	164.59/**92.02**/6.55	0.27/**0.91**/0.07	0.24/**0.93**/0.05	2.82/4.83	40	6,335	335,237	**71.28**
Kitten	**I**	FCC	56,499	16.46/**58.30**/8.13	156.81/96.84/17.37	0.35/0.89/0.08	0.32/0.90/0.07	**1.90**/4.37	8	5,430	313,779	93.27
Kitten	**D**	BCC	62,124	20.86/55.65/5.21	150.06/93.38/7.85	0.31/0.88/0.07	0.41/0.92/0.06	5.08/6.40	**0**	7,815	351,849	71.80
Kitten	**D**	FCC	59,998	**22.47**/56.31/7.24	**147.43**/96.33/14.76	0.30/0.87/0.07	**0.43**/0.91/0.06	4.21/4.40	**0**	**4,491**	336,382	135.46



(a) CVT (b) ODT (c) BCC (d) FCC

Figure 4.4: Comparisons with CVT and ODT on Fandisk model. The first row shows the clipping views. The second row shows the tetrahedra with $\theta_{min} < 40°$. The following rows are histograms of dihedral angles, edge radius ratio, condition, and alignment error.



(a) CVT (b) ODT (c) BCC (d) FCC

Figure 4.5: Comparisons with CVT and ODT on Bimba model. The red tetrahedra shown in the second row have $\theta_{min} < 40°$.

Table 4.2: Comparison with CVT and ODT. #V and #T are the numbers of vertices and tetrahedra in the output meshes. The mean value of smallest dihedral angle θ_{min}, largest dihedral angle θ_{max}, edge radius ratio ρ, condition κ, and alignment error ϵ are provided. The computation time is provided, which is the total time including particle optimization and mesh generation. Note that the best values are highlighted in bold for each group.

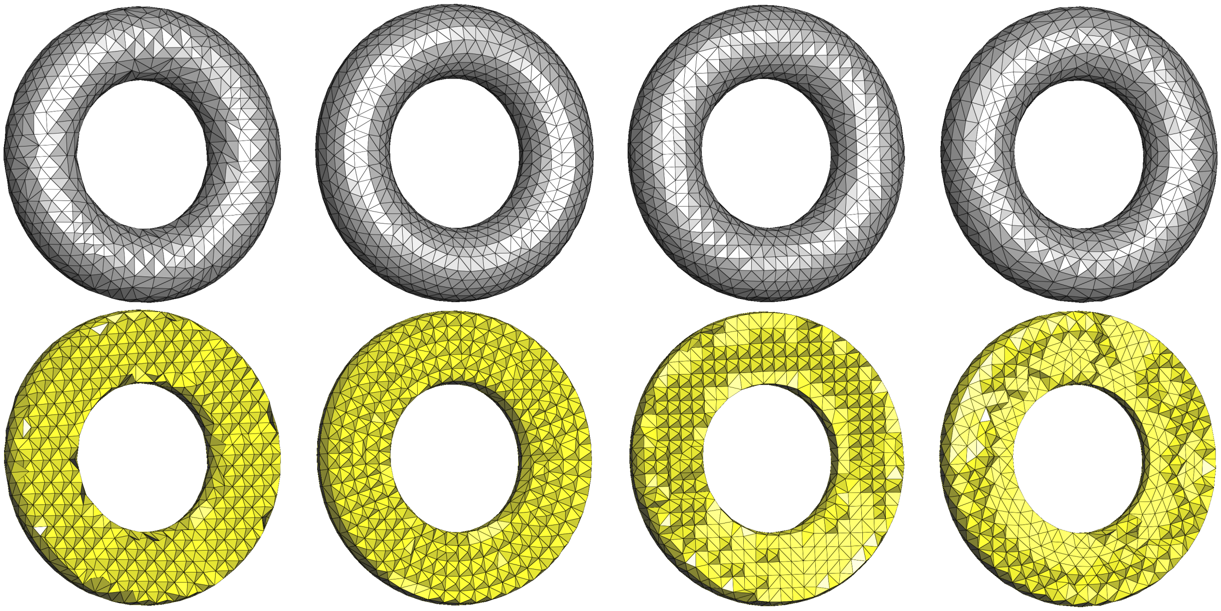
Model	Alg.	#V	$\theta_{min}/\overline{\theta_{min}}$	$\theta_{max}/\overline{\theta_{max}}$	$\overline{\rho}$	$\overline{\kappa}$	$\overline{\varepsilon}$	$\#T_{<20^\circ}$	$\#T_{<40^\circ}$	#T	Time(s)
Bimba	CVT	20,000	0.41/50.20	179.34/98.12	0.81	0.86	20.75	2,002	14,273	109,806	58.67
Bimba	ODT	20,000	2.69/51.36	175.47/93.97	0.81	0.89	20.78	543	9,757	107,400	39.34
Bimba	BCC	20,042	16.48/57.22	159.03/**92.39**	**0.90**	**0.93**	3.10	15	2,789	108,168	**20.66**
Bimba	FCC	19,477	**18.02/57.56**	**153.06**/96.94	0.88	0.90	**2.75**	**4**	**2,715**	104,340	48.03
Cube	CVT	10,000	0.50/50.05	179.22/98.32	0.81	0.86	21.01	1,037	7,311	54,221	28.34
Cube	ODT	10,000	2.06/51.19	176.65/94.21	0.81	0.89	21.02	315	5,326	52,975	18.32
Cube	BCC	9,009	44.48/59.01	120.61/**91.86**	**0.93**	**0.94**	1.00	**0**	**0**	49,152	**10.06**
Cube	FCC	9,842	**50.79/59.89**	**111.29**/96.49	0.91	0.91	**0.10**	**0**	**0**	52,728	28.61
Elephant	CVT	10,000	0.80/50.85	178.41/97.43	0.81	0.87	20.88	823	5,932	49,596	27.11
Elephant	ODT	10,000	0.80/50.99	178.41/94.53	0.80	0.88	20.85	359	5,003	48,801	16.24
Elephant	BCC	10,389	16.27/55.87	158.97/**93.46**	**0.88**	**0.92**	4.45	14	2,209	51,579	**11.23**
Elephant	FCC	10,050	**16.88/56.30**	**158.46**/97.31	0.86	0.89	**4.17**	**5**	**1,929**	49,299	21.86
Fandisk	CVT	18,000	0.49/49.92	179.22/98.54	0.81	0.86	20.92	1,876	12,939	94,977	51.24
Fandisk	ODT	18,000	3.72/50.99	173.48/94.60	0.81	0.88	20.89	639	9,172	92,939	34.00
Fandisk	BCC	18,518	**18.09**/57.25	155.71/**92.80**	**0.90**	**0.93**	3.20	**2**	**1,809**	99,005	**16.39**
Fandisk	FCC	17,920	18.08/**57.47**	**151.07**/96.70	0.88	0.90	**2.59**	3	2,077	94,509	25.50

angle $[54.735^\circ(4), 90^\circ, 109.47^\circ]$, which also explains the peeks in the dihedral angle histogram. The tetrahedra with $\theta_{min} < 40^\circ$ are on boundary surfaces. The more detailed quality statistics are given in Table 4.2. Our methods have about 6°to 10°growth on $\overline{\theta_{min}}$, 0.05 to 0.12 increment on $\overline{\rho}$ and much less elements with $\theta_{min} < 40^\circ$. Besides that, our optimization is faster since there is no computation of Voronoi diagram or connectivity in each iteration.

Rotation Field Alignment for Improving Mesh Quality: After showing our better mesh quality results as compared to Particle2013, CVT, and ODT methods, we would like to show that for some models with rotational features, the alignment with its rotation field can produce meshes with better quality. We use two models for such illustration: Torus (Figure 4.6) and Fancyring (Figure 4.7). The frame field we tried on Torus is the rotation along y-axis $\mathbf{R}_y$, and the frame field for Fancyring is a discrete cross field generated by (Huang et al., 2011), denoted as $\mathbf{D}$. We compare them with the results generated by identity field

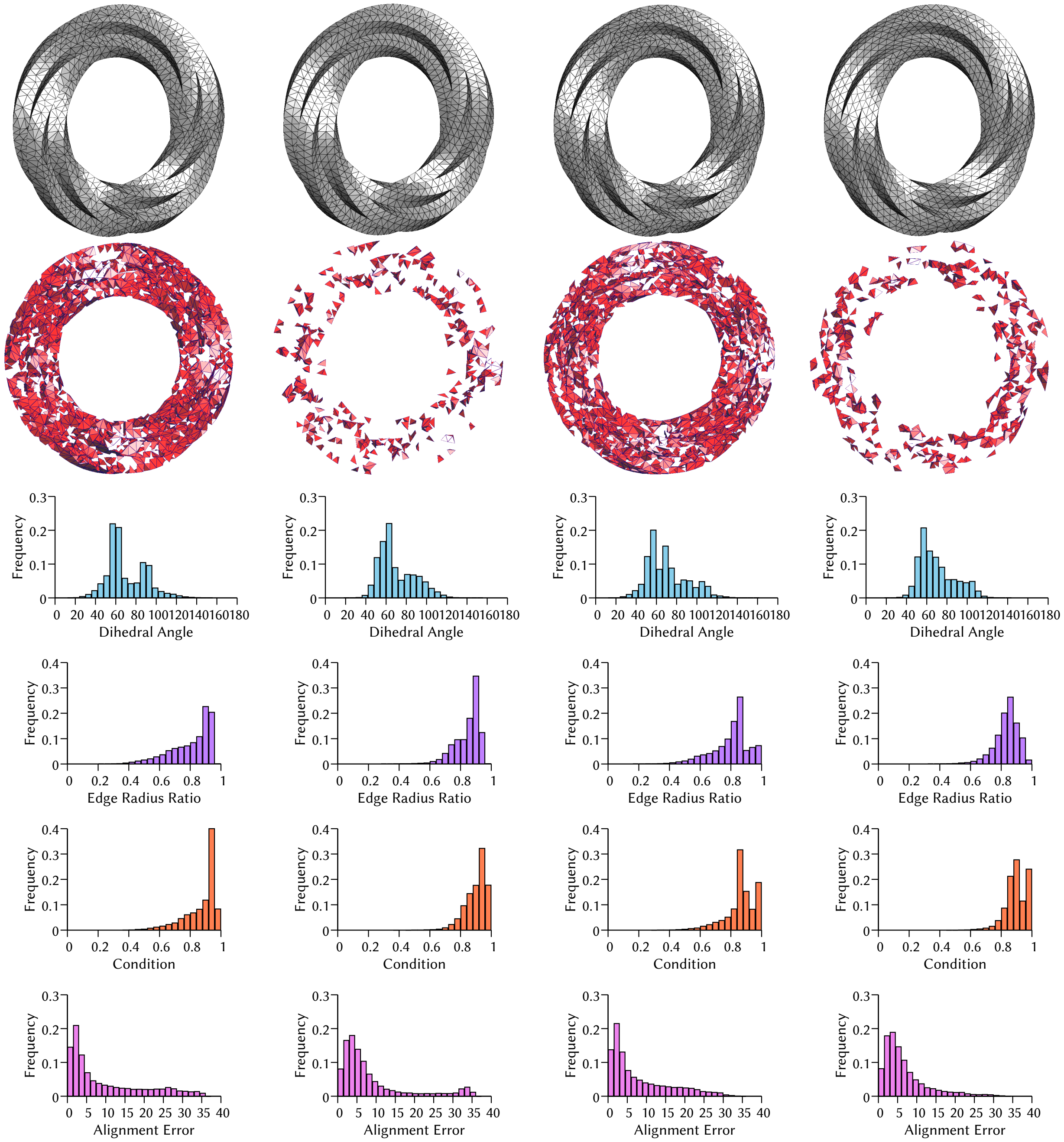
Table 4.3: The quality statistics of rotation alignment experiments on Torus and Fancyring. #V and #T are the numbers of vertices and tetrahedra of the result meshes. $\#T_{<20^\circ}$ and $\#T_{<40^\circ}$ are the numbers of tetrahedra with $\theta_{min} < 20^\circ$ and $\theta_{min} < 40^\circ$, respectively. The minimum, mean, and standard deviation of smallest dihedral angle θ_{min}, edge radius ratio ρ, condition κ are provided. The mean and standard deviation of ϵ are also listed. Dist is the Hausdorff distance between the boundary surfaces of the result and input meshes. Note that the best values are highlighted in bold for each group.

Model	T	Alg.	#V	$\theta_{min}/\overline{\theta_{min}}/\sigma(\theta_{min})$	$\rho_{min}/\overline{\rho}/\sigma(\rho)$	$\kappa_{min}/\overline{\kappa}/\sigma(\kappa)$	$\overline{\varepsilon}/\sigma(\varepsilon)$	$\#T_{<20^\circ}$	$\#T_{<40^\circ}$	#T	Dist	Time(s)
Torus	$\mathbf{I}$	BCC	2,274	18.70/54.06/7.78	0.37/0.85/0.11	0.34/0.90/0.09	6.02/7.65	5	798	10,605	0.500	6.98
Torus	$\mathbf{I}$	FCC	2,299	15.62/**55.37**/8.70	0.34/0.85/0.09	0.30/0.89/0.08	**4.98**/6.37	1	541	10,486	0.341	**6.66**
Torus	$\mathbf{R}_y$	BCC	2,440	**28.10**/55.05/6.55	**0.43**/**0.86**/0.09	**0.55**/**0.91**/0.07	5.69/6.97	**0**	**520**	11,389	0.377	7.91
Torus	$\mathbf{R}_y$	FCC	2,347	20.87/54.38/8.34	0.35/0.85/0.10	0.42/0.89/0.09	5.60/6.19	**0**	677	11,106	**0.318**	11.76
Fancyring	$\mathbf{I}$	BCC	4,775	17.22/51.76/8.68	0.23/0.81/0.13	0.37/0.88/0.10	9.66/9.72	14	2,587	21,420	0.542	**8.42**
Fancyring	$\mathbf{I}$	FCC	4,810	13.79/52.05/9.56	0.19/0.80/0.12	0.29/0.87/0.10	8.15/7.70	15	2,493	21,094	0.480	14.44
Fancyring	$\mathbf{D}$	BCC	4,187	**22.93**/**53.98**/6.40	0.29/**0.85**/0.08	0.37/**0.90**/0.07	8.55/8.84	**0**	**417**	18,992	**0.195**	21.56
Fancyring	$\mathbf{D}$	FCC	3,979	22.14/53.57/6.96	**0.33**/0.84/0.08	**0.48**/**0.90**/0.07	**6.76**/5.76	**0**	527	17,256	0.203	14.93



(a) BCC ($\mathbf{B} = \mathbf{I}$) (b) BCC ($\mathbf{B} = \mathbf{R}_y$) (c) FCC ($\mathbf{B} = \mathbf{I}$) (d) FCC ($\mathbf{B} = \mathbf{R}_y$)

Figure 4.6: BCC and FCC experiments on Torus with $\mathbf{I}$ and $\mathbf{R}_y$ rotation fields. The yellow ones are the clipping views.



(a) BCC ($\mathbf{T} = \mathbf{I}$) (b) BCC ($\mathbf{T} = \mathbf{D}$) (c) FCC ($\mathbf{T} = \mathbf{I}$) (d) FCC ($\mathbf{T} = \mathbf{D}$)

Figure 4.7: BCC and FCC experiments on Fancyring with different rotation fields. The red ones are the tetrahedra with $\theta_{min} < 40°$.

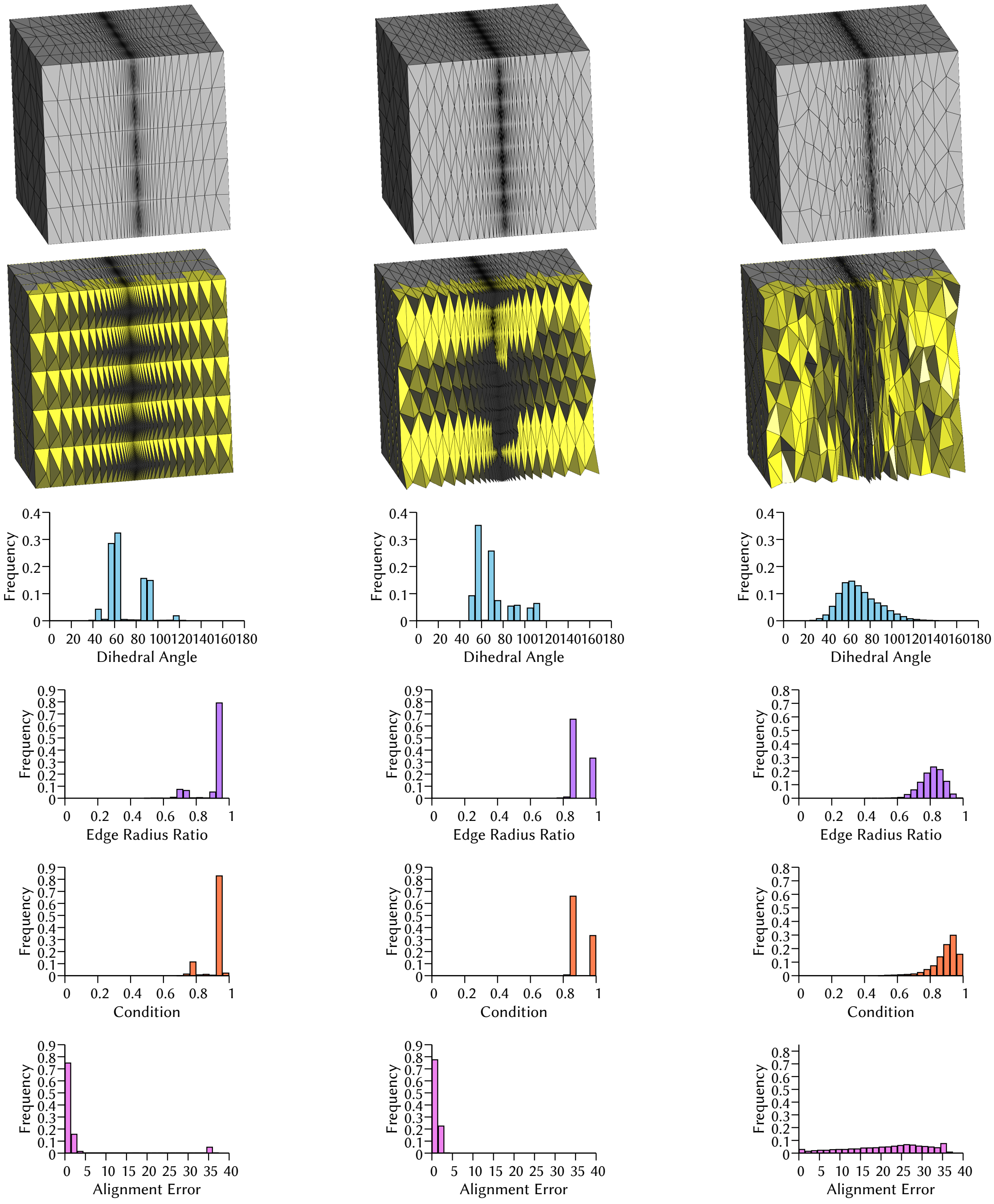
Table 4.4: Statistics of mesh quality and time consumption compared with LCT (Fu et al., 2014). #V and #T are the numbers of vertices and tetrahedra of the result meshes. $\#T_{<20^\circ}$ and $\#T_{<40^\circ}$ are the numbers of tetrahedra with $\theta_{min} < 20^\circ$ and $\theta_{min} < 40^\circ$, respectively. The mean of smallest dihedral angle θ_{min}, edge radius ratio ρ, condition κ, and computation time are provided. Note that the best values are highlighted in bold for each group.

Model	Alg.	#V	θ_{min}	$\overline{\theta_{min}}$	$\overline{\rho}$	$\overline{\kappa}$	$\overline{\varepsilon}$	$\#T_{<20^\circ}$	$\#T_{<40^\circ}$	#T	Time(s)
Figure 4.8	LCT	1,869	24.84	51.71	0.81	0.89	21.59	**0**	585	8,117	**8.3**
	BCC	2,476	36.88	56.95	**0.90**	**0.92**	3.09	**0**	**0**	11,988	16.28
	FCC	2,471	**52.85**	**58.75**	**0.90**	0.91	**0.78**	**0**	**0**	11,520	31.13
Figure 4.9	LCT	6,338	15.51	49.87	0.78	0.88	20.94	88	4,220	31,840	72.6
	BCC	6,410	14.26	53.73	**0.85**	0.89	5.37	8	2,297	33,099	**56.05**
	FCC	6,498	**23.44**	**55.27**	**0.85**	**0.90**	**3.84**	**0**	**632**	32,993	87.59

I. It can be seen from these two experiments: if the rotation field aligns very well with the shape or the features of the geometry, we can get better shape approximation as well as higher tetrahedral qualities as shown in Table 4.3.

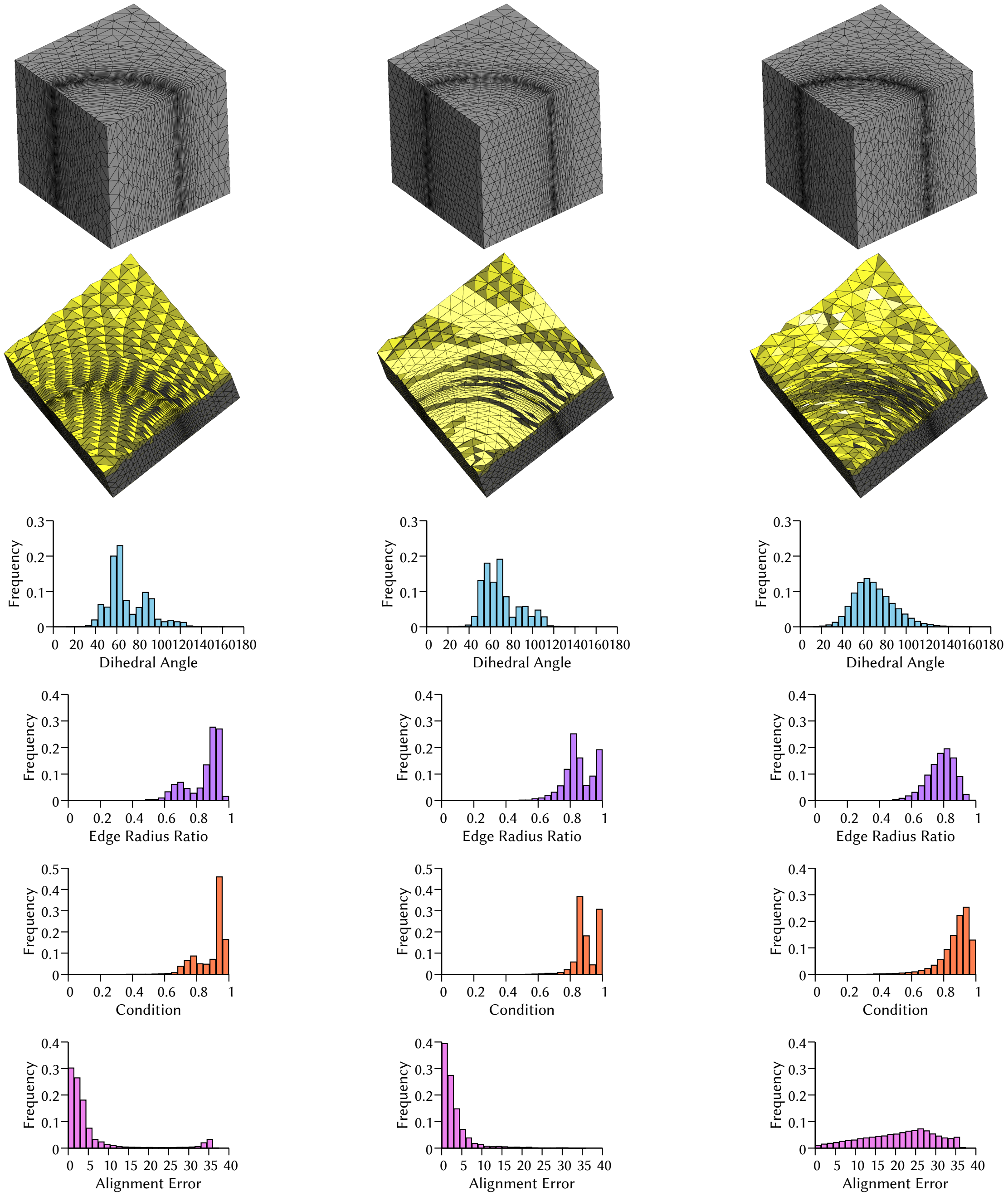
Field Alignment for Anisotropic Tetrahedral Meshing: To further explore the performance of our field-aligned and lattice-guided methods, we conduct experiments using the highly anisotropic fields on Cube, and compare with LCT method (Fu et al., 2014) as shown in Figure 4.8 and Figure 4.9. Compared with LCT, our BCC and FCC results show higher quality due to the strong directional control and the advantage of lattice-alignment. The detailed quality statistics are given in Table 4.4.

Robustness: We demonstrate the robustness of our method by experiments on Teddy with different numbers of vertices on the same discrete frame field. The minimal dihedral angle $\overline{\theta_{min}}$, edge radius ratio $\overline{\rho}$, condition $\overline{\kappa}$, alignment error $\overline{\epsilon}$ of the resulting meshes, and their optimization time along with the different vertex numbers are shown in Figure 4.10. The results of BCC and FCC are shown in red and blue curves, respectively. The convergence of our particle optimization scheme is tested on Kitten Model with 20 rounds of LBFGS optimization as shown in 4.2. The energy decreases the most at the first few rounds. Due to particle deletion and insertion scheme, the increase of energy at the beginning of the round is due to the particle deletion and insertion scheme. The number of vertices being inserted



(a) BCC (b) FCC (c) LCT

Figure 4.8: Anisotropy variation along a single direction on Cube $[0.1, 1.1]^3$. The inverse of frame field is defined as $\mathbf{B} = diag\left(\left(1.0125 - e^{-|x-0.6|}\right)^{-1}, 1, 1\right)$. The second row shows the clipping views of the result tetrahedral meshes. The last four rows show the histograms of dihedral angle, edge radius ratio, condition, and alignment error.



(a) BCC (b) FCC (c) LCT

Figure 4.9: Cylindrical anisotropy on Cube $[1, 11]^3$. The inverse of frame field is defined as $\mathbf{B} = \mathbf{S} * \mathbf{R}$, where $\mathbf{S} = diag\left(\left(1.05 - e^{-0.01|x^2+y^2-49|}\right)^{-1}, 1, 1\right)$, and the three columns of $\mathbf{R}$ are $\left(-x/\sqrt{x^2+y^2}, y/\sqrt{x^2+y^2}, 0\right)$, $\left(-y/\sqrt{x^2+y^2}, x/\sqrt{x^2+y^2}, 0\right)$, and $(0, 0, 1)$, respectively. The second row is the clipping view of the result tetrahedral meshes. The last four rows show the histograms of dihedral angle, edge radius ratio, condition, and alignment error.

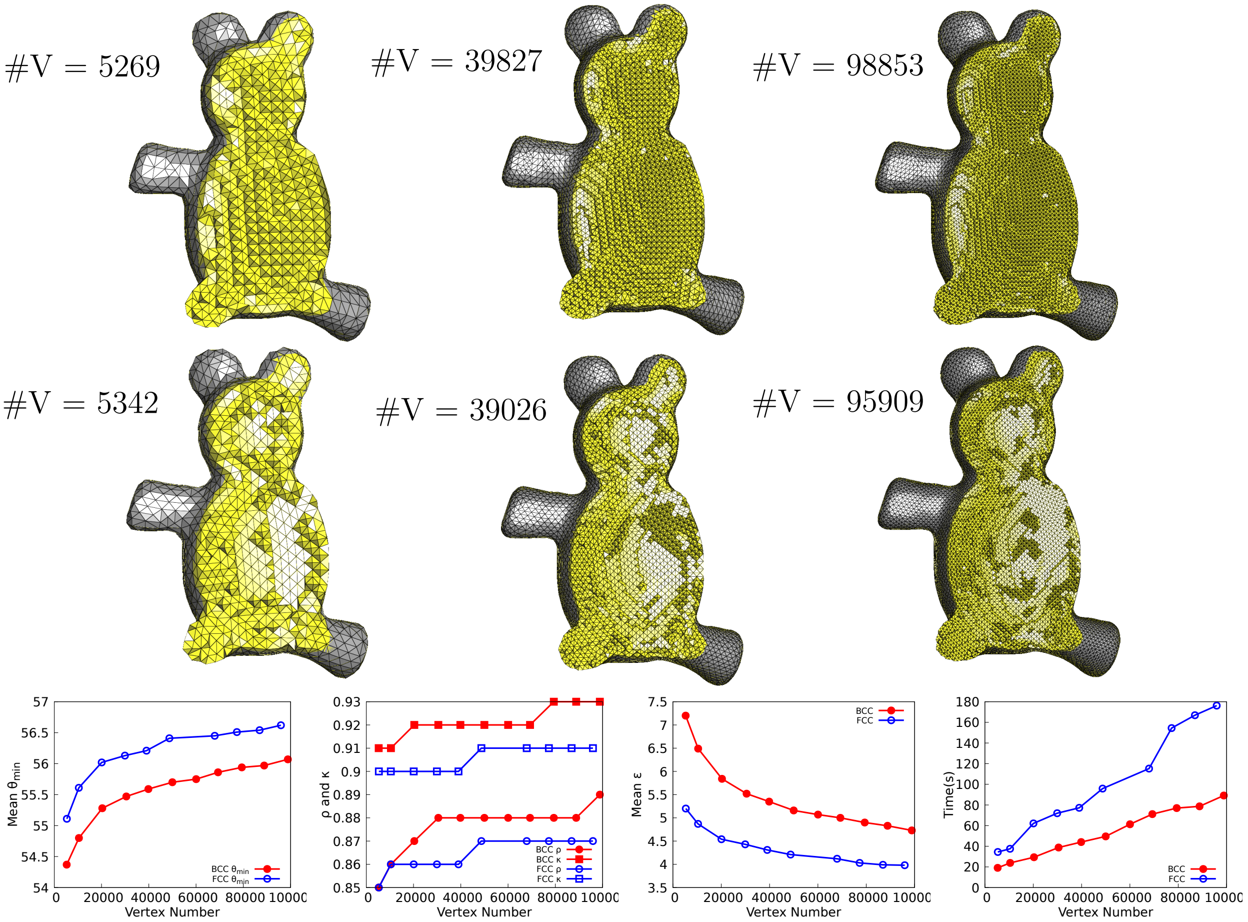


Figure 4.10: The first two rows shows BCC and FCC results in different vertex numbers. The next two rows show the quality statistics and time consumption with different numbers of vertices on Teddy.

and deleted are given. At the end of each round, the energy is optimized to be smaller than the previous round.

4.3.4 Discussion and Future Work

It should be noted that our method can be easily extended to solve field-aligned anisotropic triangular meshing for surfaces. This can be achieved by defining six *holes* in the GHK of Eq. (4.1) on the tangent plane of surface. As shown in Figure 4.11, our method can obtain better mesh quality of anisotropic triangular meshes compared to Particle2013 (Zhong et al.,

2013) and LCT (Fu et al., 2014). Here r_6 is the ratio of vertices with degree-6. The quality of a triangle is measured by $\xi = 4\sqrt{3}ap/h$, where a is its area, p is its perimeter and h is its longest edge length in its mapped isotropic space.

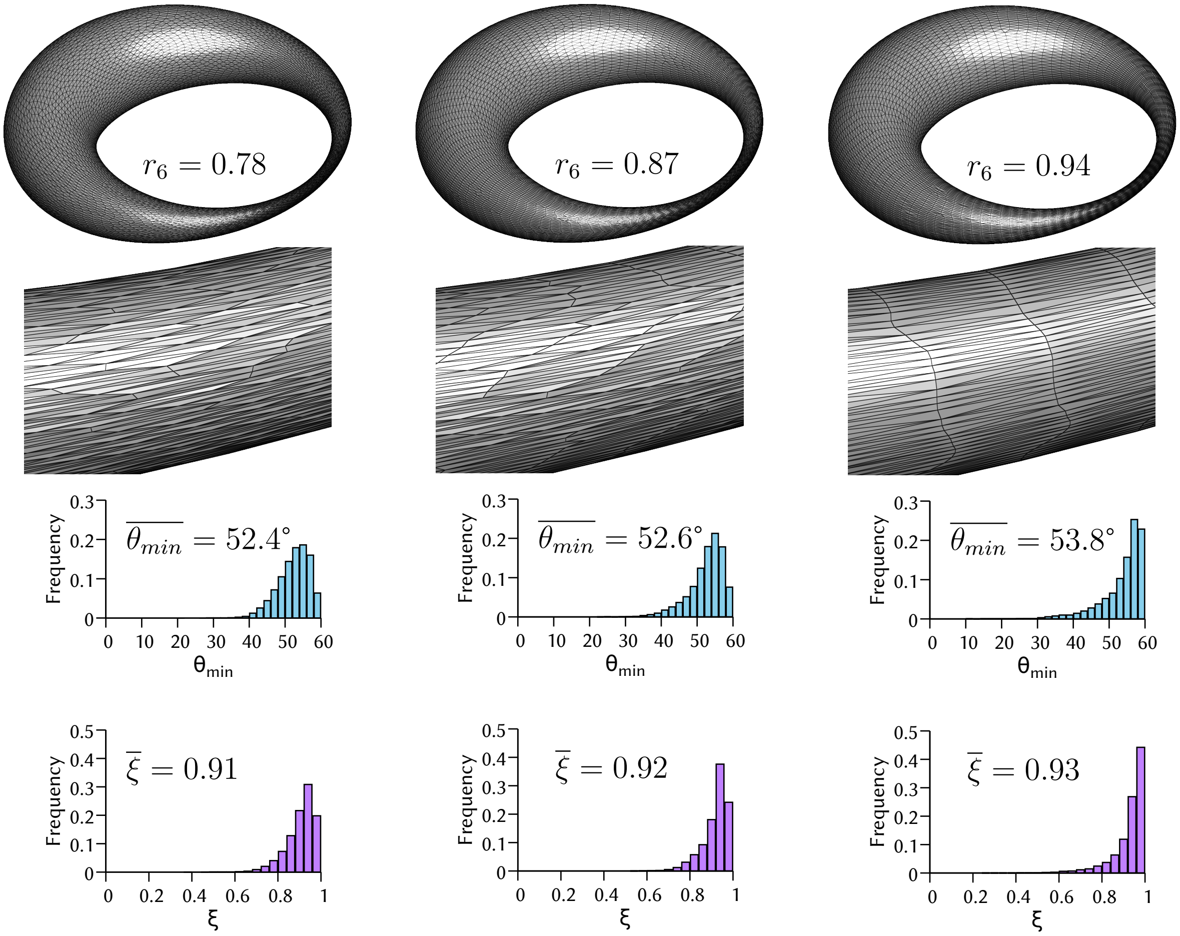
Figure 4.6 and Figure 4.7 show two examples that our rotation-field-aligned BCC and FCC methods might improve mesh quality for models having rotational shapes and features. In the future we would like to investigate in depth the relationship between the generation of frame-fields and the quality of field-aligned BCC and FCC meshes, in order to come up with some better field generation methods that are specifically tailored for such lattice meshes. In addition, we would like to investigate the possibility of other lattices (Du and Wang, 2005b), such as A15 and Z-type configurations, etc.

The lattice structure is only used to guide the particle optimization in this work. Building a tetrahedral mesh based on the one ring structure for a field-aligned particle set will be one of our future work. Unlike the isosurface stuffing work (Labelle and Shewchuk, 2007), which only need to cut BCC tetrahedra along the domain boundary, our situation is more complicated and undetermined. The incomplete lattice structure mays appear any place and in any form.

4.4 Hexahedral-Dominant Meshing

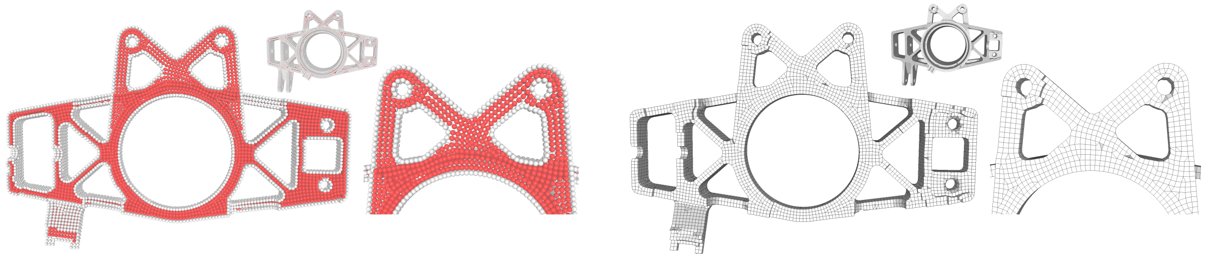
4.4.1 Overview

The state-of-the-art hexahedral-dominant meshing methods (Lévy and Liu, 2010; Botella et al., 2016; Baudouin et al., 2014; Sokolov et al., 2016; Gao et al., 2017), follow a three-step pipeline: (1) generating a frame field; (2) optimizing a set of vertices locally aligned with the frame field; (3) building a hexahedral-dominant mesh from the optimized vertices. High-quality frame field can be generated by some existing methods (Huang et al., 2011; Ray et al., 2016; Gao et al., 2017). Then, it is crucial to determine a set of optimized vertex positions. The quality of resulting point set directly affects the final mesh quality.



(a) Particle2013 (b) LCT (c) Our Method

Figure 4.11: Field-aligned anisotropic triangular meshing on Cyclide, compared with Particle2013 (Zhong et al., 2013) and LCT (Fu et al., 2014). The three columns are the results of Particle2013 method, LCT method, and our method, respectively. The first two rows show the resulting surface meshes and the zoom-in views of the narrow part. r_6 shown in the first row is the ratio of vertices with degree 6 in the result mesh. The last two rows are the histograms of $\overline{\theta_{min}}$ and $\overline{\xi}$.



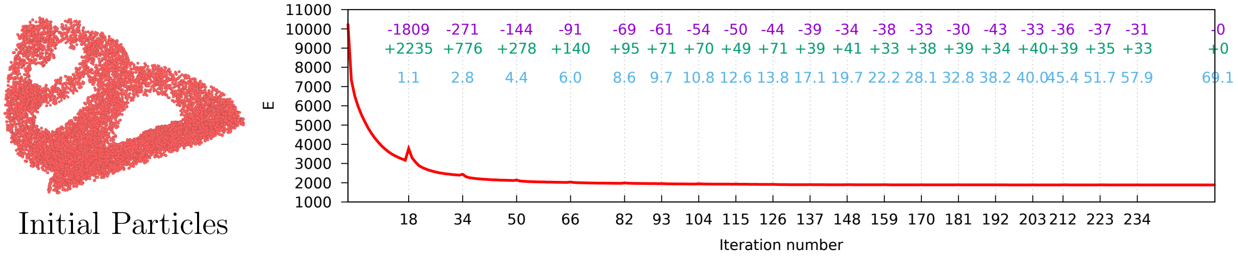
(a) Particles

(b) Hexahedral-dominant mesh

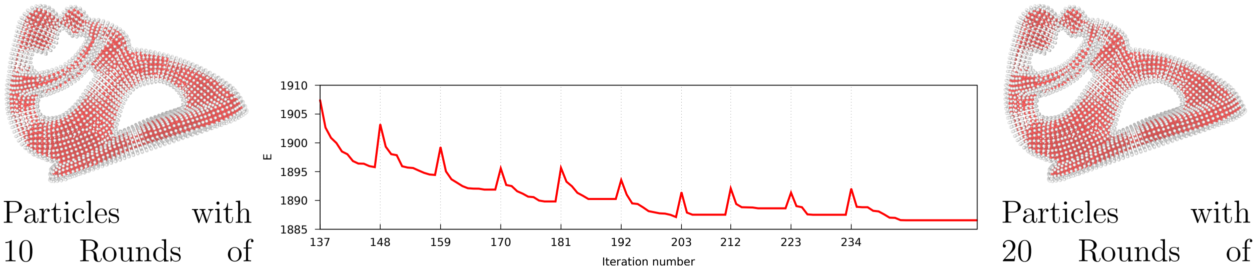
Figure 4.12: The particles and mesh results on Front_Upright_ASM005 Model. The three images on the left show the particle results in zoom-in views. The gray particles are on the domain boundary surface and the red ones inside the domain. The three images on the right show the corresponding hexahedral-dominant mesh in cut-out views.

We propose a novel particle-based variational method to achieve the vertex optimization in the second step. Our variational method is capable of handling complicated CAD models of large sizes, as well as volume domains bounded by freeform surfaces. It is fully automatic, robust to domain shape, and also supports various frame fields, which can be either highly-warped or having large stretching ratios. Finally, we apply the same connectivity method as used by PGP3D (Sokolov et al., 2016; Meshkat and Talmor, 2000) for the third step of generating hex elements from the optimized vertices. Figure 4.12 shows an example of our optimized particles and constructed hexahedral-dominant mesh for Front_Upright_ASM005 model. Our particle optimization is simple to understand and easy to implement. The resulting hexahedral-dominant meshes usually have higher percentage of hexahedra and better quality. The speed is also competitive.

To design a particle system, the main challenge is to define a potential energy whose minimization drives the particles to be aligned with the given frame field to form a desired hexahedral pattern. We design a special kernel to define the inter-particle energy, called *Gaussian hole kernel*, which is a summation of seven Gaussian kernels. For each particle, we place one positive Gaussian kernel exactly at its position and six negative Gaussian kernels at its six expected neighborhood positions. We call the six negative Gaussian kernels as “holes”, which are placed along the three vectors of the local frame field. Then we define the inter-particle energy as the sum of these Gaussian hole kernels for all the particles. We use a quasi-Newton L-BFGS algorithm to efficiently minimize this energy function to make the neighboring particles of each particle to fall into the positions of the specified holes which are aligned with the given frame field, thus forming a hexahedral pattern. The energy minimization curve on fertility model is given in Figure 4.13. We also proposed effective particle deletion and insertion schemes to further improve the regularity of the particle arrangement by reducing the number of non-hex elements.



Initial Particles



Particles with 10 Rounds of L-BFGS Optimization

Particles with 20 Rounds of L-BFGS Optimization

Figure 4.13: The 20 rounds of L-BFGS optimization on the Fertility model. The optimization starts from a random initialization of 10,000 particles. The purple numbers are the numbers of particles being deleted after each round. The green numbers are the numbers of particles being inserted after each round. The blue numbers are the computation time in seconds. The second curve shows the zoom-in view of the last 10 rounds. The energy raise at the beginning of each round is caused by two reasons: one is particle insertion and deletion; the other one is particle projection.

4.4.2 Mesh Generation

The final output hexahedral-dominant mesh is generated by the same method as in PGP3D (Sokolov et al., 2016). The mesh generation has three steps: 1) remesh the boundary using the resulting point set; 2) build the tetrahedral mesh; 3) merge the tetrahedral mesh into a hexahedral-dominant mesh.

In the first step, Restricted Voronoi Diagram (RVD) (Yan et al., 2009) is utilized to extract the boundary of the domain from the resulting point set. We use the RVD implementation in GEOGRAM (Lévy, 2015). Sometimes the remeshed boundary is not dense enough to capture all the geometric features. A mesh refinement step is performed by inserting more vertices to the remeshed boundary until the Hausdorff distance between the refined mesh boundary and input domain boundary is less than a certain threshold. We follow the algorithm and use the same parameters introduced in PGP3D to do the mesh refinement. Once we get the refined boundary, restricted Delaunay tetrahedral mesh is built by adding the remaining domain-internal particles using TetGen (Si, 2015). TetGen constructs the tetrahedral mesh without considering the frame field. When the frame field has scale variance, the output tetrahedral mesh may not be a Delaunay tetrahedral mesh with respect to the frame field. It can be transformed to a Delaunay tetrahedral mesh by performing a set of topological flipping operations (Shewchuk, 2002a). In all of our experiments shown in this paper, we did not find any improvements by such topological operations. The last step is to merge the tetrahedral mesh to a hexahedral-dominant mesh. It is carried out in the following two steps.

The first step is to find out all the possible combinations of tetrahedra that form hexahedron, prisms and pyramids. There are 6 ways to merge 5 or 6 tetrahedra to a hexahedron (Meshkat and Talmor, 2000). Sokolov et al. (Sokolov et al., 2016) proved that these are all possible ways of combining 5 or 6 tetrahedra to a hexahedron. Besides that, they also provided 4 ways to merge 7 tetrahedra to a hexahedron, where one of the 7 tetrahedra is a sliver that prevents merging into two prisms. The implementation details of finding all hexahedron candidates are given in the Supplementary Appendix, which follows the algorithm introduced in (Meshkat and Talmor, 2000).

If we denote all possible combinations found in previous step as $C = \{c_1, c_2, \ldots, c_n\}$, where $c_i = \{t_1, t_2, \ldots, t_m\}$ is a combination of m tetrahedra that forms a hexahedron, prism, or pyramid, then the goal of the second step is to find a subset C' of C such that $\bigcup_{c \in C'} c$ covers the whole tetrahedral mesh, and there is no conflict among elements of C'. The most obvious example of conflicts is that $c_i \cap c_j \neq \emptyset$, i.e., they share a tetrahedron, then they are conflicting with each other. There are also some other conflicting conditions elaborated in Sokolov et al.'s work (Sokolov et al., 2016), which are all considered in our implementation. Since this optimization problem is similar to a weighted set cover problem, which has no efficient optimal solution, we use a greedy algorithm to find a good solution by considering firstly the hexahedron element with better quality. All possible hexahedral combinations in C are checked with descending order of their quality Q_{hex}, and add them to C' if they are not conflicting with combinations already in C'. After traversing through all hexahedron candidates, we work on the prisms and pyramids. Finally, the remaining tetrahedra are added to conclude the construction of hexahedral-dominant mesh.

The quality of hexahedral element Q_{hex} is combined by two parts, i.e., the planar quality Q_p of its six quad faces and the orthogonality Q_o of its eight corners. $Q_{hex}(h) = Q_p * \min_c Q_o(c)$. Every two connected edges of a quad face form a normal vector $\mathbf{n}_i$. The planar quality is defined as $Q_p = 1.0 - 2/\pi \arccos(\min_{0 \le i < j < 4} \mathbf{n}_i \cdot \mathbf{n}_j)$. Q_o is measured by the shape quality of the tetrahedron constructed by three edge vectors from a corner. $Q_o = \frac{12(3V_t)^{\frac{2}{3}}}{\sum_{0 \le i < j < 4} l_{ij}^2}$, where V_t is the volume of the tetrahedron and l_{ij} is the edge length of vertices i and j. V_t and l_{ij} are calculated in the imagined space. Higher Q_{hex} denotes higher quality of the hexahedron. In mesh generation, there is one more parameter Q_{bound}, which is set to avoid generating hexahedra with bad quality. Higher Q_{bound} leads to higher hex quality but lower hex volume ratio. It balances the hex quality and hex volume ratio of the resulting mesh. In the experiments, we show the relationship between those two qualities by setting different values of Q_{bound}.

The implementation details about finding all hexahedron candidates from the given tetrahedral mesh are shown in Alg. 8, which consider all 10 combinations discussed in PGP3D paper (Sokolov et al., 2016). We use $N(v)$ to indicate the set of direct neighboring vertices of v.

Algorithm 6: NextVertex: Find the vertex on the opposite side of a face.

```
Input: vertices $i$, $j$, $k$, $l$ of a tetrahedron
Output: vertex that forms a tetrahedron with $i$, $j$, $k$, and on the opposite side with $l$
$S \leftarrow N(i) \cap N(j) \cap N(k) \setminus \{l\}$;
if $\|S\| = 0$ then // $S$ may have 0 or 1 element
    return NONE;
else
    return $S[0]$;
end
```

Algorithm 7: FindRing: Find the ring of vertices around an edge.

```
Input: vertices $i$, $j$, $k$, $l$ of a tetrahedron, target number of vertices in the ring $n$
Output: $n$ vertices forming a ring around edge $i$, $j$
$V[0] \leftarrow k$;
$V[1] \leftarrow l$;
for $m \leftarrow 2$ to $n$ do
    $t \leftarrow NextVertex(i, j, V[m-1], V[m-2])$;
    if t = NONE then
        return $\emptyset$;
    else if $m = n$ and $t = V[0]$ then
        return $V$;
    else if $t = V[0]$ then
        return $\emptyset$;
    else
        $V[m] \leftarrow t$;
    end
end
```

Algorithm 8: FindHexahedrons: Find all hexahedron candidates.

```
Input: tetrahedral mesh
Output: set of all candidate hexahedrons
$S \leftarrow \emptyset$;
for every tetrahedron (a, b, c, d) do
    for every face (i, j, k) of tetrahedron (a, b, c, d) do
        for every vertex p in $N(i) \cap N(j) \setminus \{a, b, c, d\}$ do
            for every vertex q in $N(i) \cap N(k) \setminus \{a, b, c, d\}$ do
                for every vertex r in $N(j) \cap N(k) \setminus \{a, b, c, d\}$ do
                    for every vertex s in $N(p) \cap N(q) \cap N(r) \setminus \{a, b, c, d\}$ do
                        $S \leftarrow S \cup \{(a, b, c, d, p, q, r, s)\}$;
                    end
                end
            end
        end
    end
end
for every tetrahedron (a, b, c, d) do
    for every edge (i, j) of tetrahedron (a, b, c, d) do
        suppose $\{k, l\} \leftarrow \{a, b, c, d\} \setminus \{i, j\}$;
        $V \leftarrow FindRing(i, j, k, l, 6)$;
        if $V \neq \emptyset$ then
            $S \leftarrow S \cup \{(i, j, V[0], V[1], \ldots, V[5])\}$;
        end
    end
end
for every tetrahedron (a, b, c, d) do
    for every edge (i, j) of tetrahedron (a, b, c, d) do
        suppose $\{k, l\} \leftarrow \{a, b, c, d\} \setminus \{i, j\}$;
        $V \leftarrow FindRing(i, j, k, l, 4)$;
        if $V = \emptyset$ then
            break;
        end
        $U \leftarrow FindRing(k, l, i, j, 4)$;
        if $V = \emptyset$ then
            break;
        end
        $S \leftarrow S \cup \{(i, j, V[0], \ldots, V[3], U[0], \ldots, U[3])\}$;
    end
end
```

4.4.3 Experiment and Comparison

In this section, we present some implementation details and also compare our method with state-of-the-art methods. The experiments are conducted on a workstation with Intel(R) Xeon E5645 2.40GHz CPU, and 32GB DDR3 RAM.

The input of our algorithm includes: 1) a volume domain, defined by a closed surface or a tetrahedral mesh, 2) a frame field of the volume domain, and 3) an expected edge length l^* of output hexahedron element. Note that l^* is the edge length of a regular hexahedron in the imagined space if the frame field is defined by non-orthogonal or non-unit-length vectors based on l^*, the initial number of particles can be estimated by $N = V_d/(l^*)^3$, where V_d is the volume of domain in the imagined space.

Alignment error: Our particle-based optimization method is designed to align particles with the input frame field. A perfect alignment means the edges between each particle and its one ring neighbors align with the six vectors formed by the local frame field. To define a frame field independent metric, we first transform the edge vector between two neighbor particles by the local frame field, then we use the smallest angle θ_a between the edge vector to the six directions $(-1,0,0),(1,0,0),(0,1,0),(0,-1,0),(0,0,1),(0,0,-1)$ to measure the alignment error of an edge vector. The average alignment error $\overline{\theta_a}$ is used to measure the overall error of all edges.

Mesh quality: The quality of the resulting hexahedral-dominant meshes is measured in two aspects. One is the proportion of hexahedron elements including $H_{vol}\%$, the percentage of domain volume covered by hexahedral elements, and $\#Hex\%$, the percentage of hexahedral elements in number. We also provide the percentage of prisms, pyramids, and tetrahedra, denoted as $\#Prism\%, \#Pyr\%$, and $\#Tet\%$ respectively. Another one is the geometric quality of a hexahedral element, which is measured by Scaled Jacobian:

$$Q_{SJ} = \min_{i=1\ldots8} \left| \frac{\mathbf{v}'_{i1} \cdot (\mathbf{v}'_{i2} \times \mathbf{v}'_{i3})}{\|\mathbf{v}'_{i1}\|\|\mathbf{v}'_{i2}\|\|\mathbf{v}'_{i3}\|} \right|, \tag{4.5}$$

where $\mathbf{v}'_{ij}, 1 \leq j \leq 3$ are three edge vectors $\mathbf{v}_{ij}$ transformed to imagined space, that is $\mathbf{v}'_{ij} = \mathbf{B}_{ij}\mathbf{v}_{ij}$. Front propagation-based method (Baudouin et al., 2014), PGP3D (Sokolov et al., 2016) and Gao et al.'s method (Gao et al., 2017) use the same metric for evaluation. Furthermore, the Hausdorff distance between the result mesh and the domain boundary is used to measure the geometric approximation.

Input frame field: In our experiments, the input frame field is either defined analytically on some simple geometry shapes, or generated by methods of (Huang et al., 2011), (Ray et al., 2016) and (Gao et al., 2017), which are essentially cross fields. The analytically defined ones have closed-form definition in the whole domain. The other generated cross field is defined on a tetrahedral mesh. A cross field either defines a local frame field for each vertex (Ray et al., 2016) or for each tetrahedron (Huang et al., 2011). If the frame field is defined on vertices, we use the local frame field of the closest vertex to assign to a particle. If the frame field is defined on tetrahedra, we will find the tetrahedron that contains the particle and assign its corresponding frame field to the particle. During the particle optimization, the particle positions keep updating. To speed up the frame field query, the frame field is stored in a regular grid, with each cell small enough to cover less than k tetrahedra, where k is a small number. Thus for each particle, the frame field query can be performed in a constant time.

Kernel width and neighbor radius: Kernel width σ is an important parameter in our proposed Gaussian hole kernel. The largest slope of a single Gaussian kernel happens at $\sqrt{2}\sigma$ from its center. We have tested different σ and found that the performance is similar when σ is in the range of $[0.25l^*, 0.35l^*]$. In all of our experiments, we use $\sigma = 0.3l^*$. Neighbor radius R is used to cut off the Gaussian kernel supporting region to make the computation faster. Both the value and the slope of Gaussian kernel are almost zero at $5\sqrt{2}\sigma$ away from its center. It is unnecessary to consider the neighboring particles more than $5\sqrt{2}\sigma$ away. We set $R = 1.5l^*$, which includes all its one-ring neighbors while leaving some buffers to attract potential neighboring particles right outside its one-ring domain.

Our Gaussian hole kernel is the summation of one positive Gaussian kernel and six negative ones as shown in Eq(4.6). The positive Gaussian kernel pushes particles away and the negative ones attract neighboring particles into holes. Here we use the weight w to illustrate the balance of these two effects.

$$E_{ij} = w e^{-\frac{\left\|\mathbf{v}_{ij}\right\|^2}{2\sigma^2}} - \frac{1}{6}\sum_{k=1}^{6} e^{-\frac{\left\|\mathbf{v}_{ij}-\mathbf{h}_6(k)\right\|^2}{2\sigma^2}}, \tag{4.6}$$

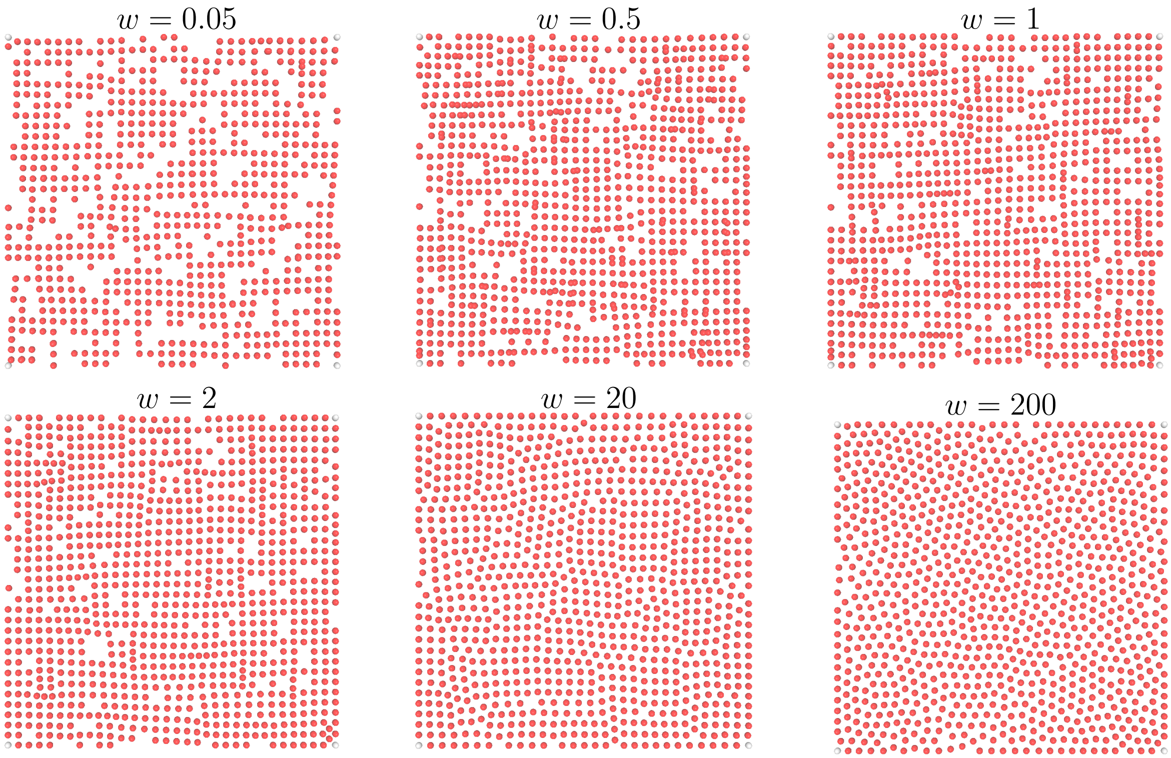

$w = 0.05$
$w = 0.5$
$w = 1$
$w = 2$
$w = 20$
$w = 200$

Figure 4.14: Optimization of particles on a 2D plane with different weights w.

If $w >> 1$, our energy will perform like a single Gaussian kernel where particles are evenly pushed away without hexahedral/quadrilateral pattern. If $w << 1$, particles fall into holes in coincidence without spreading out. Figure 4.14 shows the effect of different w on particle optimization in a 2D plane, after one round of L-BFGS optimization. We find out that setting $w = 1$ achieves a good balance between these two effects. Thus we use it for all of our experiments in this paper.

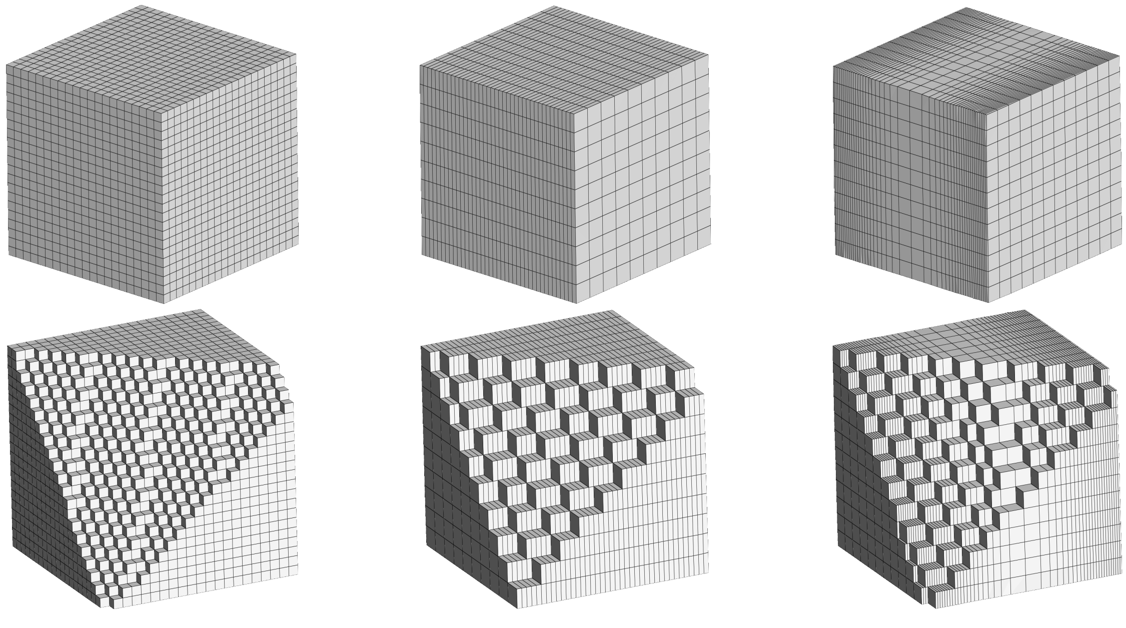
Our algorithm has no restriction on the size or the topology of the volume domain. It is capable of handling all kinds of frame fields, including strongly stretched or highly warped ones. Our method is tested on a large data set with both CAD models and non-CAD models. It always generates high-quality hexahedral-dominant meshes that follow the given frame field. All the experiments are conducted without any user-interaction or parameter tweaking. The main parameters are summarized in Table 4.5.

Table 4.5: Experimental parameters

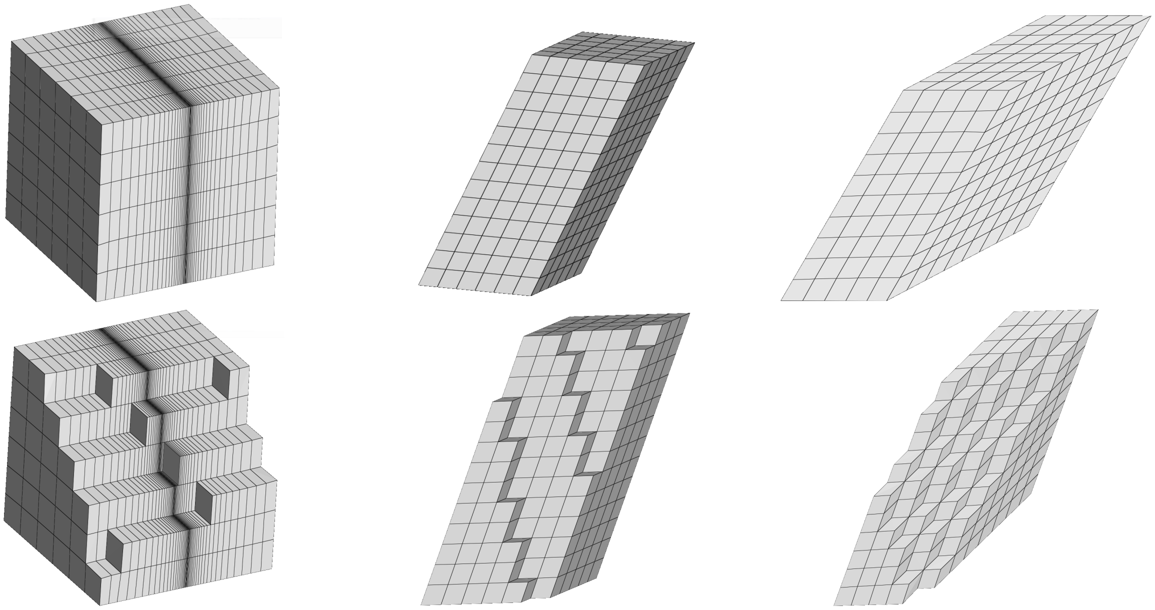
$d_{particle}$	d_{edge}	d_{quad}	d_{hex}	e_{ins}	d_{ins}	σ	R
$0.5l^*$	$0.75l^*$	$0.9l^*$	$0.9l^*$	-0.1	$0.8l^*$	$0.3l^*$	$1.5l^*$

Meshing with analytic frame field: To verify the performance of our method, we test different frame fields on simple geometry volumes, e.g., Cubes, Deformed Cubes, and Cylinder at first. We have tried the following metrics: an identity metric $\mathbf{T} = \mathbf{I}$ (Figure 4.15a), a constant scaling metric $\mathbf{T} = diag\{4, 1, 1\}$ (Figure 4.15b), two varying scaling metrics $\mathbf{T} = diag\{8x, 1, 1\}$ (on cube $[-1, 1]^3$ Figure 4.15c), $\mathbf{T} = diag\left\{\left(1.0125 - e^{-|x-0.6|}\right)^{-1}, 0, 0\right\}$ (on cube $[0.1, 1.1]^3$ (Figure 4.15d), and two non-orthogonal warping metrics: shear on y axis $\mathbf{T} = \{\{1, 0, 0\}^T, \{1, 2, 0\}^T, \{0, 0, 1\}^T\}$ (Figure 4.15e), and shear on y axis and z axis $\mathbf{T} = \{\{1, 0, 0\}^T, \{1, 2, 0\}^T, \{1, 1, 1\}^T\}$ (Figure 4.15f). All the optimizations start from a randomized initialization. The resulting particles are perfectly aligned with the given frame field. Based on the particle results, all-hex meshes are generated.

Importance of scale correction: We use the Cylinder model to show the importance of scale correction in frame fields. Different frame fields may leads to different mesh results, which also happens for PGP3D, LpCVT and Gao et al.'s method. Generating frame fields for arbitrary domain, which leads to all-hex meshes, is still an open problems and also one of the most difficult problems in all-hex meshing. The existing methods generate only cross fields for arbitrary domain. How to robustly and effectively generate hex-dominant meshes from those fields is the point of hex-dominant meshing. In Figure 4.16, we test three different



(a) Identity (b) Stretch (c) Density1

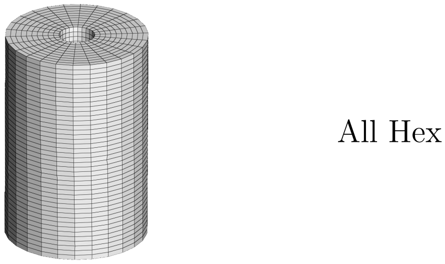


(d) Density2 (e) Shear_y (f) Shear_yz

Figure 4.15: Hexahedral meshing with different frame fields on Cube and Warped Cube models. (a) $\mathbf{T} = \mathbf{I}$; (b) stretch along x-axis; (c) linear scale along x axis, first linearly decreasing then linearly increasing; (d) scale linearly increasing along x axis; (e) shear along y; (f) shear along both y and z.

$$\begin{bmatrix} \cos\theta & \frac{-9\sin\theta}{5\pi r} & 0 \\ \sin\theta & \frac{9\cos\theta}{5\pi r} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

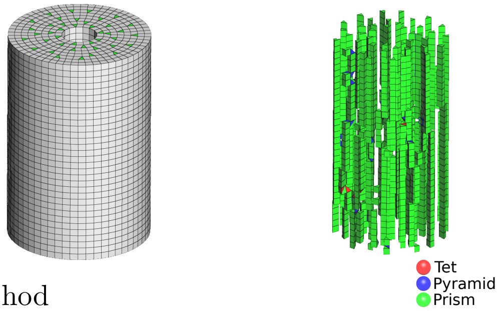
$$\#Vert = 12298$$
$$H_{vol} = 100\%$$
$$\#Hex\% = 100\%$$



(a) Particle method

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

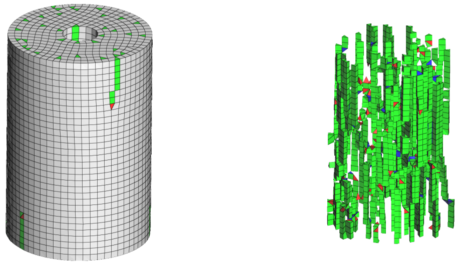
$$\#Vert = 12117$$
$$H_{vol} = 94.49\%$$
$$\#Hex\% = 87.90\%$$
$$Q_{SJ} = 0.98$$



(b) Particle method

Discrete
frame
field
by (Gao et al., 2017)

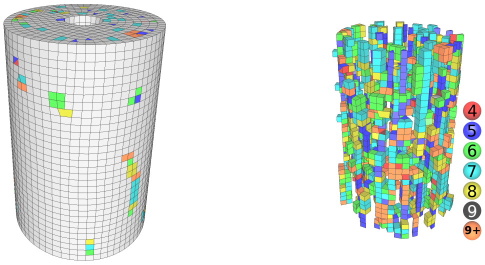
$$\#Vert = 12187$$
$$H_{vol} = 93.64\%$$
$$\#Hex\% = 86.16\%$$
$$Q_{SJ} = 0.98$$



(c) Particle method

Discrete
frame
field
by (Gao et al., 2017)

$$\#Vert = 12619$$
$$H_{vol} = 85.20\%$$
$$\#Hex\% = 84.52\%$$
$$Q_{SJ} = 0.98$$



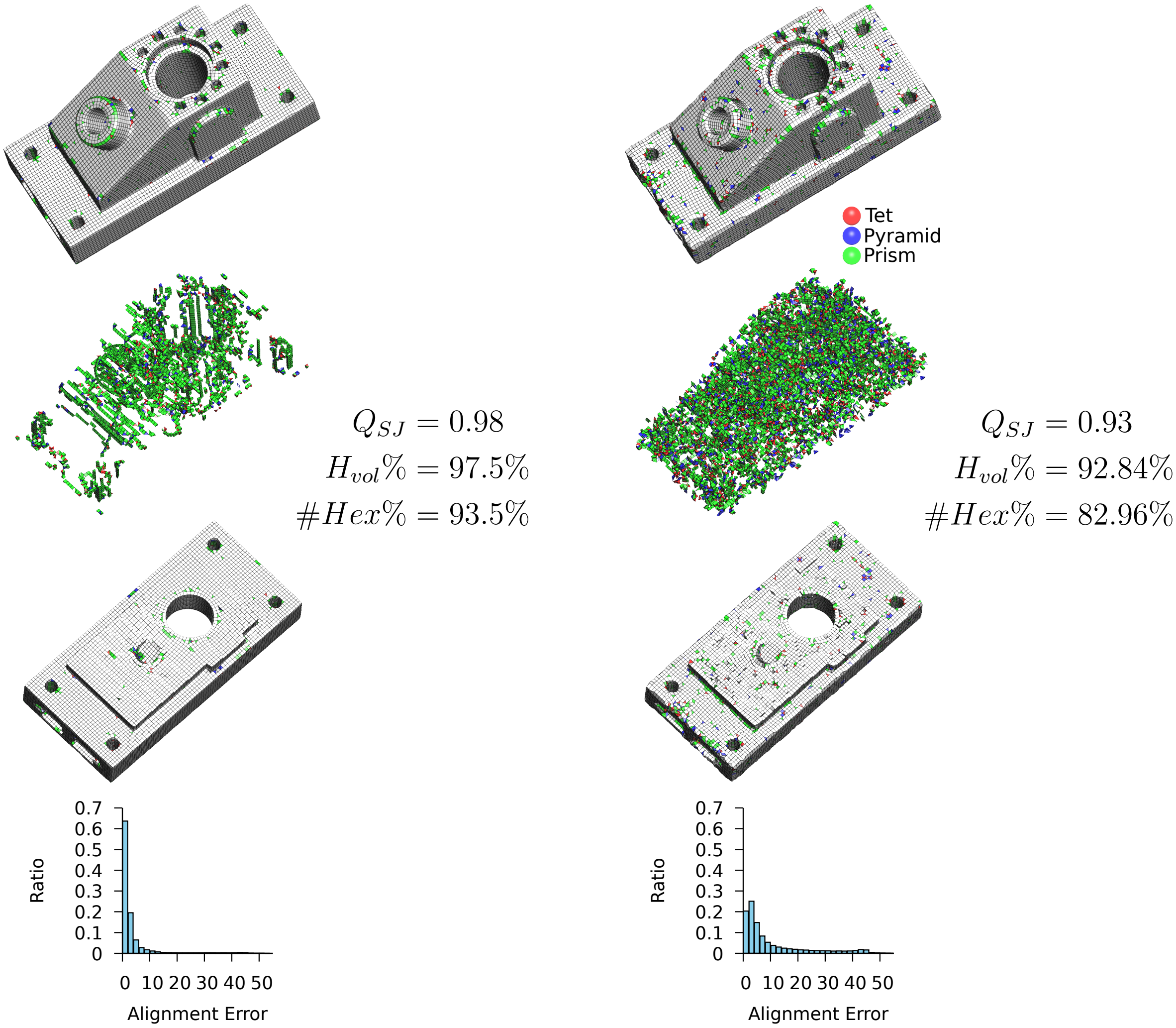
(d) Gao et al.'s method

Figure 4.16: Three different frame fields are tried on the Cylinder model. The first and second rows show our results with closed-form frame fields, with and without scale correction along the radius, respectively. The third and fourth rows show our Particle and Gao et al.'s results with a discrete frame field generated by their method. The last column shows the corresponding non-hex elements. Non-hex elements including prisms, pyramids, and tetrahedra are shown in green, blue, and red, respectively. In Gao et al.'s results, non-hex polyhedra with different number of faces are rendered in different colors, including 6-face polyhedra with triangle faces shown in green color. Our result in the first row does not have non-hex element.

frame fields on a cylinder to show the following two aspects: (1) the robust performance of our method on different frame fields and the reason why prism appears most in our results due to cross files; (2) with the same frame field, our result has higher quality than that of (Gao et al., 2017) method. The three different frame fields are closed-form cross fields with and without scale correction, and a discrete cross field generated by (Gao et al., 2017). We obtain an all-hex result with the scale-corrected frame field as shown in Figure 4.16a. The frame field used in Figure 4.16b does not have any singularity as compare to the discrete frame field used in Figure 4.16c and Figure 4.16d. And we observe higher hex volume ratio in the result of Figure 4.16b. This example tells us that even if the frame field is smoothly defined everywhere, the result mesh may still contain non-hex elements. Among the non-hex elements, prisms appear most often to fill the gap of scale-changing along the radius direction. The pyramids appear consecutively along an axis, which indicates the particles are well distributed. Once we add a scale correction on the frame field, we got an all-hex result shown in Figure4.16a.

Hexahedral-dominant meshing with discrete cross fields: In the following experiments and comparisons, all the frame fields are generated by either Huang et al.'s method (Huang et al., 2011) or Gao et al.'s method (Gao et al., 2017).

Comparison with L_p-CVT (Lévy and Liu, 2010): According to our knowledge, L_p-CVT is the only variational method in hexahedral meshing. Our particle optimization does not need to calculate the Voronoi cells. It is much faster than L_p-CVT method and the optimization converges to much better results due to the nice property of our energy. The result of L_p-CVT method on Anc101 model has 52676 hexahedra, 3035 tetrahedra, 6098 prisms and 1889 pyramids. The L_p-CVT optimization takes 921 seconds. Our result has 55422 hexes, 704 tets, 2755 prisms and 472 pyramids. The optimization takes 147 seconds. The comparison is shown in Figure 4.17. The result of our method has less non-hex elements and better alignment with the input frame field.



(a) Particle method

(b) L_p-CVT method

Figure 4.17: A comparison of our particle (left) and L_p-CVT (right) methods on Anc101 model. The first row is the result mesh on domain boundary. The second row shows the non-hex elements including prisms, pyramids, and tetrahedra in green, blue, and red, respectively. The third rows is the clipping view. The fourth row shows the histograms of alignment errors measured in degrees. .

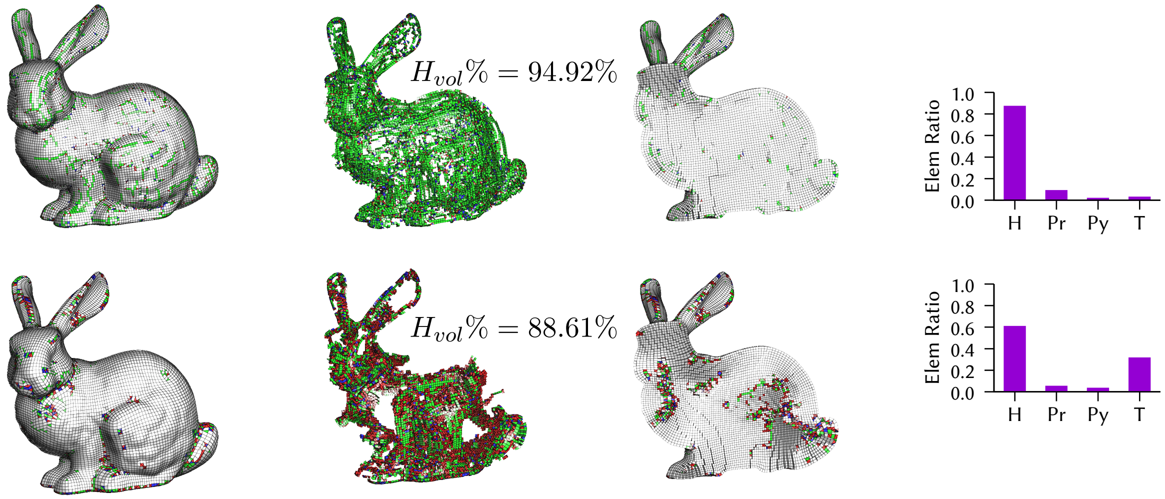
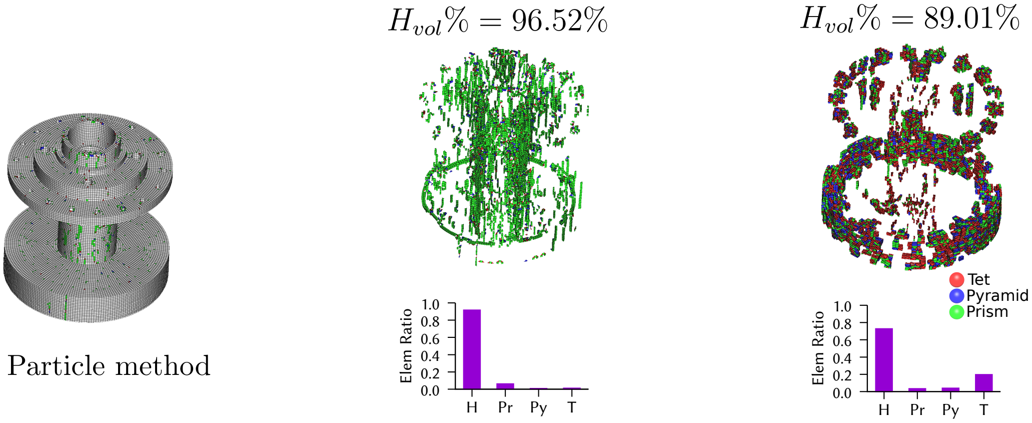


Figure 4.18: Comparison between our method (left) and PGP3D (right) methods on Bunny model. The non-hex elements are shown in the second row, including prisms, pyramids, and tetrahedra rendered in green, blue, and red, respectively. The third row is the clipping view. The bar graphs in the fourth row shows the ratio of numbers of all elements including hexahedra (H), prisms (Pr), pyramids (Py) and tetrahedra (T). Note that the views of non-hex elements in the second row may give you the wrong impression of more non-hex elements in our result, which is caused by 3D projection. Non-hex elements in different depth are overlaid together. The following two rows are good proof of the better quality of our results.

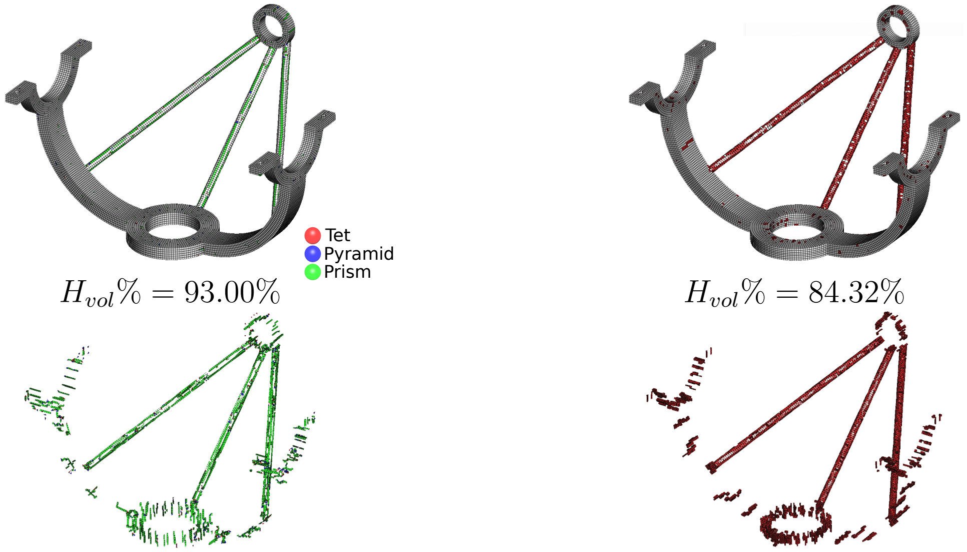


(a) Particle method

(b) Particle method

(c) PGP3D method

Figure 4.19: Comparison between our method and PGP3D method on Cubo model. The non-hex elements including prisms, pyramids, and tetrahedra are rendered in green, blue, and red, respectively. The bar graph shows the ratio of numbers of all elements including hexahedra (H), prisms (Pr), pyramids (Py) and tetrahedra (T).



(a) Particle method (b) PGP3D method

Figure 4.20: Comparison between our method and PGP3D method on Switchmec model. The non-hex elements shown in the second row including prisms, pyramids, and tetrahedra are rendered in green, blue, and red, respectively.

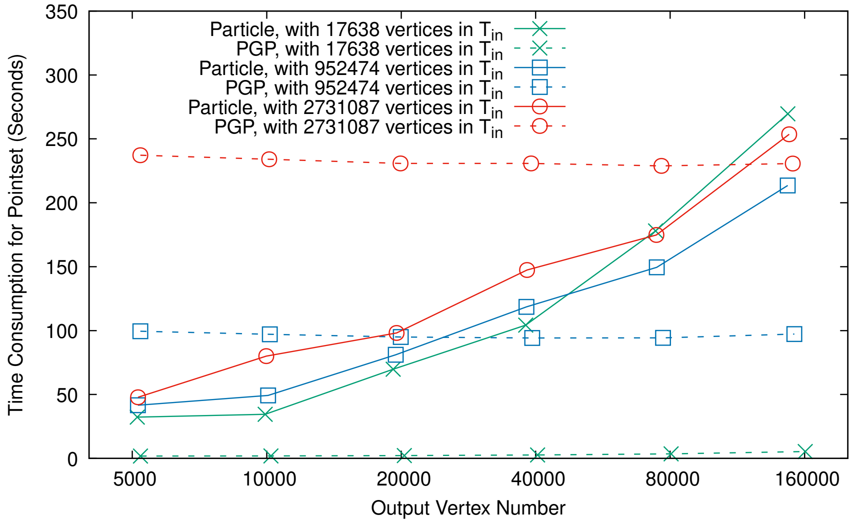


Figure 4.21: Comparison of computation time to generate point set between our method and PGP3D method on Venus model. Three input tetrahedral meshes of different densities are used. Their vertex numbers are given in the legend. For each input tetrahedral mesh, point sets of different sizes are generated. The experiments of PGP3D are done by two machines: Intel(R) Core(TM) i7-6820HQ CPU @ 2.70GHz (the experiments with the two smaller tetrahedral meshes) and Intel(R) Core(TM) i7-6800K CPU @ 3.40GHz (the experiments with the largest tetrehedral mesh).

Comparison with PGP3D (Sokolov et al., 2016): In this subsection, the comparison is with the PGP3D method which outperforms L_p-CVT method and front propagation-based method. Figure 4.18 and Figure 4.19 show the detailed comparison between our particle method and PGP3D method on two models. Note that the non-hex elements shown in the figures is 3D projections. The non-hex elements on different depth planes are overlaid together. The $H_{vol}\%$ is more convincing. The pipeline of PGP3D has six steps: frame field generation, curl correction, global parameterization, point set extraction, mesh refinement, and tet-to-hex conversion. PGP3D method has excellent performance on most CAD models. Nonetheless, the authors pointed out that PGP3D may fail on small parts of models, e.g., the three cylinders of Switchmec model as shown in Figure 4.20. Our particle method successfully generates hexahedral-dominant meshes, while PGP3D only generates tetrahedra in those small parts. To generate the point set, PGP3D method first calculates the global parametrization, and then extracts points with integer valued coordinates in the map. Based on fancy periodic variables and transition functions, the problem is formulated as an overdetermined linear system. The solution to an overdetermined system may have larger errors on small parts of the model. That is why PGP3D method may fail at the small parts of a model. Our particle method directly optimizes the positions of vertices. So we have more controls on the vertex position to preserve the sharp feature and mesh boundary. Besides that our Gaussian hole kernel provides some space to accept small difference of edge length. When the targeting edge length is not suitable for the small parts, our optimization will allow a little smaller / larger distance between particles. The statistic of quality results are shown in Table 4.6. For both CAD and non-CAD models, our results demonstrate better $\#Hex\%$ and $H_{vol}\%$, especially in non-CAD models. We notice that the non-hex parts of the resulting meshes in PGP3D and our methods are different. PGP3D uses curl correction, so prism is the least common non-hex element and tetrahedron is the dominant non-hex element. However, in our meshing result, prism is the dominant non-hex element. Note that our pipeline does not include the curl correction. The prisms usually appear to fill the scale gap, and they usually appear consecutively in lines, which demonstrates that our particle positions are well optimized.

The complexity of our particle method scales with the particle number of the output mesh. On the contrary, point set optimization of PGP3D scales with input size, because it models the problem as a large scale linear system whose dimension is proportional to edge number of the input tetrahedral mesh. When the output hex-dominant mesh has a very large vertex number, our method may be slower than PGP3D. When the domain is complicated and a dense tetrahedral mesh is used as input, PGP3D would take longer time. The comparison of time consumption of point set generation on the Venus model is shown in Figure 4.21.

Comparison with Gao et al.'s method (Gao et al., 2017): Their method automatically and robustly converts any tetrahedral meshes into an isotropic hexahedral-dominant meshes. Our particle method not only can generate isotropic but also anisotropic hexahedral-dominant meshes as shown in Figure 4.15 and Figure 4.16a. Mesh extraction of Gao et al.'s method relies on a set of local topological operations to collapse and split edges, faces and polyhedra. The input tetrahedral mesh requires to be denser than the output hexahedral-dominant mesh. The polyhedral agglomeration complexity is proportional to the size of input tetrahedral mesh. To construct a dense hexahedral-dominant mesh, their method needs a much denser input tetrahedral mesh, which means more extraction time. Their resulting meshes may contain inverted, self-intersecting or collapsed elements. And the non-hex elements in Gao et al.'s results are very complicated and undetermined, which are p-face polyhedra with k triangle faces and $p - k$ quad faces, where $0 \leq k \leq p$. The comparison between our method and Gao et al.'s method are shown in Figure 4.22, Figure 4.23, and Figure 4.24. The non-hex polyhedra with the same number of faces are rendered with the same colors. The max p of the result meshes is given in the $\#p$ column in Table 4.7. Our results only contain tetrahedron, pyramid and prism as non-hex elements. More comparisons are given in Table 4.7. Our results have better hex quality. By setting a higher Q_{bound} in the mesh generation stage, our result mesh will have a higher Q_{SJ} but lower $H_{vol}\%$. In the comparisons, we adjust the Q_{bound} to make sure that Q_{SJ} are the same in both methods, then compare the $H_{vol}\%$. The results show that our method has larger $H_{vol}\%$. We also draw the relationship between Q_{SJ} and $H_{vol}\%$ in Figure 4.22c and Figure 4.23 by setting Q_{bound} from 0 to 0.9, which shows that our results have better quality. In Table 4.7, the $H^*_{vol}\%$ and Q^*_{SJ} columns list the hex quality of our results when setting $Q_{bound} = 0$.

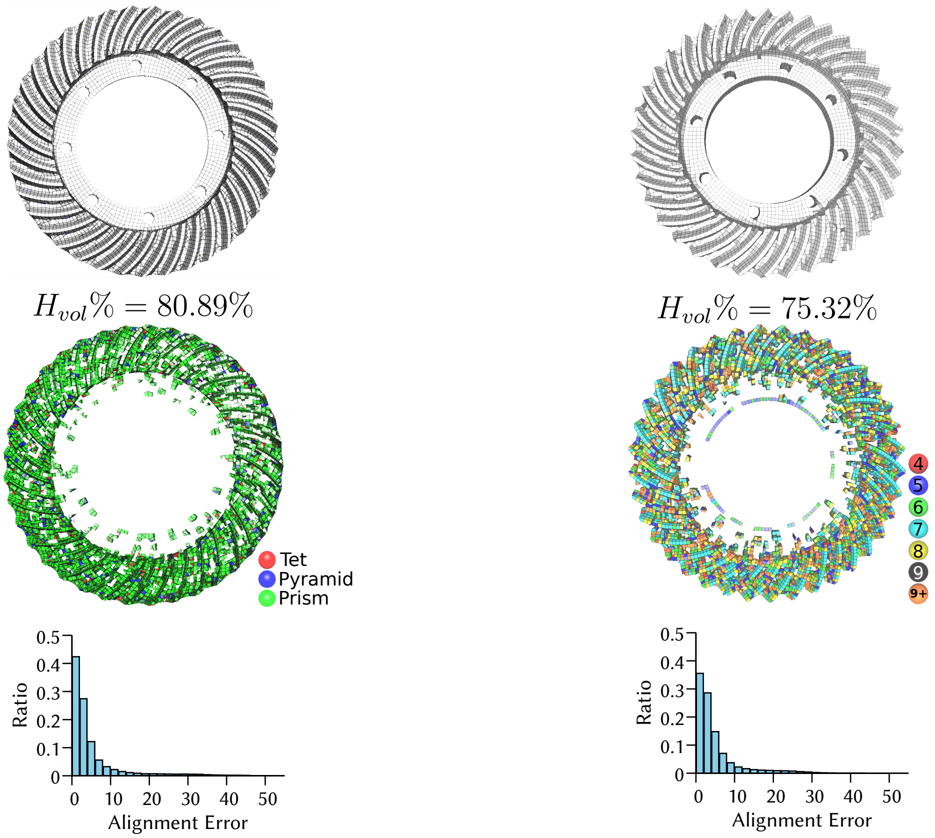
Discussion on mesh refinement: Mesh refinement is performed by inserting more vertices to the surface to guarantee the boundary geometry being well preserved as shown in Figure 4.25. We use Skull model as an example to show the effect of mesh refinement on a model with fine details. Table 4.8 shows the statistics before and after mesh refinement. We can see the Hausdorff distances are reduced dramatically by about 90 percent with mesh refinement. However, this step also causes more non-hex elements along the boundary. To make a fair comparison, we include this refinement step when we compare with PGP3D, and remove this step when we compare with Gao et al.'s method.

4.4.4 Limitation and Future Work

The extensive experimental results show that our method obtains higher quality hexahedral-dominant meshes than other state-of-the-art algorithms. However, there are still some limitations of our proposed work: (1) If the object has thin voids or thin concave regions (e.g., thinner than 1 element size), there may be a problem because we use the Euclidean distance for the computation. The thin parts may only hold one layer of particles, in which case the mesh generation stage would fail. (2) For arbitrary models, generating a smooth frame field with correct scale to reduce the number of non-hex elements is still an open problem. Our particle method produces perfect results on frame fields with correct scales, as shown

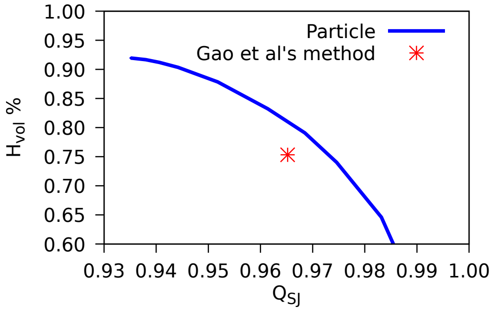
Table 4.6: Statistics and time consumption of models with our particle method and PGP3D method. #Vert is the number of particles in the resulting mesh. $\#Hex\%$ and $H_{vol}\%$ are used to measure the proportion of the hexahedral elements. Non-hex element proportions are measured by $\#Pri\%$, $\#Pyr\%$ and $\#Tet\%$. The shape quality of the hex element is measured by Scaled Jacobian Q_{SJ} in Eq. (4.5). The Hausdorff distance before and after refinement are listed in column dist1 and dist2 respectively. The number before "\|" is the Hausdorff distance of the result mesh to the input boundary mesh. The number after "\|" is the Hausdorff distance of the input boundary mesh to the result mesh. The time consumption for the particle optimization (Pointset), mesh refinement (Refine) and Tet. to Hex. mesh construction (Tet2Hex) are given. The total time is provided. The time items with * are the total time given in PGP3D paper, which include all the steps. All the times are measured in seconds. Note: n/a denotes that we do not have their results.

Model	Method	#Vert	$H_{vol}\%$	$\#Hex\%$	$\#Pri\%$	$\#Pyr\%$	$\#Tet\%$	Q_{SJ}	dist1	dist2	Pointset(s)	Refine(s)	Tet2Hex(s)	Total(s)
`cubo`	Particle	113,457	96.52	91.77	6.24	0.83	1.17	0.98	0.560 |0.441	0.469 |0.147	305.50	23.70	95.46	424.66
	PGP3D	109,182	89.01	72.87	3.35	4.06	19.72	0.98	0.782 |0.671	0.705 |0.257	12.88	69.54	98.01	180.43
`propeller`	Particle	142,287	95.43	87.24	8.15	1.94	2.67	0.97	0.261 |0.212	0.031 |0.039	506.63	51.62	116.09	674.34
	PGP3D	132,469	91.10	71.27	4.33	2.43	21.97	0.97	0.355 |0.327	0.035 |0.062	80.23	119.82	123.28	323.33
`bunny`	Particle	122,969	94.92	86.98	8.81	1.58	2.62	0.97	0.511 |0.376	0.079 |0.081	327.99	17.70	124.16	469.85
	PGP3D	116,149	88.61	60.57	5.05	3.11	31.27	0.95	0.791 |0.779	0.123 |0.129	49.28	42.6	136.43	228.31
`david`	Particle	37,130	88.26	69.76	17.43	6.80	9.00	0.93	0.560 |0.487	0.080 |0.288	85.93	53.10	29.83	168.86
	PGP3D	32,021	76.03	35.38	n/a	n/a	n/a	0.93	n/a	n/a	n/a	n/a	n/a	651.86*
fertility	Particle	26,855	86.71	56.87	20.76	8.82	13.55	0.93	0.849 |0.583	0.044 |0.044	158.60	28.34	18.99	205.93
	PGP3D	20,068	78.40	33.63	9.09	8.03	49.24	0.93	1.292 |0.904	0.069 |0.071	149.21	145.71	16.51	311.43
`gargo`	Particle	45,684	83.89	47.61	21.16	12.86	18.37	0.90	0.875 |0.615	0.073 |0.237	179.94	97.75	33.32	311.01
	PGP3D	29,936	68.91	25.81	n/a	n/a	n/a	0.91	n/a	n/a	n/a	n/a	n/a	707.52*
`impeller`	Particle	15,287	89.06	73.67	18.26	3.38	4.67	0.94	1.154 |1.027	0.121 |0.143	75.44	5.89	9.70	91.03
	PGP3D	13,896	81.11	53.31	7.13	4.58	34.97	0.95	1.408 |0.848	0.333 |0.219	5.85	11.9	10.7	28.45
`skull`	Particle	39,682	88.31	49.45	18.34	12.62	19.59	0.93	1.170 |0.849	0.063 |0.040	168.88	42.89	30.75	242.51
	PGP3D	33503	82.93	38.03	n/a	n/a	n/a	0.93	n/a	n/a	n/a	n/a	n/a	2429.86*
`venus`	Particle	44,450	84.60	37.34	21.90	16.60	24.16	0.91	1.020 |0.666	0.050 |0.033	116.47	53.37	29.54	199.38
	PGP3D	37258	79.11	29.59	n/a	n/a	n/a	0.93	n/a	n/a	n/a	n/a	n/a	5404.35*
`switchmec`	Particle	40,995	93.00	85.66	12.42	0.91	1.00	0.99	0.007 |0.170	0.007 — 0.170	173.11	0	23.31	194.42
	PGP3D	41,318	84.32	48.95	0	0	51.05	0.99	n/a	n/a	n/a	n/a	n/a	n/a



(a) Particle method

(b) Gao et al.'s method



(c) Hex shape quality V.S. hex ratio

Figure 4.22: A comparison between our method and Gao et al.'s method on ASM_Diff027, a CAD model with thin features. The first row shows the result on domain boundary. The second row shows the non-hex elements. In our result, non-hex elements including prisms, pyramids, and tetrahedra are shown in green, blue, and red, respectively. In Gao et al.'s results, non-hex polyhedra with different number of faces are rendered in different colors, including 6-face polyhedra with triangle faces shown in green color. The third row shows the histogram of alignment error measured in degrees. (a) and (b) are comparisons with the same Q_{SJ}. The relationship between Q_{SJ} and $H_{vol}\%$ is shown in (c).

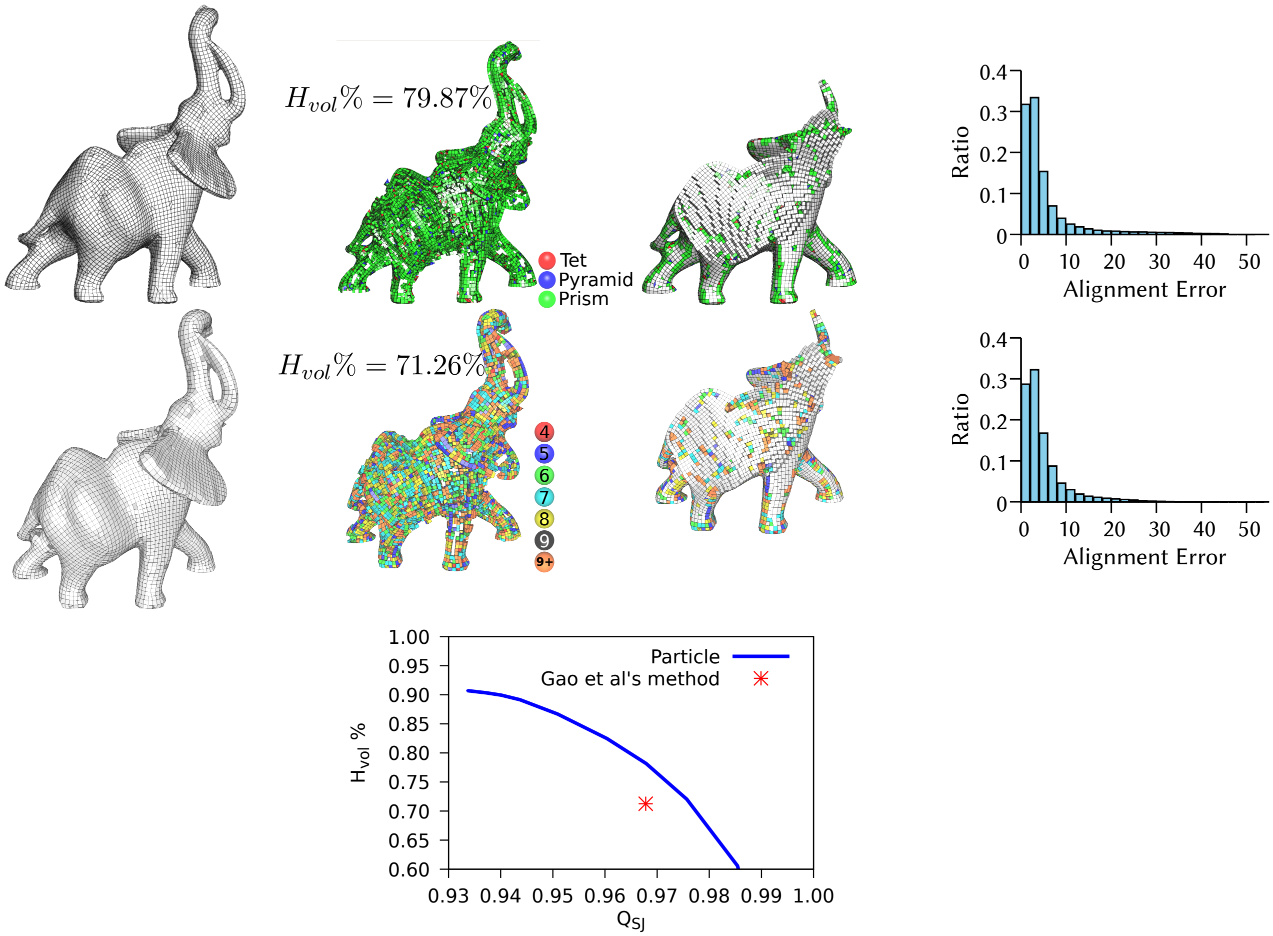
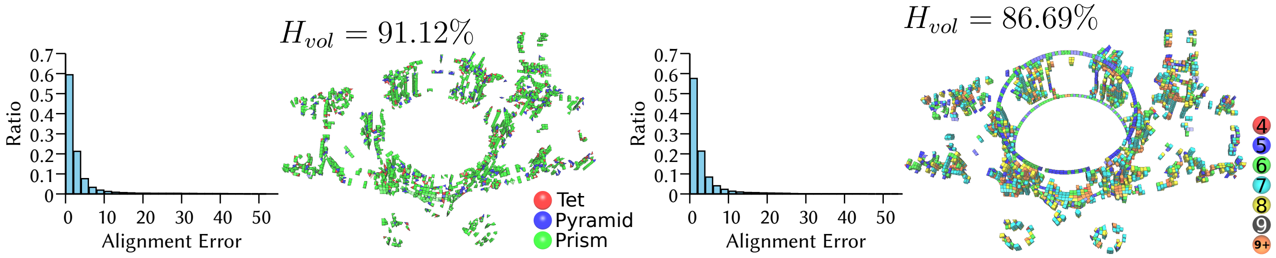


Figure 4.23: A comparison of our method and Gao et al.'s method on the Elephant Model. The first and second row are the results of our particle method and Gao et al.'s method. The first column shows the result mesh on domain boundary. The second column shows the non-hex elements. In our result, non-hex elements including prisms, pyramids, and tetrahedra are shown in green, blue, and red, respectively. In Gao et al.'s results, non-hex polyhedra with same number of faces are rendered in one color, including 6-face polyhedra with triangle faces shown in green color. The clipping view is shown in the third column. The fourth column shows the histogram of alignment error measured in degrees. The first two rows are comparisons under the same Q_{SJ}. The relationship between Q_{SJ} and$H_{vol}\%$ is shown in the last row.

Table 4.7: Statistics and time consumption of models with our particle method and Gao et al.'s method. Our results are in the gray rows, and Gao et al.'s results are in the white rows. #Vert is the number of particles in the resulting mesh. $H_{vol}\%$ is the volume percentage of hex elements. The quality of hex element is measured by Scaled Jacobian Q_{SJ}. To show the advantage of our method, we adjust Q_{bound} to get the same Q_{SJ} as Gao et al.'s result, while obtaining higher $H_{vol}\%$. $H^*_{vol}\%$ and Q^*_{SJ} are the hex quality of our method when setting $Q_{bound} = 0$. $\overline{\theta_a}$ is the mean alignment error. $\#p$ is the max number of faces in the result hexahedral-dominant mesh. The total computation time is provided in seconds.

Model	#Vert	$H_{vol}\%$	Q_{SJ}	$H^*_{vol}\%$	Q^*_{SJ}	$\overline{\theta_a}$	#p	Time (s)
ASM_Diff027	27398	80.89	0.97	91.98	0.93	4.89	6	199.69
(Figure 4.22)	29,813	75.32	0.97	n/a	n/a	4.78	44	389.67
Boeing_part	29,883	95.35	0.99	96.94	0.98	2.00	6	176.31
	29,011	92.63	0.99	n/a	n/a	2.17	40	170.32
Cylinder	12,187	93.64	0.98	94.69	0.97	3.73	6	70.55
(Figure 4.16)	12,619	85.20	0.98	n/a	n/a	3.41	20	347.65
Daratech	27,084	91.73	0.98	95.99	0.96	3.17	6	141.05
	27,129	86.54	0.98	n/a	n/a	2.62	22	627.70
Elephant	28,498	79.87	0.97	90.70	0.93	5.39	6	177.95
(Figure 4.23)	28,796	71.26	0.97	n/a	n/a	4.78	42	378.41
Fertility	16,422	82.56	0.97	90.16	0.94	5.06	6	112.22
	16,381	77.50	0.97	n/a	n/a	4.28	27	772.84
Front_Upright	33,428	91.12	0.98	95.51	0.96	3.35	6	164.60
(Figure 4.12 and Figure 4.24)	33,119	86.69	0.98	n/a	n/a	3.32	32	217.79
Mazewheel	27,253	89.34	0.98	94.75	0.96	3.35	6	155.93
	27,145	84.78	0.98	n/a	n/a	2.78	19	494.17
Upright_6_3	31,567	89.34	0.98	95.27	0.96	3.48	6	179.53
	31,558	86.14	0.98	n/a	n/a	3.21	19	1565.74
Venus	23,640	85.07	0.98	93.55	0.96	4.43	6	147.98
	23,844	81.81	0.98	n/a	n/a	4.14	30	403.29
Rockerarm	15,704	84.25	0.98	92.71	0.95	3.86	6	127.02
	15,345	82.08	0.98	n/a	n/a	3.21	26	641.10



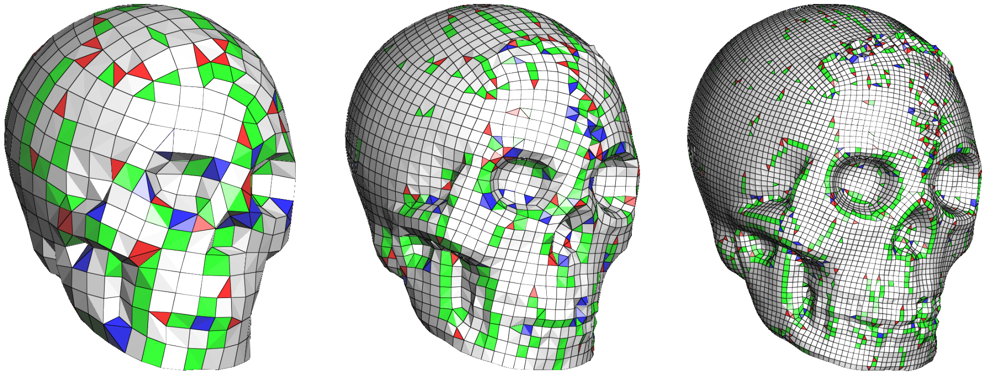
(a) Particle method: alignment error and non-hex elements (b) Gao et al.'s method: alignment error and non-hex elements

Figure 4.24: Comparison of our result (left) and Gao et al.'s result (right) on Front_Upright Model. In our result, non-hex elements including prisms, pyramids, and tetrahedra are shown in green, blue, and red, respectively. In Gao et al.'s results, non-hex polyhedra with different number of faces are rendered in different colors, including 6-face polyhedra with triangle faces shown in green color.

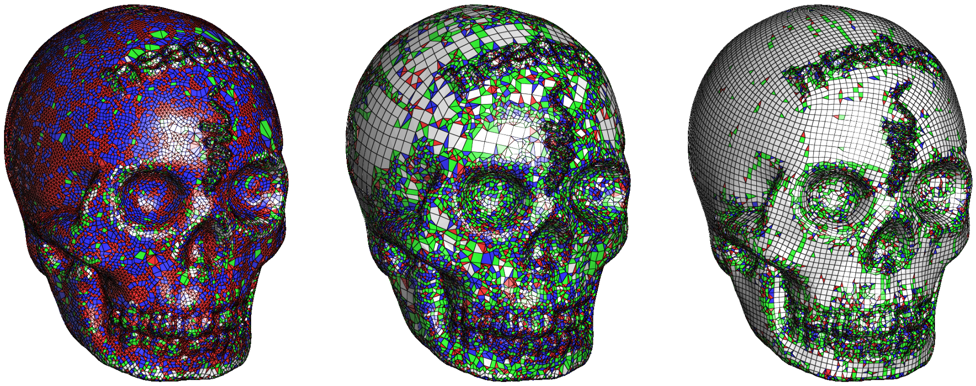
Table 4.8: The quality statistics and time consumption of meshing before and after mesh refinement on the Skull model. The gray and white rows are the results before and after mesh refinement respectively. dist is the Hausdorff distance. The number before |is the Hausdorff distance from the resulting mesh to the input boundary, the number after |is the Hausdorff distance in the reverse way.

#vert	$H_{vol}\%$	$\#Hex\%$	$\#Prism\%$	$\#Pyr\%$	$\#Tet\%$	dist	Q_{SJ}	Time (s)
12837	0.90	77.27	15.49	2.76	4.47	0.0398 |0.0266	0.94	116.61
28190	0.81	26.44	21.11	20.26	32.19	0.0220 |0.0009	0.90	159.78
96504	0.94	86.26	9.45	1.62	2.67	0.0210 |0.0134	0.96	432.74
104741	0.93	76.28	11.86	4.63	7.22	0.0018 |0.0009	0.95	477.64

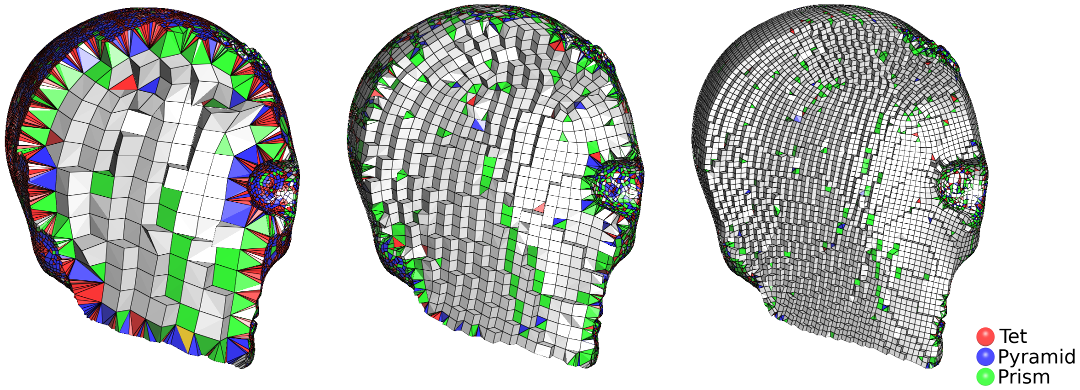
in Figure 4.15 and Figure 4.16a. We would like to investigate scale correction methods for arbitrary frame fields, so that we can further reduce the number of non-hex elements and improve the quality of output meshes. (3) The non-hex elements we generated are only tetrahedra, prisms, and pyramids, which is fundamentally different from Gao et al.'s 2017 results where their non-hex elements may include polyhedra with some undetermined number of triangle faces and quad faces. Our non-hex elements can be transformed to all-hex mesh with one more subdivision step, although the mesh qualities, including scaled Jacobian and alignment error, may not be guaranteed after such subdivision. We would like to investigate such possibility for all-hex meshing in our future work.



(a) Before mesh refine.



(b) After mesh refine.



(c) Clipping view

Figure 4.25: The meshing results before and after mesh refinement on the Skull model. The vertex number is changing from 12837 to 27597 after mesh refinement in the first column. The vertex number is changing from 96504 to 104741 after mesh refinement in the second column. The first row shows the results before mesh refinement. The second row shows the results after mesh refinement. The third row shows the clipping view after mesh refinement.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this dissertation, we first introduce a sliver suppression energy based on shape matching idea. Iteratively minimizing the energy by updating the vertex positions and connectivity generates high-quality isotropic, adaptive, or anisotropic tetrahedral meshes. However, the computation speed of the current implementation is slow. In the future, we would like to improve the computation speed with GPU parallelization.

In the next part of this dissertation, we propose frame field-aligned framework that generates high quality tetrahedral meshes and hexahedral-dominant meshes. This is the first field-aligned tetrahedral meshing method ever proposed. Besides that, our Gaussian Hole Kernel can be easily extended to field-aligned triangular meshing and quadrilateral meshing. Actually, it can be extended to any mesh applications with well-defined one ring structure. In the future, we would explore more applications of our Gaussian Hole Kernel, e.g., other lattices such as A15 and Z-type configurations, etc..

Currently, one ring structure is only used to guide the particle optimization. Mesh generation based on the one ring structure for the resulting particle set will be one of our future work, which avoids the complex computation of anisotropic Voronoi Diagram or the mesh flip operations. Its main task will be to tackle all the possible non-perfect one ring structures in the result particle sets.

Last but not least, the relationship between the input frame field and the output mesh quality is also worthwhile to explore. We also want to find out more field aligned meshes applications.

REFERENCES

Alliez, P., D. Cohen-Steiner, M. Yvinec, and M. Desbrun (2005). Variational tetrahedral meshing. *ACM Transactions on Graphics 24*(3), 617–625.

Ando, R., N. Thürey, and C. Wojtan (2013). Highly adaptive liquid simulations on tetrahedral meshes. *ACM Transactions on Graphics 32*(4), 103.

Barnes, E. and N. Sloane (1983). The optimal lattice quantizer in three dimensions. *SIAM Journal on Algebraic Discrete Methods 4*(1), 30–41.

Baudouin, T. C., J.-F. Remacle, E. Marchandise, F. Henrotte, and C. Geuzaine (2014). A frontal approach to hex-dominant mesh generation. *Advanced Modeling and Simulation in Engineering Sciences 1*(1), 1.

Boissonnat, J.-D., K.-L. Shi, J. Tournois, and M. Yvinec (2015, March). Anisotropic Delaunay Meshes of Surfaces. *ACM Transactions on Graphics 34*(2), 1–11.

Botella, A., B. Lévy, and G. Caumon (2016). Indirect unstructured hex-dominant mesh generation using tetrahedra recombination. *Computational Geosciences 20*(3), 437–451.

Carbonera, C. D. and J. F. Shepherd (2010). A constructive approach to constrained hexahedral mesh generation. *Engineering with Computers 26*(4), 341–350.

CGAL (2017). CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

Chen, L. and M. Holst (2011). Efficient mesh optimization schemes based on Optimal Delaunay Triangulations. *Computer Methods in Applied Mechanics and Engineering 200*(9-12), 967–984.

Chen, L. and J.-c. Xu (2004). Optimal delaunay triangulations. *Journal of Computational Mathematics*, 299–308.

Chen, Z., W. Wang, B. Lévy, L. Liu, and F. Sun (2014). Revisiting optimal delaunay triangulation for 3D graded mesh generation. *SIAM Journal on Scientific Computing 36*(3), A930–A954.

Cheng, S.-W., T. Dey, H. Edelsbrunner, M. Facello, and S.-H. Teng (2000). Sliver exudation. *Journal of the ACM 47*(5), 883–904.

Cheng, S.-W., T. Dey, and J. Shewchuk (2012). *Delaunay Mesh Generation*. Chapman and Hall/CRC.

Chew, L. P. (1997). Guaranteed-quality Delaunay meshing in 3D. In *Proceedings of the 13th Annual Symposium on Computational Geometry*, pp. 391–393.

Doran, C., A. Chang, and R. Bridson (2013). Isosurface stuffing improved: acute lattices and feature matching. In *ACM SIGGRAPH 2013 Talks*, pp. 38.

Du, Q., V. Faber, and M. Gunzburger (1999). Centroidal voronoi tessellations: applications and algorithms. *SIAM review 41*(4), 637–676.

Du, Q. and D. Wang (2003). Tetrahedral mesh generation and optimization based on centroidal voronoi tessellations. *International Journal for Numerical Methods in Engineering 56*(9), 1355–1373.

Du, Q. and D. Wang (2005a). Anisotropic centroidal Voronoi tessellations and their applications. *SIAM Journal on Scientific Computing 26*(3), 737–761.

Du, Q. and D. Wang (2005b). The optimal Centroidal Voronoi Tessellations and the Gersho's conjecture in the three-dimensional space. *Computers & Mathematics with Applications 49*(9-10), 1355–1373.

Du, X., X. Liu, D.-M. Yan, C. Jiang, J. Ye, and H. Zhang (2018). Field-aligned isotropic surface remeshing. *Computer Graphics Forum*.

Fang, X., W. Xu, H. Bao, and J. Huang (2016). All-hex meshing using closed-form induced polycube. *ACM Transactions on Graphics (TOG) 35*(4), 124.

Freitag, L. and P. Knupp (2002). Tetrahedral Mesh Improvement via Optimization of the Element Condition Number. *International Journal for Numerical Methods in Engineering 53*(6), 1377–1391.

Fu, X.-M., Y. Liu, J. Snyder, and B. Guo (2014). Anisotropic simplicial meshing using local convex functions. *ACM Transactions on Graphics 33*(6), 182.

Gao, X., W. Jakob, M. Tarini, and D. Panozzo (2017). Robust hex-dominant mesh generation using field-guided polyhedral agglomeration. *ACM Transactions on Graphics 36*(4), 114.

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

Guo, J., D.-M. Yan, L. Chen, X. Zhang, O. Deussen, and P. Wonka (2016). Tetrahedral meshing via maximal Poisson-disk sampling. *Computer Aided Geometric Design 43*, 186–199.

Han, S., J. Xia, and Y. He (2010). Hexahedral shell mesh construction via volumetric polycube map. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, pp. 127–136. ACM.

Huang, J., Y. Tong, H. Wei, and H. Bao (2011). Boundary aligned smooth 3d cross-frame field. In *ACM transactions on graphics*, Volume 30, pp. 143. ACM.

Ito, Y., A. Shih, and B. Soni (2004). Reliable isotropic tetrahedral mesh generation based on an advancing front method. In *Proceedings of 13th International Meshing Roundtable*, pp. 95–106.

Jakob, W., M. Tarini, D. Panozzo, and O. Sorkine-Hornung (2015). Instant field-aligned meshes. *ACM Transactions on Graphics 34*(6), 189.

Jamin, C., P. Alliez, M. Yvinec, and J.-D. Boissonnat (2015). CGALmesh: A generic framework for Delaunay mesh generation. *ACM Transactions on Mathematical Software 41*(4), 23:1–23:24.

Jiang, T., J. Huang, Y. Wang, Y. Tong, and H. Bao (2014). Frame field singularity correctionfor automatic hexahedralization. *IEEE transactions on visualization and computer graphics 20*(8), 1189–1199.

Klingner, B. M. and J. R. Shewchuk (2007). Agressive tetrahedral mesh improvement. In *Proceedings of 16th International Meshing Roundtable*, pp. 3–23.

Knupp, P. (2001). Algebraic Mesh Quality Metrics. *SIAM Journal on Scientific Computing 23*(1), 193–218.

Labelle, F. and J. R. Shewchuk (2003, June). Anisotropic voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *Proceedings of the nineteenth conference on Computational geometry - SCG '03*, New York, New York, USA, pp. 191. ACM Press.

Labelle, F. and J. R. Shewchuk (2007). Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics 26*(3), 57.1–57.10.

Lévy, B. (2015). Geogram, a programing library of geometric algorithm. http://alice.loria.fr/software/geogram/doc/html/index.html.

Lévy, B. and Y. Liu (2010). Lp centroidal voronoi tessellation and its applications. In *ACM Transactions on Graphics*, Volume 29, pp. 119. ACM.

Li, X.-Y., S.-H. Teng, and A. Üngör (2000). Biting: Advancing front meets sphere packing. *International Journal for Numerical Methods in Engineering 49*(1–2), 61–81.

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

Liu, D. C. and J. Nocedal (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming 45*(3), 503–528.

Liu, Y., W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang (2009). On centroidal voronoi tessellation &mdash; energy smoothness and fast computation. *ACM Transactions on Graphics (ToG) 28*(4), 101.

Lyon, M., D. Bommes, and L. Kobbelt (2016). Hexex: robust hexahedral mesh extraction. *ACM Transactions on Graphics 35*(4), 123.

Maréchal, L. (2009). Advances in octree-based all-hexahedral mesh generation: handling sharp features. *Proceedings of the 18th International Meshing Roundtable*, 65–84.

Meshkat, S. and D. Talmor (2000). Generating a mixed mesh of hexahedra, pentahedra and tetrahedra from an underlying tetrahedral mesh. *International Journal for Numerical Methods in Engineering 49*(1-2), 17–30.

Meyer, M. D., P. Georgel, and R. T. Whitaker (2005). Robust particle systems for curvature dependent sampling of implicit surfaces. In *International Conference on Shape Modeling and Applications*, pp. 124–133.

Mitchell, S. A. and S. A. Vavasis (1992). Quality mesh generation in three dimensions. In *Proceedings of the eighth annual symposium on Computational geometry*, pp. 212–221.

Molino, N., R. Bridson, and R. Fedkiw (2003). Tetrahedral mesh generation for deformable bodies. In *Proc. Symposium on Computer Animation.*

Möller, P. and P. Hansbo (1995). On advancing front mesh generation in three dimensions. *International Journal for Numerical Methods in Engineering 38*(21), 3551–3569.

Neil Molino, Robert Bridson, J. T. and R. Fedkiw (2003). A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *Proceedings of 12th International Meshing Roundtable*, pp. 103–114.

Ni, S., Z. Zhong, Y. Liu, W. Wang, Z. Chen, and X. Guo (2017). Sliver-suppressing tetrahedral mesh optimization with gradient-based shape matching energy. *Computer Aided Geometric Design 52*, 247–261.

Nieser, M., J. Palacios, K. Polthier, and E. Zhang (2012, June). Hexagonal global parameterization of arbitrary surfaces. *IEEE Transactions on Visualization and Computer Graphics 18*(6), 865–878.

Nieser, M., U. Reitebuch, and K. Polthier (2011). Cubecover–parameterization of 3d volumes. In *Computer Graphics Forum*, Volume 30, pp. 1397–1406. Wiley Online Library.

Owen, S. J. (1998). A survey of unstructured mesh generation technology. In *Proceedings of 7th International Meshing Roundtable*, pp. 239–267.

Owen, S. J. and S. Saigal (2000). H-morph: an indirect approach to advancing front hex meshing. *International Journal for Numerical Methods in Engineering 49*(1-2), 289–312.

Panozzo, D., E. Puppo, M. Tarini, and O. Sorkine-Hornung (2014). Frame fields: anisotropic and non-orthogonal cross fields. *ACM Transactions on Graphics 33*(4), 134.

Radovitzky, R. and M. Ortiz (2000). Tetrahedral mesh generation based on node insertion in crystal lattice arrangements and advancing-front-delaunay triangulation. *Computer Methods in Applied Mechanics and Engineering 187*(3-4), 543–569.

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

Ray, N., D. Sokolov, and B. Lévy (2016). Practical frame field generation. *ACM Transactions on Graphics 35*(233).

Schöberl, J. (1997). NETGEN An advancing front 2D/3D-mesh generator based on abstract rules. *Computing and visualization in science 1*(1), 41–52.

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

Shephard, M. S. and M. K. Georges (1991). Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical methods in engineering 32*(4), 709–749.

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

Shepherd, J. F., S. A. Mitchell, P. Knupp, and D. R. White (2000). Methods for multisweep automation. In *Proceedings, 9th International Meshing Roundtable*, pp. 77–87.

Shewchuk, J. R. (2002a). Two discrete optimization algorithms for the topological improvement of tetrahedral meshes. *Unpublished manuscript 65*.

Shewchuk, J. R. (2002b, September). What Is a Good Linear Finite Element? - Interpolation, Conditioning, Anisotropy, and Quality Measures.

Shimada, K. (2006). Current trends and issues in automatic mesh generation. *Computer-Aided Design & Applications 3*, 741–750.

Si, H. (2015). Tetgen, a Delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software 41*(2), 11:1–11:36.

Sokolov, D., N. Ray, L. Untereiner, and B. Lévy (2016). Hexahedral-dominant meshing. *ACM Transactions on Graphics 35*(5), 157.

Solomon, J., A. Vaxman, and D. Bommes (2017). Boundary element octahedral fields in volumes. *ACM Transactions on Graphics 36*(3), 28.

Staten, M. L., S. J. Owen, and T. D. Blacker (2005). Unconstrained paving and plastering: a new idea for all hexahedral mesh generation. In *Proceedings, 14th International Meshing Roundtable*, pp. 399–416.

Tarini, M., K. Hormann, P. Cignoni, and C. Montani (2004). Polycube-maps. *ACM Transactions on Graphics 23*(3), 853–860.

Tournois, J., R. Srinivasan, and P. Alliez (2009). Perturbing slivers in 3D delaunay meshes. In *Proceedings of 18th International Meshing Roundtable*, pp. 157–173.

Turk, G. (1992). Re-tiling polygonal surfaces. In *ACM SIGGRAPH Computer Graphics*, Volume 26, pp. 55–64.

Velho, L., J. de Miranda Gomes, and D. Terzopoulos (1997). Implicit manifolds, triangulations and dynamics. *Neural Parallel and Scientific Computations 5*, 103–120.

Witkin, A. P. and P. S. Heckbert (2005). Using particles to sample and control implicit surfaces. In *ACM SIGGRAPH 2005 Courses*, pp. 260.

Xu, K., X. Gao, Z. Deng, and G. Chen (2017). Hexahedral meshing with varying element sizes. In *Computer Graphics Forum*. Wiley Online Library.

Yamakawa, S. and K. Shimada (2000). High quality anisotropic tetrahedral mesh generation via ellipsoidal bubble packing. In *Proceedings of International Meshing Roundtable*, pp. 263–274.

Yamakawa, S. and K. Shimada (2003). Fully-automated hex-dominant mesh generation with directionality control via packing rectangular solid cells. *International Journal for Numerical Methods in Engineering 57*(15), 2099–2129.

Yan, D.-M., B. Lévy, Y. Liu, F. Sun, and W. Wang (2009). Isotropic remeshing with fast and exact computation of restricted voronoi diagram. In *Computer graphics forum*, Volume 28, pp. 1445–1454. Wiley Online Library.

Yan, D.-M., W. Wang, B. Lévy, and Y. Liu (2010). Efficient computation of 3D clipped Voronoi diagram. In *Proceedings of the 6th International Conference on Advances in Geometric Modeling and Processing*, pp. 269–282.

Yerry, M. A. and M. S. Shephard (1983). A modified quadtree approach to finite element mesh generation. *IEEE Computer Graphics and Applications 3*(1), 39–46.

Yerry, M. A. and M. S. Shephard (1984). Automatic three-dimensional mesh generation by the modified-octree technique. *International Journal for Numerical Methods in Engineering 20*(11), 1965–1990.

Zhong, Z., X. Guo, W. Wang, B. Lévy, F. Sun, Y. Liu, W. Mao, et al. (2013). Particle-based anisotropic surface meshing. *ACM Transactions on Graphics 32*(4), 99–1.

BIOGRAPHICAL SKETCH

Saifeng Ni received her Bachelor's degree and Master's degree from the University of Science and Technology of China, and currently she is a PhD candidate in the Department of Computer Science at The University of Texas at Dallas. Under the supervision of Dr. Guo, Saifeng has been conducting research on mesh optimization. Her research interests include computer graphics, computer vision, and 3D reconstruction, with an emphasis on geometric modeling and processing.

CURRICULUM VITAE

Saifeng Ni

sxn124030@utdallas.edu

Education:

Ph.D.	Computer Science, The University of Texas at Dallas	2012 - 2018
M.S.	E.E., University of Science and Technology of China	2009 - 2012
B.S.	E.E., University of Science and Technology of China	2005 - 2009

Internships:

Research Intern *Virtualbloks, LLC.* *Summer 2015*

- Reconstructed 3D indoor environment by structure from motion and multiview stereo.
- Estimated 3D room layout based on Manhattan world assumption.
- Augmented virtual decorations onto a real room using Vuforia SDK.
- Implemented and evaluated these AR/VR systems on GearVR and Epson Movirio glasses.

Research Intern *Samsung Research America* *Summer 2018*

- Investigate 3D face modeling.
- Implement 3 face fitting pipelines.

Projects:

- *National Graduate Student Mathematical Contest in Modeling in China* (2011)
  - Modeled wedge absorber and microwave anechoic chamber using Matlab, Won **second prize**.
- *Dense stereo reconstruction algorithms comparison on ideal and real data.*

Skills:

- Proficient at **C/C++** and **Matlab**; familiar with Python, Linux.
- Graphics & Vision: OpenGL, GLSL, CGAL, VTK.

Awards:

- “Yang Ya” Scholarship (top 1 female graduate student) in USTC, 2011
- Second Prize in National Graduate Student Mathematical Contest in Modeling in China, 2011