High quality mesh generation for volumetric data processing

Han, Shuchu

2010

Han, S. (2010). High quality mesh generation for volumetric data processing. Master' s thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/42442>

<https://doi.org/10.32657/10356/42442>

NANYANG TECHNOLOGICAL UNIVERSITY

HIGH QUALITY MESH GENERATION FOR VOLUMETRIC DATA PROCESSING

HAN SHUCHU

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Master of Engineering by Research

2010

Abstract

High quality tetrahedral/hexahedral mesh generation is important for a wide range of applications in computer graphics, finite element analysis (FEA), physics-based simulation, and solid modeling. Some examples include realistic simulation of deformable objects in computer graphics and numerical solvers for partial differential equations (PDE) in computational science, especially, the FEA. Due to the importance of mesh generation, or meshing, many researchers and practitioners have studied extensively the theory and applications of meshing technologies over the past decades.

Automatic generating high quality tetrahedral meshes is a difficult task for a variety of reasons. For examples, there are basic mathematical difficulties which make tetrahedral meshing significantly harder than meshing in 2D: 1) regular tetrahedron, the most isotropic 3D simplex, does not tile 3D space, while the equilateral triangle does tile the plane; 2) unlike the 2D case, even well-spaced vertices can create degenerate 3D elements such as *slivers*; 3) dealing with boundaries is also fundamentally more difficult in 3D. The current automatic meshing technologies offer reasonably good solutions for basic linear Finite Element Method (FEM) and basic rendering tasks. More complex analyses and non-linear analyses always require high quality meshes that cannot be generated automatically by current commercial mesh generators. The user still has to spend considerable time and manual labor to make ideal meshes for such analyses.

Due to the fast development of 3D scanning devices (such as MRI and CT), it is very common that a scanned data is extremely bulky. Thus, automatic converting and meshing these huge data set is highly desired. The main objective of this thesis is to investigate efficient techniques to construct a high quality tetrahedral mesh for general volume data. We build the quality tetrahedral mesh in two stages: the first stage is automatic low quality tetrahedral mesh generation, and the second stage is mesh quality improvement by using optimization method.

In this thesis, we investigate the existing unstructured mesh generation methods, the subdivision algorithms and the tetrahedral remeshing techniques. We develop an isotropic mesh generation method by combining and improving several existing techniques. Besides, we successfully apply this technique for anisotropic solid texture transfer by using volume parameterization techniques. The experimental results demonstrate the efficacy of our proposed remeshing technique.

Furthermore, we propose a hexahedral shell mesh construction method based on the developed isotropic tetrahedral remeshing method and volume polycube map. Given a closed 2-manifold and the user-specified thickness, we construct the shell space using the distance field and then parameterize the shell space to a polycube domain. By constructing a hexahedral mesh in the polycube domain naturally, we can obtain a quality hexahedral mesh in the object shell space induced by volume parameterization. We prove that our parameterization is guaranteed to be a homeomorphism. As a result, the constructed hexahedral mesh is free of degeneracy, such as self-intersection, flip-over, etc. Various examples with different topology are provided to demonstrate the efficacy of our method.

Acknowledgments

I am extremely grateful to my supervisor Dr. He Ying, for being a great advisor during my Master of Engineering study. Through his guidance, I learnt to define my research direction and incorporated innovative research ideas. His encouragement and suggestions have propelled me this far.

Special thanks go to Dr. Philip Fu Chi-Wing, for his help and advices during my studies.

I am very grateful to all my group mates during my study at School of Computer Engineering, Nanyang Technology University. My group mates, Xia Jiazhi, Sun Qian, Dao Thi Phuong Quynh, Ying Xiang and William Lai Chi-Fu have spent time and provided valuable suggestions to my work. Also, I would like to express my gratitude to Dr. Zhang Long, Dr. Jin Jianqiu, Dr. Lin Juncong and Dr. Chen Xiaoming for their kindly helping during my research works.

Finally, I would like to express my earnest gratitude to my family for their love and support. I would not have reached the accomplishment of the current stage without their strong support. Thanks Zhou Ying, for her understanding and support, and for her tolerance when I am back home at midnight.

Contents

Abstract	i
Acknowledgments	iii
List of Figures	vi
List of Tables	viii
1 Introduction	1
1.1 Volume data	1
1.2 Tetrahedral mesh generation	2
1.3 Mesh Quality	4
1.4 Motivation	5
1.5 Research Scope	6
1.6 Objectives	6
1.7 Organization	7
2 Related Work	9
2.1 Tetrahedral meshing techniques	9
2.2 Polycube map	13
2.3 Parameterization of Star-shaped volume	14
2.4 Lapped solid textures	14
2.5 Hexahedral elements meshing and shell structure	15
3 Anisotropic Solid Texture Transfer for Star-Shaped Volume	17
3.1 Background	18
3.2 Tetrahedral Mesh Generation	19
3.3 Parameterization of star shapes	24

3.4	Texture transfer	26
3.5	Experimental Results	27
3.6	Summary	27
4	Hexahedral Shell Mesh Construction Based on Tetrahedral Mesh	29
4.1	Background	30
4.2	Theory foundation	31
4.3	Hexahedral Shell Meshing	34
4.3.1	Overview	34
4.3.2	Isotropic tetrahedral remeshing	35
4.3.3	Constructing Shell Space	35
4.3.4	Parameterizing Shell Space	36
4.3.5	Layered Hexahedral Meshing	40
4.4	Experimental Results	42
4.5	Summary	45
5	Contribution and Future Work	49
5.1	Contribution	49
5.2	Future Work	50
	References	52
	Publication	60

List of Figures

1.1	Volume data of skull model. From left to right: scanned data; isosurfaces; surface remeshing result; and tetrahedral mesh.	2
1.2	Tetrahedral mesh generation of an ellipsoid. Left: input surface. Right: output tetrahedral mesh.	3
2.1	Octree-based methods of bunny model [41].	10
2.2	Delaunay approaches for tetrahedral meshing by using constrained delaunay triangulation [9].	11
2.3	Variational tetrahedral meshing approach of bunny model with adapted tetrahedra. Left: input surface. Right: resulted tetrahedral mesh [1]. . .	12
2.4	Polycube map of bunny model. Left: input bunny surface; Right: output polycube [56].	13
2.5	Lapped textures. From left to right: kiwi model, carrot model, tree model [55].	15
3.1	Anisotropic solid textures of star shapes. From left to right: kiwi, watermelon, carrot.	18
3.2	Flow chart of tetrahedral mesh generation.	20
3.3	Inside of Star model by cutting view.	20
3.4	Subdivision scheme.	20
3.5	Subdivision of a tetrahedron.	21
3.6	PWL approximations.	22
3.7	Mesh extraction of star model.	25
3.8	Parameterization of star shape	26
3.9	Anisotropic solid texture transfer.	28

4.1	Shell space parameterization	33
4.2	Shell space construction.	36
4.3	Volumetric harmonic map on the shell space.	39
4.4	Tracing integral curves. Each integral curve follows the gradient vector field of the harmonic function. Thus, it is orthogonal to the iso-surface of the harmonic function (including the two boundary surfaces). We show that each integral curve has unique ending points and all integral curves do not intersect.	41
4.5	Hexahedral meshing of the Skull model (three layers). Left column: 1/4 cutting view; Right column: view of three layers from two different directions.	42
4.6	Layered hexahedral meshing of the Duck model embedded in a sphere. Note the outer and inner boundaries are significantly different (see (a)). (b) shows the isotropic tetrahedral mesh of the shell space. We parameterize this shell object to a ball by volumetric harmonic map (see (c) and (d)). The outer boundary is tessellated into truncated icosahedron and the shell space is segmented into 10 layers (see (e)). (f), (g) and (h) show the 3rd, 6th and 10th layers respectively.	44
4.7	Hex shell meshing results of Moai model.	46
4.8	Experimental results on genus-0 models	47
4.9	Experimental results on high genus models	48

List of Tables

4.1	Meshing results	43
4.2	Meshing quality results	43

Chapter 1

Introduction

1.1 Volume data

In computer graphics, volume data means a large set of voxel data acquired by CT, MRI, or MicroCT scanners. The number of voxels contained in a single volume data can be easily in millions. Considering the huge size of data, the efficiency of processing volume data voxel by voxel is low, and usually, high performance hardware support is required. For the goal to save the computation cost, researchers have developed methods to represent the volume data in a better way. For example, in volume rendering, a volume is viewed by extracting surface of equal isovalue from the volume and rendering them as polyhedral meshes or by rendering the volume directly as a block of data. The algorithm *Marching Cubes* [28] is always used to extract the surface from volume data. In this thesis, we intend to use isotropic tetrahedral meshes to represent the volume data, as show in Fig 1.1. The isosurfaces are extracted from the volume data, and the tetrahedral meshes are constructed from these isosurfaces. The isosurfaces are expected to have intersection-free and watertight characters. Usually, the extracted isosurfaces are in low quality. To improve the quality of them, surface remeshing techniques are required.

Figure 1.1: Volume data of skull model. From left to right: scanned data; isosurfaces; surface remeshing result; and tetrahedral mesh.

1.2 Tetrahedral mesh generation

Tetrahedral mesh generation aims at tiling a bounded 3D domain with tetrahedra so that any two of them are either disjoint or sharing a lower dimensional face (triangle). For the volume data, isotropic tetrahedral meshes are widely used to represent the space it enclosed. Many papers [16, 1, 58] have been published on the isotropic tetrahedral mesh generation. Generally, there are two major approaches: Delaunay triangulation methods and advancing front methods. The Delaunay triangulation is based on the empty sphere property. The mathematical criterion is a major backbone of the Delaunay approach. It enables high quality meshing and relatively fast processing runtime. The empty sphere property is, however, very sensitive to truncation errors in practical computations in 3D. For examples, more than five points can sometimes be located on almost the same sphere. To resolve this problem, Shewchuk proposed an efficient algorithm for arbitrary precision floating-point arithmetic [49].

The boundary recovery is another issue needed to be resolved in the Delaunay approach. The empty sphere property does not ensure that the surface boundary preserves the original connectivity. The constrained Delaunay approach [6] cannot be easily extended to 3D complex domains. There are sometimes needs to generate a set of volume meshes for multi-connected domains, for example, the skull model has two components

Figure 1.2: Tetrahedral mesh generation of an ellipsoid. Left: input surface. Right: output tetrahedral mesh.

and with different genus. For parallel mesh generation, it is common to decompose the entire domain into a number of sub-domains and to tetrahedralize each sub-domain simultaneously. In this case, the shared internal boundaries must be identical. Although they can be changed during the parallel mesh generation process, the parallel efficiency often deteriorates due to the required communication between the processors.

The advancing front approach is based on the generation of tetrahedra by marching a front toward the interior. An initial front is usually defined as a water-tight surface mesh. Elements are created on the front, which is achieved by adding new points in the interior of the domain. This enables the generation of elements in variable size with desired stretching. Local mesh density near the initial front (i.e. boundaries) can be controlled easily. In addition, the topology of the initial front can be naturally preserved. The disadvantages of this approach are slow computational speed due to geometric search during the process and the lower quality of resulting meshes.

Combination of the advantages of the Delaunay and the advancing front approaches is widely used [34, 31, 29]. Our algorithm starts with a Delaunay triangulation of a set of boundary nodes. This is used as a background mesh. New nodes are then added by the

advancing front approach. This combined approach can shorten the runtime and produce high quality meshes. Local node density can be controlled easily. However, it still inherits the drawback of the Delaunay approach. Although the initial Delaunay triangulation is not difficult in 2D, the surface recovery is often troublesome in 3D. Because Delaunay mesh triangulates the convex hull of vertices, and it does not conform (match exactly) to the input domain boundary.

1.3 Mesh Quality

The success of geometry processing, scientific visualization and finite elements methods based on the meshes whose elements have right shapes and sizes. A few bad elements can limit the accuracy and speed of those applications. The quality metrics for tetrahedral meshes, hex meshes or more general meshes had been studied during the past decades, for example, P.M. Knupp [19] discussed the quality metrics for structured and unstructured mesh in general. Shewchuk [51] discussed the quality measures for tetrahedral mesh. In this thesis, several quality metrics are used for measuring the quality of tetrahedral meshes and hexahedral meshes.

For tetrahedral meshes, we use the *radius ratio* [1] of a tetrahedron instead of the popular *radius* metric and *radius-edge ratios* [38]. The *radius ratio* is the quotient of inscribed and circumscribed sphere radii (times three for normalization purpose). Low value means low quality of a tetrahedron.

To measure the quality of the hexahedral mesh, we choose the scaled Jacobian metric [20], the condition number of the Jacobian matrix [17, 18] and the Oddy metric [35].

For a vertex of a hexahedron the metrics are formed as follows. Assume $x \in R^3$ is the position vector of this vertex and $x_i \in R^3$ for $i = 1, 2, 3$ are its three neighbors in some fixed order. Edge vectors are defined as $e_i = x_i - x$ with $i = 1, 2, 3$ and the Jacobian matrix is $J = [e_1, e_2, e_3]$. The determinant of the Jacobian matrix is called *Jacobian*, or

scaled Jacobian if edge vectors are normalized. An element is said to be *inverted* if one of its Jacobians ≤ 0 . We use the *Frobenius norm* as a matrix norm, $|J| = (\text{tr}(J^T J))^{1/2}$. The condition number of the Jacobian matrix is defined as $k(J) = |J||J^{-1}|$, where $|J^{-1}| = \frac{|J|}{\det(J)}$. Therefore, the three quality metrics for a vertex in a hexahedron are defined as follows:

$$\begin{aligned} \text{Jacobian}(x) &= \det(J) \\ k(x) &= \frac{1}{3}|J^{-1}||J| \\ \text{Oddity}(x) &= \frac{(|J^T J|^2 - \frac{1}{3}|J|^4)}{\det(J)^{\frac{4}{3}}} \end{aligned}$$

1.4 Motivation

Numerical analysis and computation in geometry modeling or mechanics always require a high quality tetrahedral mesh to represent the volume space. As introduced, there is no perfect solution for automatic mesh generation that can produce very high quality tetrahedral meshes. In this thesis, we aim at developing high quality tetrahedral meshes to fulfil the requirement of volume parameterization. The quality of parameterization is related to the quality of generated tetrahedral meshes. Especially, when we do numerical analysis on the volume vector field which is the parameterization result, the success of tracing vector flow highly depends on the quality of vector field. Regarding this requirement, we have to develop tetrahedral mesh generation method manually by combining existing methods and apply mesh quality optimization methods on it. In this thesis, we split the mesh generation into several processes. Different mesh generation and refinement techniques are applied to create high quality tetrahedral meshes. The resulted meshes will be used for volumetric data processing in this thesis.

1.5 Research Scope

This research work focuses on volumetric data processing by using volume parameterization methods. Parameterization is hard problem for surfaces and volumes [30]. Unlike surface parameterization which has been widely used in many applications [11, 47], volume parameterization is still under study and its applications are limited. The research scope of this thesis can be catalogued into two topics.

- **Applications of volume parameterization.** In this thesis, two volume parameterization methods are introduced. The first one is about parameterizing a star shaped volume to a sphere ball. The second one is about parameterize a shell space. Different applications are developed for this two methods.
- **Quality tetrahedral generation.** In practical, our volume data are represented by tetrahedral meshes. For volume parameterization, the basic parameterizing unit is a tetrahedron. For a given volume, the parameterization result is a vector field inside the volume space. The quality of this vector field (e.g. smooth) is determined by the quality of tetrahedral mesh. In consequence, quality tetrahedral mesh generation method is highly demanded. In this thesis, latest isotropic tetrahedral meshing methods are investigated first. After that, several improvements are developed to satisfy the quality requirements of volume parameterization methods. The detail of these improvements will be introduced in the following chapters, based on different applications.

1.6 Objectives

The objectives of this research are to develop high quality mesh generation method to fulfill the requirements of volumetric data processing. Two applications are implemented

to show the correctness and robustness of our developed methods. Specifically, the objectives are described as follows.

- **Anisotropic solid texture transfer.** Synthesis of solid texture is a very time consuming process. If there are two different geometry models, we have to synthesize the solid texture twice. To address this problem, we try to reduce the number of synthesis and allow smooth transfer of anisotropic solid texture between two star-shaped volumes. Firstly, we develop a mesh generation method to tiling the star-shaped volumes. Secondly, we apply the star-shaped parameterization method to build a bijective mapping between two star-shaped volumes. Lastly, we transfer the texture according to the parameterized volumes.
- **Hexahedral shell mesh construction.** We try to develop a hex meshing method for the shell structures based on the polycube map and volume harmonic map. The input of this method is a water-tight and closed 2-manifold in arbitrary topology. The output is a layered all-hex shell mesh with user-specified thickness and number of layers. We first map the 2-manifold to a polycube. We then build two shell space for the manifold and the polycube. We would like to name them as "object space" and "polycube space," respectively. Next, we generate tetrahedral meshes in these two spaces and parameterize them. Finally, we build a hex mesh in the polycube space naturally and map this hex mesh back to the object space via the volumetric mapping.

1.7 Organization

The rest of the thesis is organized as follows. In Chapter 2, a brief review of tetrahedral mesh generation methods and other related techniques are given. In Chapter 3, we develop a tetrahedral mesh generation method by combining advancing front method

and Delaunay triangulation optimization method. The geometry of the resulted meshes are limited to star-shapes. A volume parameterization method is used to develop the application of anisotropic solid texture transfer. In Chapter 4, we implement a method to generate isotropic tetrahedral meshes for the shell space and propose a hexahedral mesh construction method via volume parameterization. The Chapter 5 concludes this thesis and several future works are discussed.

Chapter 2

Related Work

In this chapter, we give a brief review of the existing tetrahedral mesh generation techniques and introduce the related parameterization and texture mapping algorithms to be used in the next chapters.

2.1 Tetrahedral meshing techniques

Creating high quality tetrahedral meshes from volume data is a difficult task for a variety of reasons. First, the representing of resulting meshes requires robust, disciplined data structures and algorithms. There are also basic mathematical difficulties which make tetrahedral meshing significantly harder than its 2D counterpart: the regular tetrahedron, the most isotropic 3D simplex, does not tile 3D space (let alone specific domains), while the equilateral triangle does tile the plane; Dealing with the boundaries is also fundamentally more difficult in 3D: while it always exists a 2D triangulation conforming to any set of non intersecting constraints, this is no longer true in 3D [50]. These facts make both the development of algorithms and suitable error analysis for the optimal 3D meshing problem very challenging. In this section, we will give a short discussion about the volumetric remeshing skills.

Advancing front. Advancing front is a very popular family of triangle and tetrahedral mesh generation algorithms. Two of the main contributors to this method are Rainald

Lohner [27, 26] and S. H. Lo [25]. Starting from the boundary of the domain, new vertices are added by a local heuristic to ensure that the generated tetrahedra have acceptable shapes and sizes and conform to the desired sizing field. Global optimization steps can also be performed sporadically to improve the mesh quality further. A number of variants exist, such as sphere or bubble packing [23], which provide better tetrahedral shape and size control albeit adding a significant computational overhead.

Octree-based methods. The Octree technique was primarily developed in the 1980s by Mark Shephard’s [48] group at Rensselaer. An octree is first refined until each of its leaves is either strictly inside or strictly outside of a finely voxelized version of the domain. Proper connections of the interior leaves through, for instance, a red-green strategy [33] then ensure a good initial mesh of the domain, usually improved through optimization or physically-based relaxation in particular to better approximate the domain boundary. Other similar methods offer bounds of worst dihedral angles even without a relaxation stage [32]. Unfortunately, octree-based meshes have preferred edge directions, which may be detrimental to subsequent use in simulation.

Figure 2.1: Octree-based methods of bunny model [41].

Delaunay approaches. For a given set of sample points in 3D, its Delaunay triangulation has the canonical property of minimizing the maximum radius of the minimum

containment sphere. This property is very useful in approximation theory: this radius provides an upper bound on the L^∞ difference between any function f and its piecewise linear approximate, assuming f has bounded second derivatives. Thus a Delaunay triangulation provides good control over the worst interpolation error inside a domain. Consequently a large body of work in numerical analysis provides error estimates for a variety of applications using these meshes. Because of these as well as many other optimality properties, mesh generation relying on Delaunay triangulation such as Delaunay refinement, unit mesh, or centroidal Voronoi tessellations[10] have flourished in the meshing and Computational Geometry communities. Delaunay refinement methods offer some theoretical guarantees on the resulting meshes: they provide bounds on the radius-edge ratio, and are shown to be asymptotically optimal with respect to the number of elements in the mesh. Delaunay refinement, however, can generate slivers; some attempts have been made to handle the sliver problem within Delaunay refinement [5, 22]. Unfortunately the theoretical guarantees are quite poor, and the mesh either is no longer Delaunay but a regular (weighted Delaunay) triangulation, or comes with degraded bounds on the radius-edge ratio.

Figure 2.2: Delaunay approaches for tetrahedral meshing by using constrained delaunay triangulation [9].

Mesh optimization techniques. Even if fast and robust Delaunay triangulators are

available, the previous strategies can require substantial implementation effort to make them robust to arbitrary input domains. A large number of practical meshing techniques employ local optimization methods which move vertices adjacent to poorly-shaped tetrahedrons to improve mesh quality. Coupled with local face swapping between adjacent tetrahedrons as well as tetrahedron insertions and deletions, these strategies can result in nice final meshes [8, 12]. Unfortunately, these optimizations often use highly non-convex functionals and get easily stuck in local minima.

Figure 2.3: Variational tetrahedral meshing approach of bunny model with adapted tetrahedra. Left: input surface. Right: resulted tetrahedral mesh [1].

Variational tetrahedral meshing. Variational tetrahedral meshing (VTM) was proposed by Alliez [1]. It is a method based on Delaunay-based optimization techniques. Unlike other methods, it does not optimize mesh quality as an after-thought, but integrates the optimization in the construction procedure. The pivotal idea of VTM is to optimize the quality of the elements and to work on boundary conformance in a single iterative process. The steps of VTM likes following: firstly, the boundary is meshed and then the interior is meshed, conforming to the boundary. This means the mesh on the boundary evolves during the optimization process. However, the recovery of the boundary, especially for the sharp feature, still need to be improved. In [53], the author proposed an enhanced version of VTM algorithm which can preserve the feature edges

and let generated tetrahedral mesh can be used for Finite Element Analysis (FEA) and other numerical analysis.

2.2 Polycube map

The polycube, a natural generalization of the cube, can serve for the parametric domain of shapes with complicated topology and geometry. Tarini et al presented a method to construct polycube map by projecting the vertices of the 3D model to the polycube domain [56]. Wang et al introduced an intrinsic approach that first maps the 3D model and the polycube to the canonical domains and then seek the map between them [61]. The resulting polycube map is guaranteed to be a bijection. Want et al. proposed the user-controllable polycube map where the users can specify the pre-images of the polycube corners [62]. Lin et al. proposed an automatic algorithm to construct polycube map [24]. He et al. proposed a divide-and-conquer approach to constructing polycube map of arbitrary topology [13].

Figure 2.4: Polycube map of bunny model. Left: input bunny surface; Right: output polycube [56].

2.3 Parameterization of Star-shaped volume

A volume M is called a *star shape* if there exists a point $c \in M$ such that any ray casted from c intersects the boundary of M only once. The point c is called the center of M . In particular, any convex volume is a star shape, where any interior point can serve as the center. Parameterization of a star-shaped volume was proposed by [14]. They parameterize star shapes by using Green’s function and proved that the Green’s function can induce a diffeomorphism between two star-shaped volumes.

2.4 Lapped solid textures

Solid textures [39, 40] are textures in 3D and have several notable advantages over 2D textures. First, many natural materials, such as wood and stone, may be more realistically modeled using solid textures. Second, solid textures obviate the need for finding a parameterization for the surface of the object to be textured, which is a challenging problem in itself. In fact for objects of general topology it is not possible to find a parameterization that avoids seams and/or distortion. Although these problems maybe alleviated by synthesizing directly on the surface of an object(e.g. [59, 60, 65, 66]), they cannot be avoided altogether.

Furthermore, solid textures provide texture information not only on surfaces, but also throughout the entire volume occupied by a solid object. This is a highly convenient property, as it makes it possible to perform high-fidelity sub-surface scattering simulations, as well as break objects to pieces and cut through them.

Lapped solid textures in 3D case was proposed by [55]. Unlike the classic approaches of created solid texture: procedural approach, run-time 2D texture synthesis on cross-section and example-based 3D solid texture synthesis, this method represents solid objects with spatially-varying oriented textures by repeatedly pasting solid texture exemplars.

Figure 2.5: Lapped textures. From left to right: kiwi model, carrot model, tree model [55].

The underlying concept is to extend the 2D texture patch-pasting approach of Lapped textures [43] to 3D solids using a tetrahedral mesh and 3D textures patches. The system places texture patches according to the user-defined volumetric tensor fields over the mesh to represent oriented textures. This method also extends the original technique to handle nonhomogeneous textures for creating solid models whose textural patterns change gradually along the depth fields.

2.5 Hexahedral elements meshing and shell structure

Hexahedral meshing has been widely studied in the past two decades. From the literature, hex meshing can be broadly classified into the direct method and the indirect method [37]. Direct methods generate a hex mesh directly from a solid. Indirect methods first subdivide a solid into a tetrahedral mesh; the tetrahedral mesh is then converted to a hex mesh. One approach is to subdivide each tetrahedral into four hexahedral elements. Even though this approach generates all-hex mesh, the quality will be poor.

Popular techniques include grid-based algorithm [45], plastering [54], whisker weaving algorithm [57], embedded Voronoi graph [46], chordal surface [44], just name a few. The readers are referred to the comprehensive survey of general hexahedral and tetrahedral meshing construction. There are also a few techniques that aim to improve the quality

of hexahedral mesh, such as [68] [67].

Shell is defined as the area between two nearly parallel surfaces of the same topology. Shell structure is widely used in computer graphics applications. Porumbescu et al. [42] presented an algorithm to build a bijective map between shell space and texture space that can be used to generate small-scale features on surfaces. Wang et al. [63] presented technique for rendering heterogenous translucent materials by solving diffusion equation in the shell space.

Chapter 3

Anisotropic Solid Texture Transfer for Star-Shaped Volume

Solid textures are an efficient instrument to compactly represent both the external and internal appearance of 3D objects, providing practical advantages with respect to classical 2D texturing. There are three main approaches to creating solid texture models: the procedural approach, run-time 2D texture synthesis on cross-sections, and example based 3D solid texture synthesis. However, these generation approaches are computation costly.

3D texture transfer means to transfer solid texture from one object to another object. For anisotropic solid textures, this procedure is also a computing tense one as it requires the correlation analysis between two objects. For the reason to save computing time, in this chapter, we develop a method to make anisotropic solid texture transferring smoothly between two star-shaped volumes without correlation analysis. This method uses the advantage of volume parameterization techniques and broadens the applicability of the lapped solid texture results to a larger pool of geometric models. Firstly, we give a brief introduction to the solid texture. Next, a detail description of tetrahedral mesh for the star shapes is given. Finally, several solid texture transfer examples are provided to show the correctness of our proposed method.

3.1 Background

Anisotropic solid textures [55], allow to fill the interior of 3D models with spatially-varying and anisotropic texture patterns. Takayama et al. [55] proposed a lapped texture approach [43] to synthesize anisotropic solid textures by pasting solid texture exemplars [21] repeatedly over the tetrahedron structure of 3D geometries. This approach can result in high-quality and large-scale solid textures with low computation cost. To create such a texture, the user, however, has to mark up volumetric tensor field and edit the texture in a geometry-dependent fashion.

Solid texture generation or synthesis is a time consuming process as the texel count grows cubically with the spatial resolution. Usually, when transfer texture from a source object to a target object, we may need to capture the correlation between two geometry first, and then synthesize the solid texture for the target object. Correlation analysis calculates a set of geometric features between two objects, such as curvature, and the observed diffuse texture. Obviously it is a cost computing procedure.

Figure 3.1: Anisotropic solid textures of star shapes. From left to right: kiwi, watermelon, carrot.

For the goal to reduce the computing cost, we try to use the parameterization tools to find a mapping between the source object and the target object, limited to star shaped object right now. Considering the parameterization method used here is diffeomorphism, the ongoing texture transfer can be easily done via barycentric coordinates and mapping.

3.2 Tetrahedral Mesh Generation

In this section, the procedure of tetrahedral mesh generation is presented. So far, we only have the boundary surface mesh of the model. The tetrahedral mesh will be generated from the surface mesh by using the existing unstructured tetrahedral mesh generation algorithms. The input surface mesh should be intersection-free and watertight.

The algorithms of [52] are used here for the initial unstructured tetrahedral mesh generation. The density, or the number of sample points of this initial tetrahedral mesh is less than what we expect. Considering this requirement, subdivision method has to be employed to increase the density, and the optimization method was used to improve the quality of the mesh.

Flow of Mesh generation. The whole procedure of the isotropic tetrahedral mesh generation can be described by the following flow chart 3.2:

Unstructured tetrahedral mesh generation. We will use Tetgen to generate an initial constrained Delaunay mesh here. The radius-edge ratio is set to 1.414.

Subdivision. For the application of anisotropic solid texture, the number of tetrahedra has to be large enough. We use a very simple scheme that splits each tetrahedral into eight small tetrahedrons. The scheme can be described as follows:

- (i) Step 1: Splits a tetrahedron into four tetrahedra and an octahedron.
- (ii) Step 2: The octahedron is quadrisected into four tetrahedra by choosing one diagonal.

There was a special processing in our isotropic meshing. That is, all the boundary tetrahedra will not be subdivided, as we do not want to change the boundary surface mesh.

Figure 3.2: Flow chart of tetrahedral mesh generation.

Figure 3.3: Inside of Star model by cutting view.

Figure 3.4: Subdivision scheme.

Figure 3.5: Subdivision of a tetrahedron.

A subdivision result can be shown in Figure 3.5.

Optimization. In the optimization step, the tetrahedral meshes are remeshed into optimal Delaunay triangulations. We are using the variational approach here. Variational approaches (that is, method relying on energy minimization) have been advocated as a powerful and robust tool in meshing both in graphics for triangle [15] and tetrahedral [8, 33] meshes. These methods basically define non-convex energies that they minimize through vertex displacements and/or connectivity changes in the current mesh.

The Energy function used here is based on Chen [1] :

$$E_{ODT} = \|f - f_{PWL}^{primal}\|_L^1 \quad (\text{Eq. 3.1})$$

The volume between a paraboloid and an overlaid, circumscribing piecewise linear approximate f_{PWL}^{primal} formed by a linear interpolation of points on the paraboloid. Chen made the observation that changing the energy from E_{CVT} [10] to E_{ODT} amounts to only a slight change. That is:

$$E_{CVT} = \sum_{i=1 \dots n} \int_{V_i} \|x - x_i\|^2 dx \quad (\text{Eq. 3.2})$$

where x_i is the vertex position, and V_i is a local cell associated with each x_i .

$$E_{ODT} = \frac{1}{n+1} \sum_{i=1 \dots n} \int_{\Omega_i} \|x - x_i\|^2 dx \quad (\text{Eq. 3.3})$$

The integral is now taking over each *1-ring region* Ω_i . Notice that these regions overlap. These quadratic energies differ quite significantly: Chen's E_{ODT} energy measures a quality of the *simplicial mesh*, not of its dual. It is thus more prone to generate well-shaped primal elements, while E_{CVT} was maximizing the compactness of the dual Voronoi cells.

Our variational approach is based on the consistently energy minimizing algorithm proposed in [4]. In [4], the minimization of the energy E_{ODT} is not just through a smoothing procedure, but through a full-blown minimization procedure for both vertex positions and connectivity.

Figure 3.6: PWL approximations.

- (i) Optimizing connectivity. We optimize the connectivity by using the Delaunay triangulation. The optimal connectivity which minimizes E_{ODT} just as it is optimal

for E_{CVT} [43] Therefore, we compute the (global) Delaunay connectivity systematically, guaranteeing optimality of the connectivity at each iteration.

(ii) Optimizing vertex positions. The update function of the vertex position is:

$$x_i^* = x_i - \frac{1}{|\Omega_i|} \sum_{T_j \in \Omega_i} |T_j| c_j \quad (\text{Eq. 3.4})$$

where c_j is the circumcenter of tetrahedron T_j . This term shows that, although we move each vertex to a local average, the optimal placement heavily depends on the local distribution.

Mesh extraction. After a sufficient number (usually 50 60, depends on the given threshold of energy) of optimizations steps, the mesh representing the model has to be extracted from the resulting Delaunay triangulation, which covers the convex hull of the nodes. As the input domain boundary is not necessary to be convex one, we have to remove the tetrahedra that are located outside the domain. This process is called "Peeling".

We follow the mesh extraction rules that discussed in [53]. An adaptive way of peel off which can be described as follows:

- (i) All tetrahedra are assumed to be inside at the start. On inspection we can decide to remove one.
- (ii) All tetrahedra that have at least one interior point as one of its four nodes are inside, and therefore will never be removed. We thus only consider removing those tetrahedra that have four boundary nodes.
- (iii) If the centroid of a tetrahedron having four boundary nodes falls outside the control mesh, then the tetrahedron is directly considered outside.

- (iv) The only tetrahedra we are left with are those having four boundary nodes and the centroid inside the control mesh. Assuming that a correct boundary is present, we must conclude that these tetrahedra are inside; a tetrahedron that has its centroid inside the mesh is either completely inside or it intersects the boundary. Only if a tetrahedron has a volume-length ratio smaller than 0.1, we add it to a list to be considered for a clean-up peeling.
- (v) Clean-up: we want to remove very flat tetrahedra that might lie on the boundary and can be removed without negatively affecting the quality of the boundary representation. We try to remove the tetrahedra iteratively, by inspecting for removal only those tetrahedra that are currently considered on the boundary. If such a tetrahedron has two or more faces on the current boundary, it is removed. If it has only one face on the current boundary, it is only removed if its volume-length ratio is smaller than 10^{-4} . This process is continued until no more tetrahedra can be removed.

3.3 Parameterization of star shapes

A volume M is called a *star shape* if there exists a point $c \in M$ such that any ray cast from c intersects the boundary of M only once. The point c is called the center of M . In this section, we introduce the parameterization method of star shapes. This parameterization method is described in He. et al.'s paper [14]. Given two star shape M_0, M_1 , a bijective mapping can be built by this method. First, M_0 and M_1 are parameterized into two unit balls: B_0^3, B_1^3 . Next, considering B_0^3 and B_1^3 have same geometry (same math description), a bijective mapping is existed between these two balls. Then, a bijective mapping is existed between M_0 and M_1 as the volume parameterization method is diffeomorphism [14]. The detail of this volume parameterization method can be described as follow.

Figure 3.7: Mesh extraction of star model.

Algorithm 1: Ball parameterization of star shapes.

Input: S , the boundary mesh of a star-shaped volume M

Output: $f : M \rightarrow \mathbb{B}^3$ is diffeomorphism

- 1 Find the center of M ;
 - 2 Compute the Green's function on M , $G_M : M \rightarrow \mathbb{R}$;
 - 3 Map the center $\mathbf{c}$ to the center of $\mathbb{B}^3$, $f(\mathbf{c}) = \mathbf{0}$;
 - 4 Parameterize the boundary points by constructing a conformal spherical mapping $\phi : \partial M \rightarrow \partial \mathbb{B}^3$;
 - 5 **for** every interior vertex $\mathbf{p} \in M$
 - 6 Trace the integration curve γ from $\mathbf{p}$ to the boundary point $\mathbf{q} \in \partial M$;
 - 7 Set $f(\mathbf{p}) = \frac{\phi(\mathbf{q})}{G_M(\mathbf{p})+1}$
 - 8 **end for**
 - 9 $\mathbb{B}^3$ — unit ball in $\mathbb{R}^3$ $\partial \mathbb{B}^3$ — surface of $\mathbb{B}^3$ ∂M — surface of volume M
-

An illustration result of this parameterization is in Figure 3.8.

Figure 3.8: Parameterization of star shape

3.4 Texture transfer

After building of volume mapping, texture transfer can be archived by tracing integral curve and barycentric coordinates. The source object has been pre-synthesized with lapped solid texture. We parameterize the source object and the target object to a common solid ball. With this common parametric ground, we can transfer the synthesized texture information from the source geometric model to the target model.

Algorithm 2: Solid Texture transfer

Input: tetrahedra mesh S_{tet} with pre-synthesized solid texture $S_{texture}$; star shaped tetrahedral mesh T_{tet}

Output: transfer the solid texture $S_{texture}$ from S_{tet} to T_{tet}

- 1 *Parameterize S_{tet} to a unit ball B_{tet} with unit radius;*
- 2 *Parameterize T_{tet} to a unit ball B'_{tet} with unit radius;*
- 3 **for** every vertex v of B'_{tet}
- 4 *Locate the tetrahedron tet in B_{tet} that contain vertex v ;*
- 5 *Calculate the barycentric coordinate of v in Tet ;*
- 6 *Calculate the texture coordinate of v by using the texture coordinates of Tet 's four vertices;*
- 7 **end for**

3.5 Experimental Results

This section showcases the experimental results of solid texture transfer. From the following figure, we can see that the anisotropic texture transfer smoothly among dog head model, moai model, star model and venus head model. The source target is a pre-synthesized Kiwi model.

3.6 Summary

In this chapter, we presented an anisotropic solid texture transfer method for star shapes. This technique bases on the proposed mesh generation method and employs volume parameterization techniques. We combine the advancing front meshing techniques and Delaunay optimization algorithms to generate high quality tetrahedral meshes. We split the mesh generation into several steps, such as unstructured tetrahedral mesh generation, subdivision, optimization and post-processing. Our proposed tetrahedral mesh generation method can be successfully applied to star shape volume tetrahedralization and volume parameterization. This can be seen by the illustrated examples.

Our solid texture transfer approach allows the reuse of synthesized anisotropic solid textures without incurring additional texture synthesis. This improvement can avoid

Figure 3.9: Anisotropic solid texture transfer.

repeatedly texture synthesizing works if the input models are star shapes. Furthermore, because our volume parameterization method is a diffeomorphism in theory, the bijectivity and smoothness in the texture transfer process can be guaranteed.

Chapter 4

Hexahedral Shell Mesh Construction Based on Tetrahedral Mesh

Shells are three-dimensional structures where one dimension, the thickness, is much smaller than the other two dimensions. Shell structures can be widely found in many real-world objects. In this chapter, we present a method to construct layered hexahedral mesh for shell objects. Given a closed 2-manifold and the user-specified thickness, we construct the shell space using the distance field and then parameterize the shell space to a polycube domain. The volume parameterization induces the hexahedral tessellation in the object shell space. As a result, the constructed mesh is an all-hexahedral mesh that most of the vertices are regular, i.e., the valence is 6 for interior vertices and 5 for boundary vertices. The mesh also has a layered structure that all layers have exactly the same tessellation. We prove our parameterization is guaranteed to be a homeomorphism. As a result, the constructed hexahedral mesh is free of degeneracy, such as self-intersection, flip-over, etc. Furthermore, our volume parameterization has low angle distortion, in particular, we show that the iso-parametric line (in the thickness dimension) is orthogonal to the other two iso-parametric lines. We demonstrate the efficacy of our method by applying to models of various topology.

4.1 Background

Finite element analysis is an essential tool to model various scientific and engineering phenomena, such as structural mechanics, heat flow, computational fluid dynamics, etc. An important requirement of the numerical approximation and simulation is to convert the solid model to a discrete mesh composed of smaller elements. The most common types of elements are tetrahedral and hexahedral elements. A 3D domain cannot always be meshed into hexahedral elements. But, it can be decomposed into tetrahedral elements more easily when compared to hexahedral elements. Thus, tetrahedral elements gain more popularity in finite element analysis. However, there are certain applications that hexahedral elements are more preferred than tetrahedral elements. For example, tetrahedral meshes typically require 4-10 times more elements than a hexahedral mesh to obtain the same level of accuracy [7]. In nonlinear elastic-plastic analysis, the linear hexahedral elements may be superior to even quadratic tetrahedral elements when shear stress is dominant [2].

Constructing hexahedral meshes is usually more challenging than tetrahedral meshes [3]. In this paper, we focus on the shell objects that are three-dimensional structures that the thickness is much smaller than the other two dimensions. These structures are widely used in manufacturing, such as automobile bodies, sheet metal parts, etc. To construct a hexahedral shell mesh, the existing method, e.g., [44], computes the chordal surface by cutting the mesh of the input CAD model at its mid plane, and then constructs a quadrilateral mesh for the chordal surface. Finally a two-way mapping between the chordal surface and the boundary is used to sweep the quad elements from the chordal surface onto the boundary, resulting in a layered all-hex mesh. This method works well for shell with simple geometry and constant thickness such that the top and bottom surface are similar. However, it may fail on models with complex geometry/topology and variable thickness which the chordal surface is hard to compute. Furthermore, there is

no guarantee that the resulting hexahedral mesh is free of degeneracy, such as flip-over, self intersection, etc.

4.2 Theory foundation

In this section, we prove the shell space parameterization is a homeomorphism and the w -isoparametric line (in the thickness dimension) is perpendicular to the other two isoparametric lines.

Theorem. Given two shelled objects $M = M_0 - M_1$ and $P = P_0 - P_1$ where M_0 (P_0) and M_1 (P_1) are the outer and inner boundary surfaces of M (P) respectively. The boundary surfaces M_i , P_i , $i = 1, 2$, are of the same topological type.

Define harmonic function on M , $f : M \rightarrow \mathbb{R}$, $\Delta f = 0$, with Dirichlet boundary condition, $f|_{M_0} = 0$ and $f|_{M_1} = 1$. Let $C_f : M_0 \times [0, 1] \rightarrow M$ be the integral curve of the gradient field ∇f such that given an arbitrary point $v_0 \in M_0$, $C_f(v_0, 0) = v_0$, $C_f(v_0, 1) = v_1$ and $C_f(v_0, t) = v_t$, where $v_1 \in M_1$ is the other ending point and $v_t \in M$ is the point satisfying $f(v_t) = t$. Similarly, we define the harmonic function on P , $g : P \rightarrow \mathbb{R}$ and the integral curve $C_g : P_0 \times [0, 1] \rightarrow P$.

Define a homeomorphic boundary map $h : M_0 \rightarrow P_0$ and construct the volume parameterization $\phi : M \rightarrow P$ as follows:

$$\phi(C_f(v_0, t)) = C_g(h(v_0), t), \quad \forall v_0 \in M_0.$$

Then the volume parametrization ϕ has the following properties:

- (i) ϕ is homeomorphic.
- (ii) the w -isoparametric line, following the gradient of the harmonic field (i.e., in the thickness dimension), is always perpendicular to the u - and v -isoparametric lines that span the iso-surfaces of the harmonic field.

Proof of (1): First, we show the ending points of each integral curve are on the inner and outer boundary surfaces respectively. Note f and g are smooth functions, their gradient vector fields are curl-free. Thus, no integral curve can form a loop inside the volume.

Second, we show that no integral curve that starts and ends on the same boundary surface. The function is harmonic and there is no critical points (where the gradient vanishes) inside the volume. Thus, the function value is strictly monotonic along the integral curve. Note that all points on the same boundary surface have the same function value, so the ending points of each integral curve must be on different boundary surface.

Third, we show that two integral curves do not intersect. Assume two integral curves $\gamma_1 \in M$ and $\gamma_2 \in M$ intersect at a point p . Then p is a critical point and the gradient ∇f vanishes at p . We consider two cases:

Case 1: p is an interior point. Since f is harmonic, the maximum and minimum must be on the boundaries. Therefore the Hessian matrix at p has negative eigenvalue values. Suppose $f(p) = s$, then according to Morse theory, the homotopy types of the level sets $f^{-1}(s - \epsilon)$ and $f^{-1}(s + \epsilon)$ will be different. At all the interior critical points, the Hessian matrices have negative eigenvalues, the homotopy type of the level sets will be changed. The changes of the homotopy type can not be canceled out. Therefore, the homotopy type of M_0 is different from that of M_1 . This contradicts the given condition.

Case 2: p is on the boundary. Without loss of generality, say $p \in M_0$. Then we can glue two copies of the same volume, along M_0 . And reverse the gradient field of one volume. The union of the two gradient fields give us a harmonic function field. Then there is no interior critical point on the doubled volume. p becomes one interior critical point, that leads to a contradiction.

Therefore γ_1 and γ_2 have no intersection points anywhere.

Last, we show ϕ is homeomorphic. From the above, we know that for an arbitrary interior point, there is a unique integral curve passing through and intersecting on the

Figure 4.1: Shell space parameterization

inner and outer boundaries. The two ending points are also unique. Thus, C_f and C_g are homeomorphisms. The given boundary map $h : M_0 \rightarrow P_0$ is homeomorphic, thus, it induces a homeomorphism between integral curves in M and P , $C_f(v_0, \cdot) \rightarrow C_g(h(v_0), \cdot)$, which in turns induces the heomorphic map ϕ .

Proof of (2): As shown in 1), ϕ bijectively maps the integral curve $\gamma \in M$ to a unique integral curve $\gamma' \in P$. Furthermore, ϕ bijectively maps every iso-surface of f to the iso-surface of g with the same iso-value. Note that the integral curve follows the gradient vector field, thus, is orthogonal to the iso-surface.

The w -isoparametric line for a fixed starting point $v_0 \in M_0$, $\phi(C_f(v_0, t))$, $t \in [0, 1]$, is in fact an integral curve. The u - and v - isoparametric lines for a fixed parameter t , $\phi(C_f(v_0, t))$, $v_0 \in M_0$, span the iso-surface (with iso-value t) of the harmonic function. Thus, the w -isoparametric line is orthogonal to u - and v - isoparametric lines.

4.3 Hexahedral Shell Meshing

4.3.1 Overview

Our hexahedral meshing algorithm consists of three steps, constructing the shell space, parameterizing the shell space and hexahedral meshing.

In step 1, we construct the offset surface using the user-specified thickness. Then we tessellate the shell space by a tetrahedral mesh. We also parameterize the given boundary surface to a polycube and construct the shelled polycube in a similar fashion.

In step 2, we compute the harmonic field in the shell spaces by solving the Laplace equation with Dirichlet boundary condition. By tracing the integral curves, we build a bijection between the shell spaces.

In step 3, we tessellate the polycube space by regular hexahedral mesh, and then construct the layered hexahedral mesh in the object space by the volume parameterization.

Let M and P denote the shell spaces of the given model and polycube respectively. Let $\partial M = M_0 \cup M_1$ where M_0 and M_1 are the outer and inner boundary surfaces. Similarly, P_0 and P_1 denote the outer and inner boundary surface of P .

The detailed algorithm is illustrated as follows:

Algorithm 3: Layered Hexahedral Meshing for Shell Objects

Input: M_0 , a closed 2-manifold

P_0 , the polycube with the same topology of M_0

d : the thickness

Output: H , a quality hexahedral mesh of M

- 1 Construct the polycube map $\phi : M_0 \rightarrow P_0$
 - 2 Create the offset surfaces M_1 and P_1 by the user-specified thickness d
 - 3 Construct isotropic tetrahedral meshes for M and P
 - 4 Parameterize M to P by volumetric harmonic field (see Alg. 2)
 - 5 Form the hexahedral mesh of P
 - 6 Construct the hexahedral mesh of M using the volume parameterization
-

4.3.2 Isotropic tetrahedral remeshing

The shell space will be represented by quality isotropic tetrahedral mesh. To generate high quality isotropic tetrahedral mesh for arbitrary topology geometry is a tough task and, to the author's knowledge, there is no perfect solution. The isotropic tetrahedral meshing method which was introduced in [1]. This method can generate high quality tetrahedral mesh, however, it has deficiencies that make it unsuitable for the generation of meshes of mechanical models for finite element analysis, or for the polycubes here which has shape boundary. The reason is that the boundary was represented incorrectly. In [53], the author try to solve this by increasing the quadrature samples around each node of boundary. Node splitting and enhanced mesh extraction method are also applied to preserve the boundary.

4.3.3 Constructing Shell Space

The shell space is enclosed by two disconnected closed surfaces, one of which is the given 3D model, and the other is an offset surface. The offset distance is specified by the user. we use the MPU method [36] to construct the distance field of the input model and then extract the isosurface whose isovalue is the user-specified thickness.

In Figure 4.2, Row 1: we construct a distance field for the given model and extract an iso-surface with the user-specified offset distance. Then, we construct an isotropic tetrahedral mesh using variational meshing technique [1]. Row 2: we map the outer boundary surface to a polycube and construct the tetrahedral mesh of the shell space in a similar way. Note the sharp features (polycube edges and corners) are well preserved in the tetrahedral mesh.

Following the divide-and-conquer approach [13], we map the boundary surface M_0 to a user-constructed polycube. The resulted polycube map has very low angle distortion and is guaranteed to be a bijection. When constructing the tetrahedral mesh for the polycube

Left: Boundary surface. Center: Shell space. Right: Close-up view.

Figure 4.2: Shell space construction.

space, we must pay special attention to the sharp features (such as edges and corners) of the polycube. To preserve the features in the isotropic tetrahedral mesh, we follow the variant variational meshing technique that is tailored for mechanical models [53]. Figure 4.2 shows the construction of shell space for the Skull model and its polycube domain.

4.3.4 Parameterizing Shell Space

To construct a map between $M = M_0 - M_1$ and $P = P_0 - P_1$, we first solve volumetric harmonic map for each shell mesh. The harmonic equations $f : M \rightarrow \mathbb{R}$ and $g : P \rightarrow \mathbb{R}$ are defined as follows:

$$\Delta f = 0 \text{ with } f|_{\partial M_0} = 0 \text{ and } f|_{\partial M_1} = 1$$

$$\Delta g = 0 \text{ with } g|_{\partial P_0} = 0 \text{ and } g|_{\partial P_1} = 1$$

Since the shells M and P are represented by tetrahedral meshes, we solve the above Laplace equations by finite element method [64].

The tetrahedral mesh M is represented by $M = (V, E, F, T)$ where V , E , F and T are the set of vertices, edges, faces and tetrahedral respectively. P is represented in the similar way. For every interior vertex v_i , the harmonic function $f : M \rightarrow \mathbb{R}$ satisfies the condition

$$\sum_{v_j \in Nb(v_i)} w_{ij}(f(v_j) - f(v_i)) = 0,$$

where w_{ij} is a weight assigned to edge e_{ij} . Suppose edge e_{ij} is shared by m adjacent tetrahedra, it lies against m dihedral angles θ_k , $k = 1, \dots, m$. Then the weight w_{ij} for e_{ij} can be defined as $w_{ij} = \frac{1}{12} \sum_{k=1}^m \|e_{ij}\| \cot \theta_k$, where $\|e_{ij}\|$ is the length of edge e_{ij} . Figure 4.3 shows the volume rendering of the harmonic fields in the Skull model.

Once we obtain the harmonic function, the gradient vector field is computed as follows: Suppose t is a tetrahedron with vertices $(v_1, \dots, v_4)$, the face on the tetrahedron against vertex v_i is f_i . We define n_i to be the vector along the normal of f_i with magnitude equalling twice the area of f_i . Then, the gradient of ∇f in t is a constant vector field

$$\nabla f = f(v_0)n_0 + f(v_1)n_1 + f(v_2)n_2 + f(v_3)n_3.$$

We then define the per-vertex gradient as the average of the per-tetrahedron gradient vectors.

Algorithm 4: Shell object parameterization**Input:** An arbitrary point $v \in M$ in the shell object $f : M \rightarrow \mathbb{R}$: harmonic function on M $g : P \rightarrow \mathbb{R}$: harmonic function on P $h : M_0 \rightarrow P_0$ the homeomorphism between the outer boundaries of M and P .**Output:** The image $\phi(v) \in P$ in the polycube domain

- 1 *Starting from v , trace the integral curve $\gamma \in M$ in both positive and negative directions of the gradient vector field ∇f . γ intersects the outer and inner boundaries at v_0 and v_1 respectively.*
- 2 *Compute $v'_0 = h(v_0) \in P_0$*
- 3 *Starting from v'_0 , trace the integral curve $\gamma' \in P$ following the positive direction of the gradient vector field ∇g . γ' intersects P_1 at v'_1*
- 4 *Locate the unique point $v' \in \gamma'$ such that $g(v') = f(v)$. Then $\phi(v) = v'$*

Given the gradient vector field ∇f , the integral curve is a curve such that the tangent vector to the curve at any point v along the curve is precisely the vector $\nabla f(v)$. In the Appendix, we show that each integral curve has unique ending points, one on the inner boundary, the other on the outer boundary. Furthermore, any two integral curves do not intersect.

In Figure 4.3, we solve a Laplace equation in the shell space with Dirichlet boundary condition such that the values of the outer and inner boundary surfaces are 0 and 1, respectively. We use volume rendering to visualize the harmonic field in the shell space.

We construct the volume parameterization $\phi : M \rightarrow P$ as follows: for every interior point $v \in M$, let $\gamma \in M$ be the integral curve that passes through v and follows the gradient vector field of f . The integral curve γ intersects M_0 and M_1 at v_0 and v_1 respectively. Let $\gamma' \in P$ be the integral curve in P that starts from $v'_0 = h(v_0) \in P_0$, follows the gradient vector field of g , and terminates at $v'_1 \in P_1$. The image of v , $v' = \phi(v) \in \gamma'$ is a unique point such that $g(v') = f(v)$. In the Appendix, we prove that the map ϕ is homeomorphic.

Figure 4.3: Volumetric harmonic map on the shell space.

The key component in our volume parameterization is to trace the integral curves in the volumes. We use half-face data structure to model the tetrahedral mesh. Each interior face is shared by two tetrahedra and each boundary face is only adjacent to one tetrahedron. We demonstrate the integral curve tracing in Figure 4.4. The detailed tracing algorithm is shown as follows:

Algorithm 5: Integral Curve Tracing

Input: p , a point inside the volume or on the boundary
 ϵ , step length of tracing
 $positive$, a boolean value indicating the tracing direction
Output: γ , the integral curve passing through p and following the positive or negative direction of the gradient vector field

- 1 *Find the tetrahedron $currTet$ such that $p \in currTet$;*
- 2 **while** $currTet \neq NULL$ **do**
- 3 *Compute the gradient vector p_v by linear interpolation of the vertex gradients of $currTet$*
- 4 **if** $positive$ **then**
- 5 $p_{next} = p + \epsilon * p_v$
- 6 **else**
- 7 $p_{next} = p - \epsilon * p_v$
- 8 **end**
- 9 **if** $p_{next} \in currTet$ **then**
- 10 update p_v
- 11 **else**
- 12 *Find the tetrahedron $nextTet$ which contains the p_{next}*
- 13 $currTet = nextTet$
- 14 **end**
- 15 $p = p_{next}$
- 16 **end**
- 17 *Project p to the shell boundary*

4.3.5 Layered Hexahedral Meshing

Note the polycube domain can be easily tessellated into quadrilaterals. We can construct a hexahedral mesh by sweeping the quads such that each vertex is moving along (or opposite to) the normal direction until they reach the other boundary surface. Then we uniformly segment the polycube domain into layers. The number of layers are specified by the user. Note that each layer has exactly the same tessellation. As the quadrilateral mesh of the outer boundary is constructed by the polycube map, all vertices, except the polycube corners, are regular, i.e., with valence 4. Therefore, after sweeping the quads to the shell space, the interior vertex are regular (i.e., with valence 6) if the corresponding

Figure 4.4: Tracing integral curves. Each integral curve follows the gradient vector field of the harmonic function. Thus, it is orthogonal to the iso-surface of the harmonic function (including the two boundary surfaces). We show that each integral curve has unique ending points and all integral curves do not intersect.

vertex on the boundary mesh is regular. We should also mention that the shelled polycube domain does not need to have the same thickness as the 3D shell object, as long as the two boundary surfaces of the polycube are similar.

As we show in the Appendix, the proposed shell parameterization algorithm is guaranteed to be a homeomorphism. Thus, the constructed hexahedral mesh is free of degeneracy, such as self-intersection and flip over. Figure 4.5 shows the layered hexahedral mesh of the Skull model.

Figure 4.5: Hexahedral meshing of the Skull model (three layers). Left column: 1/4 cutting view; Right column: view of three layers from two different directions.

4.4 Experimental Results

We tested our algorithm to 3D models of various topology. In our experiments, the user-specified offset distance (i.e., thickness) can be either positive or negative. The user also specifies the number of layers in the constructed hexahedral meshes. As mentioned before, each layer has exactly the same tessellation. Figure 4.8 and 4.9 show the layered hexahedral meshes. Cutaway views are used to show the high quality of the hexahedral inside the models. Table 4.1 shows the experimental results.

To measure the quality of the hexahedral mesh, we choose *scaled Jacobian* metric

[20] in our paper. Given a hexahedron, let $x_0 \in \mathbb{R}^3$ be a vertex and $x_i \in \mathbb{R}^3$, $i = 1, 2, 3$ its three neighbors. Edge vectors are defined as $e_i = x_i - x_0$, and the Jacobian matrix is $J = [e_1, e_2, e_3]$. The determinant of the Jacobian matrix $|\det(J)|$ measures the volume of the parallelepiped spanned by e_1 , e_2 and e_3 . The edge lengths are normalized so that the more closer of scaled Jacobian to one, the better aspect ratio of the hexahedron, and the less volume distortion of the parameterization. We compute the average of the scaled Jacobian for the constructed hexahedral meshes. As shown in Table 4.1, Table 4.2 and Figure 4.5, 4.8, 4.9, the constructed meshes are visually pleasing and with high quality.

Table 4.1: Meshing results

Model	Genus	Tetrahedral mesh ($ V $, $ T $)	# Layers	# of hexahedra in each layer	Thickness	para time #Secs	tracing time #Secs
Bunny	0	(180K, 839K)	3	7232	-0.02	18.17	6.75
Skull	0	(190K, 981K)	3	4804	-0.04	29.58	8.98
Moai	0	(180K, 974K)	5	2688	-0.08	33.72	10.09
Eight	2	(180K, 921K)	3	12800	-0.02	26.23	23.89
Squirrel	0	(180K, 839K)	5	8384	+0.02	19.88	13.04
Decocube	4	(180K, 850K)	4	7936	+0.03	21.97	14.81
Rockerarm	1	(200K, 943K)	3	4672	+0.02	20.28	4.36
Isidore horse	0	(200K, 956K)	2	4736	-0.01	22.31	2.47
Duck	0	(100K, 542K)	10	11520	NA	19.67	23.20

Table 4.2: Meshing quality results

Model	scaled Jacobian (best, aver., worst)	Oddy Metric (best, aver., worst)	Condition Number (best, aver., worst)
Bunny	(1.00, 0.94, 0.05)	(7.27, 8.57, 520.78)	(2.73, 3.14, 15.15)
Skull	(1.00, 0.94, 0.58)	(2.97, 5.28, 29.65)	(1.75, 1.99, 2.92)
Moai	(1.00, 0.92, 0.16)	(0.01, 2.18, 307.82)	(1.00, 1.22, 10.21)
Eight	(1.00, 0.94, 0.26)	(0.14, 2.85, 79.36)	(1.02, 1.48, 3.89)
Squirrel	(1.00, 0.97, 0.32)	(0.24, 2.99, 62.99)	(1.04, 1.62, 6.87)
Decocube	(1.00, 0.96, 0.70)	(0.41, 1.74, 13.04)	(1.07, 1.36, 3.31)
Rockerarm	(1.00, 0.92, 0.25)	(0.14, 2.95, 45.77)	(1.02, 1.41, 4.34)
Isidore horse	(1.00, 0.94, 0.05)	(3.38, 4.58, 280.15)	(1.89, 2.12, 15.74)
Duck	(1.00, 0.98, 0.85)	(0.02, 0.97, 46.55)	(1.00, 1.09, 2.72)

Remark In this chapter, we parameterize the shell space to a polycube domain with a goal to construct a hexahedral mesh. In fact, the proposed shell space parameterization

works for arbitrary parametric domain as long as it has the same topology as the input model. Furthermore, the inner boundary surface M_1 and P_1 could be arbitrary closed surface rather than the offset surface. The only requirement is that M_1 and P_1 are of the same topological type as M_0 and P_0 . In Figure 4.6 (a), we conduct an experiment by embedding the Duck model into a sphere. Thus, M_0 and M_1 are geometrically different but topologically equivalent. We construct the shelled ball as the parametric domain and tessellate the sphere using truncated icosahedron (i.e., the soccer ball tessellation with 12 pentagons and 20 hexagons). We then increase the resolution of the sphere by Catmull-Clark subdivision and sweep the quads inwards the ball. Figure 4.6 (e)-(f) shows the layered hexahedral mesh of the embedded Duck model.

Figure 4.6: Layered hexahedral meshing of the Duck model embedded in a sphere. Note the outer and inner boundaries are significantly different (see (a)). (b) shows the isotropic tetrahedral mesh of the shell space. We parameterize this shell object to a ball by volumetric harmonic map (see (c) and (d)). The outer boundary is tessellated into truncated icosahedron and the shell space is segmented into 10 layers (see (e)). (f), (g) and (h) show the 3rd, 6th and 10th layers respectively.

4.5 Summary

In this chapter, we developed an algorithm to parameterize the shell space. The parameterization is theoretically sound and guarantees a bijection. By parameterizing the given shell object to a shelled polycube domain, we can construct layered all-hexahedral meshes with high quality. All vertices (except the vertices on the integral curves passing polycube corners) are regular, i.e., with valence 6 for interior vertices, and valence 5 for boundary vertices. Furthermore, each layer in the constructed mesh has exactly the same tessellation. Due to the bijectivity of the proposed volume parameterization, the hexahedral mesh is free of degeneracy, such as self-intersection, flip-over, etc. We demonstrated that our method can be applied to 3D models of various topology.

Figure 4.7: Hex shell meshing results of Moai model.

Figure 4.8: Experimental results on genus-0 models

48
Figure 4.9: Experimental results on high genus models

Chapter 5

Contribution and Future Work

5.1 Contribution

In this thesis, we had developed two high quality mesh generation methods for volumetric data processing. Both methods could generate high quality meshes to fulfill the requirements of volume data processing. The first one was an isotropic tetrahedral mesh generation method. The accomplished mesh generation method was designed for star-shaped volumes specially. Moreover, we developed an anisotropic texture transfer application based on our proposed mesh generation method and a volume parameterization method. The second method was about hexahedral mesh generation in solid shell structures. We applied our proposed method to shell structures that have different genus and topologies. The experiment results showed that our mesh generation method was correct and robustness.

The contribution of this thesis can be summarized into two parts. In the first part, we develop an application for star shaped volume parameterization method. This application allows smooth transfer of anisotropic solid textures between two star-shaped volumes. At the same time, a quality tetrahedral mesh generation method which combines the advancing front method and optimization method is developed. Moreover, due to the advantages of volume parameterization method, our method can avoid the repeat texture synthesise works. In the second part, we proposed a layered all-hex meshing method by

using the volumetric harmonic mapping and polycube map technique. Our method was proved to be able to generate layered all-hex meshes. The experiment results demonstrate that our proposed method can generate high quality hex meshes. Moreover, our method is suitable for arbitrary topology. The math proof of homeomorphism of shell space mapping is also provided.

5.2 Future Work

Our proposed methods and algorithms have several limitations. Some future works can be discussed as follows.

For the anisotropic solid texture transfer part, the current framework only applies to star-shaped volumes. However, most real-world shapes are not star-shaped. One possible solution to parameterize volumes of arbitrary topology and geometry is to segment the shape into a set of disjoint star shapes, then parameterize each shape individually, and finally glue patches together with a certain order of continuity. As a future direction, we will develop automatic techniques to facilitate the segmentation and gluing procedures.

For the hex meshing of shell structure part, the following limitations should be considered. First, we choose the polycube as the parametric domain due to its regular structure that one can easily construct an all-hexahedral mesh. On one hand, the polycube should mimic the geometry of the shell object as close as possible to minimize the parameterization distortion. On the other hand, since each polycube corner is a singularity of the boundary surface parameterization, we should keep the polycube as simple as possible. These two requirements often contradict to each other, thus, it requires the users to be very skillful in designing the parametric domain. Second, polycube is not a good parametric domain for models with highly complex geometry and/or topology, such as trees. Thus, our method works only for limited range of models. Third, we use finite element method to solve the Laplace equation in the shell space and then trace the integral curves

of the gradient vector field. The robustness of tracing integral curves highly depends on the quality of the tetrahedral mesh. In our experiments, we observed that the isotropic meshes lead to good results. However, the isotropic tetrahedral meshes usually contain large number of vertices, which increases the computational cost. To address these problems, there are several interesting works in the future. Firstly, it is worthy to study the general rule of creating polycube mesh which is suitable to build a shell space in polycube domain. Different types of corners on polycube surface should be examined. Secondly, the robustness of integral curve tracing algorithm could be improved by testing with more tetrahedral models. In the end, we can apply our shell parameterization method to the applications of shell maps.

References

- [1] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. *ACM Trans. Graph.*, 24(3):617–625, 2005.
- [2] S. E. Benzley, E. Perry, K. Merkley, B. Clark, and G. Sjaardema. A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *Proceedings of 4th International Meshing Roundtable*, pages 179–191, Albuquerque, NM, 1995.
- [3] T. Blacker. Meeting the challenge for automated conformal hexahedral meshing. In *Proceedings 9th International Meshing Roundtable*, pages 11–20, New Orleans, CA, 2000.
- [4] L. Chen and J. Xu. Optimal delaunay triangulations. *Journal of Computational Mathematics*, 22(2):299–308, 2004.
- [5] S.-W. Cheng and T. K. Dey. Quality meshing with weighted delaunay refinement. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 137–146, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.
- [6] L. P. Chew. Constrained delaunay triangulations. *Algorithmica*, 4:97–108, 1989.
- [7] A. O. Cifuentes and A. Kalbag. A performance study of tetrahedral and hexahedral elements in 3-D finite element structural analysis. *Finite Elem. Anal. Des.*, 12(3-4):313–318, 1992.
- [8] B. Cutler, J. Dorsey, and L. McMillan. Simplification and improvement of tetrahedral models for simulation. In *SGP '04: Proceedings of the 2004 Eurographics/ACM*

REFERENCES

- SIGGRAPH symposium on Geometry processing*, pages 93–102, New York, NY, USA, 2004. ACM.
- [9] T. K. Dey. Delaunay mesh generation of three dimensional domains. Technical report, Ohio State University, OSU-CISRC-9/07-TR64, 2007.
- [10] Q. Du and D. Wang. Tetrahedral mesh generation and optimization based on centroidal voronoi tessellations. *International Journal on Numerical Methods in Engineering*, 55:1355–1373, 2003.
- [11] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*, pages 157–186. Springer, 2005.
- [12] L. Freitag and C. Ollivier-gooch. A comparison of tetrahedral mesh improvement techniques. In *Proceedings of 6th International Meshing Roundtable*, pages 87–100, Park City, UT, 1996.
- [13] Y. He, H. Wang, C.-W. Fu, and H. Qin. A divide-and-conquer approach for automatic polycube map construction. *Comput. Graph.*, 33(3):369–380, 2009.
- [14] Y. He, J. Xia, S. Han, C.-W. Fu, and X. Gu. Parameterization of star shaped volumes using green’s functions. In *Proceedings of Geometric Modeling and Processing (GMP 10)*, LNCS 6130, pages 219–235, CIEM, Castro Urdiales, Spain, 2010.
- [15] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *SIGGRAPH ’93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 19–26, New York, NY, USA, 1993. ACM.
- [16] Y. Ito, A. M. Shih, and B. K. Soni. Reliable isotropic tetrahedral mesh generation based on an advancing front method. In *Proceedings 13th International Meshing Roundtable*, pages 95–105, Williamsburg, VA, Sandia National Laboratories, 2004.
- [17] P. Knupp. Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part i - a framework for surface mesh optimization. *Int. J. Numer. Meth. Engng*, 48:401–420, 2000.

REFERENCES

- [18] P. Knupp. Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part ii - a framework for volume mesh optimization and the condition number of the jacobian matrix. *Int. J. Numer. Meth. Engng*, 48:1165–1185, 2000.
- [19] P. Knupp. Algebraic mesh quality metrics. *SIAM J. Sci. Comput.*, 23(1):193–218, 2001.
- [20] C. Kober and M. Muller-Hannemann. Hexahedral mesh generation for the simulation of the human mandible. In *Proceedings of the 9th International Meshing Roundtable*, pages 423–434, New Orleans, CA, 2000.
- [21] J. Kopf, C.-W. Fu, D. Cohen-Or, O. Deussen, D. Lischinski, and T.-T. Wong. Solid texture synthesis from 2d exemplars. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3):2:1–2:9, 2007.
- [22] X. Li. *Sliver-free Three Dimensional Delaunay Mesh Generation*. PhD thesis, University of Illinois at Urbana-Champaign, 2000.
- [23] X. Li and A. Teng, S.H.and Ungor. Biting: Advancing front meets sphere packing. *Int. J. on Num. Methods in Eng.*, 49:501–511, 1991.
- [24] J. Lin, X. Jin, Z. Fan, and C. C. L. Wang. Automatic polycube-maps. In *GMP’08: Proceedings of the 5th international conference on Advances in geometric modeling and processing*, pages 3–16, Berlin, Heidelberg, 2008. Springer-Verlag.
- [25] S. H. Lo. Volume discretization into tetrahedra-ii, 3d triangulation by advancing front approach, computers and structures. *Computers and Structures*, 39:493–500, 1991.
- [26] R. Lohner. Progress in grid generation via the advancing front technique. *Engineering with Computers*, 12:186–210, 1996.
- [27] R. Lohner, P. Parikh, and C. Gumbert. Interactive generation of unstructured grid for three dimensional problems. In *Numerical Grid Generation in Computational Fluid Mechanics’88*, pages 687–697, Miami, Florida, USA, 1988.

REFERENCES

- [28] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM.
- [29] D. L. Marcum and N. P. Weatherill. Unstructured grid generation using iterative point insertion and local reconnect. *AIAA Journal*, 33:1619–1625, 1995.
- [30] T. Martin, E. Cohen, and R. Kirby. Volumetric parameterization and trivariate b-spline fitting using harmonic functions. *Computer Aided Geometric Design*, 26:648–664, 2009.
- [31] D. J. Mavriplis. Abstract an advancing front delaunay triangulation algorithm designed for robustness. *Journal of Computational Physics*, 117:90–101, 1995.
- [32] S. Mitchell and S. Vavasis. Quality mesh generation in higher dimensions. *SIAM J. Sci. Comput.*, 29:1334–1370, 2000.
- [33] N. Molino, R. Bridson, J. Teran, and R. Fedkiw. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *Proceedings of the 12th International Meshing Roundtable*, pages 103–114, Santa Fe, NM, 2003.
- [34] J. Muller, P. L. Roe, and H. Deconinck. A frontal approach for internal node generation in delaunay triangulations. *International Journal for Numerical Methods in Fluids*, 17:241–255, 2005.
- [35] A. Oddy, J. Goldak, M. McDill, and M. Bibby. Distortion metric for isoparametric finite elements. *Transactions of the Canadian Society for Mechanical Engineering*, 12:213–217, 1988.
- [36] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 463–470, New York, NY, USA, 2003. ACM.
- [37] S. J. Owen. A survey of unstructured mesh generation technology. In *Proceedings of 7th International Meshing Roundtable*, pages 239–267, Dearborn, MI, 1998.

REFERENCES

- [38] V. N. Parthasarathy, C. M. Graichen, and A. F. Hathaway. A comparison of tetrahedron quality measures. *Finite Elem. Anal. Des.*, 15(3):255–261, 1994.
- [39] D. R. Peachey. Solid texturing of complex surfaces. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 279–286, New York, NY, USA, 1985. ACM.
- [40] K. Perlin. An image synthesizer. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 287–296, New York, NY, USA, 1985. ACM.
- [41] M. Pharr and R. Fernando. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley Professional, 2005.
- [42] S. D. Porumbescu, B. Budge, L. Feng, and K. I. Joy. Shell maps. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 626–633, New York, NY, USA, 2005. ACM.
- [43] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 465–470, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.
- [44] W. Quadros and K. Shimada. Hex-layer: Layered all-hex mesh generation on thin section solids via chordal surface transformation. In *Proceedings of 11th International Meshing Roundtable*, pages 169–182, Ithaca, NY, 2002.
- [45] R. Schneiders, R. Schindler, and F. Weiler. Octree-based generation of hexahedral element meshes. In *Proceedings of 5th International Meshing Roundtable*, pages 205–215, Pittsburgh, PA, 1996.
- [46] A. Sheffer, M. Etzion¹, A. Rappoport, and M. Bercovier. Hexahedral mesh generation using the embedded voronoi graph. *Engineering with computers*, 15:248–262, 1999.

REFERENCES

- [47] A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.*, 2(2):105–171, 2006.
- [48] M. Shephard and M. Georges. Automatic three-dimensional mesh generation technique by the finite element octree technique. *International Journal for Numerical Methods in Engineering*, 32:709–749, 1991.
- [49] J. R. Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete & Computational Geometry*, 18:305–363, 1996.
- [50] J. R. Shewchuk. *Delaunay refinement mesh generation*. PhD thesis, Computer Science Department, Camegie Mellon University, 1997.
- [51] J. R. Shewchuk. What is a good linear finite element? - interpolation, conditioning, anisotropy, and quality measures. In *Proceedings of the 11th International Meshing Roundtable*, pages 115–126, Sandia National Laboratories, 2002.
- [52] H. Si and K. Gaertner. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proceedings of 14th International Meshing Roundtable*, pages 147–163, San Diego, CA, 2005.
- [53] M. S. Smit and W. F. Bronsvort. Variational tetrahedral meshing of mechanical models for finite element analysis. *Computer-Aided Design and Applications*, 5:1–4, 2008.
- [54] M. L. Staten, R. A. Kerr, S. J. Owen, and T. D. Blacker. Unconstrained paving and plastering: Progress update. In *Proceedings of 15th International Meshing Roundtable*, pages 469–486, Birmingham, AB, 2006.
- [55] K. Takayama, M. Okabe, T. Ijiri, and T. Igarashi. Lapped solid textures: filling a model with anisotropic textures. *ACM Trans. Graph.*, 27(3):1–9, 2008.
- [56] M. Tarini, K. Hormann, P. Cignoni, and C. Montani. Polycube-maps. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 853–860, New York, NY, USA, 2004. ACM.

REFERENCES

- [57] T. J. Tautges, T. Blacker, and S. A. Mitchel. Whisker weaving: Invalid connectivity resolution and primal construction algorithm. In *Proceedings of 4th International Meshing Roundtable*, pages 115–127, Albuquerque, NM, 1995.
- [58] J. Tournois, C. Wormser, P. Alliez, and M. Desbrun. Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–9, New York, NY, USA, 2009. ACM.
- [59] G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 289–298, New York, NY, USA, 1991. ACM.
- [60] G. Turk. Texture synthesis on surfaces. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 347–354, New York, NY, USA, 2001. ACM.
- [61] H. Wang, Y. He, X. Li, X. Gu, and H. Qin. Polycube splines. In *SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 241–251, New York, NY, USA, 2007. ACM.
- [62] H. Wang, M. Jin, Y. He, X. Gu, and H. Qin. User-controllable polycube map for manifold spline construction. In *SPM '08: Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 397–404, New York, NY, USA, 2008. ACM.
- [63] J. Wang, S. Zhao, X. Tong, S. Lin, Z. Lin, Y. Dong, B. Guo, and H.-Y. Shum. Modeling and rendering of heterogeneous translucent materials using the diffusion equation. *ACM Trans. Graph.*, 27(1):1–18, 2008.
- [64] Y. Wang, X. Gu, and S.-T. Yau. Volumetric harmonic map. *Communications in Information and Systems*, 3(3):191–202, 2004.
- [65] L. Wei and M. Levoy. Texture synthesis over arbitrary manifold surfaces. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 355–360, New York, NY, USA, 2001. ACM.

REFERENCES

- [66] L. Ying, A. Hertzmann, H. Biermann, and D. Zorin. Texture and shape synthesis on surfaces. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 301–312, London, UK, 2001. Springer-Verlag.
- [67] Y. Zhang and C. Bajaj. Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. *Computer Methods in Applied Mechanics and Engineering*, 195(9-12):942 – 960, 2006.
- [68] Y. Zhang, R. Bajaj, and G. Xu. Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. In *Proceedings of 14th International Meshing Roundtable*, pages 449–468, San Diego, CA, 2005.

Publications

Shuchu Han, Jiazhi Xia, and Ying He. Hexahedral shell mesh generation by volumetric parameterization. *In SPM '10: Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, pp. 127-136, 2010.

Jiazhi Xia, Ying He, Shuchu Han, Chi-Wing Fu, Feng Luo, and Xianfeng Gu. Parameterization of Star Shaped Volumes Using Green's Functions. *In Proceedings of Geometric Modeling and Processing (GMP'2010)*, LNCS 6130, pp. 219-235, 2010.

Jiazhi Xia, Ying He, Xiaotian Yin, Shuchu Han, and Xianfeng Gu. Direct-Product Volumetric Parameterization of Handlebodies via Harmonic Fields. *In Proceedings of IEEE International Conference on Shape Modeling (SMI'10)*, pp. 3-12, 2010.