2013

Geometric modeling and optimization over regular domains for graphics and visual computing

Shenghua Wan
Louisiana State University and Agricultural and Mechanical College

GEOMETRIC MODELING AND OPTIMIZATION OVER REGULAR DOMAINS FOR GRAPHICS AND VISUAL COMPUTING

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The School of Electrical Engineering and Computer Sciences

by
Shenghua Wan
B.S., Harbin Institute of Technology, 2009
M.S., Louisiana State University, 2011
December 2013

To my family and friends

Acknowledgements

At LSU, I have been very fortunate to meet so many friends and colleagues. This dissertation would not have been made possible without their kind advice and help.

Above all, I want to express my sincerest appreciation to my advisor, Prof. Xin Li, for his persistent guidance and patience throughout my PhD study. His knowledge and inspiration really navigate this work to the destination. Without his encouragement, this work would have never been completed. It is truly my honor being his student.

Meanwhile, I would also like to convey my gratitude to my PhD committee members, Prof. Hongchao Zhang, Prof. Jagannathan Ramanujam, Prof. Bahadir Gunturk, and Prof. Ayman Okeil. They have been very supportive and patient, and providing constructive advices on this dissertation. I also want to thank Prof. Hongchao Zhang for the precious discussion on the optimization problems and solvers.

I want to acknowledge Prof. Sun Chang who help me gain hands-on experience in high performance computing and more importantly shares a lot of valuable life wisdom with me.

Furthermore, I would like to express my special thanks to all friends in Geometric and Visual Computing Group. I am really grateful to Wuyi Yu, Huanhuan Xu, Kang Zhang, Zhao Yin, Li Wei and Tengfei Ye for the collaboration on research projects. Their generous willingness and patience to share the ideas and discuss all kinds of problems are greatly appreciated. Many thanks to Wei Yu, Xiao Lin, Fei Zhao, Ning Zhang, Yang Chen, Peizhi Chen and Jinfang Zhou for helping me accommodate to visiting life in Xiamen University.

Moreover, I want to appreciate my friends, Qing Huang, Dongsheng Guan, Chuan Cai, Bixiang Tang, Youwen Gong and many others. My life would suffer without them.

Last but not the least, I am deeply indebted to my cherished family who always trust and support me through the years, especially my beloved wife. I would not have been able to focus on the study and chase my dream without their persistent sustainment.

This dissertation is dedicated to my family and friends for their generous support and encouragement.

Table of Contents

List of Tables

List of Figures

Abstract

The effective construction of parametric representation of complicated geometric objects can facilitate many design, analysis, and simulation tasks in Computer-Aided Design (CAD), Computer-Aided Manufacturing (CAM), and Computer-Aided Engineering (CAE). Given a 3D shape, the procedure of finding such a parametric representation upon a canonical domain is called geometric parameterization. Regular geometric regions, such as polycubes and spheres, are desirable domains for parameterization. Parametric representations defined upon regular geometric domains have many desirable mathematical properties and can facilitate or simplify various surface/solid modeling and processing computation.

This dissertation studies the construction of parameterization on regular geometric domains and explores their applications in shape modeling and computer-aided design. Specifically, we studies (1) the surface parameterization on the spherical domain for closed genus-zero surfaces; (2) the surface parameterization on the polycube domain for general closed surfaces; and (3) the volumetric parameterization for 3D-manifolds embedded in 3D Euclidean space. We propose novel computational models to solve these geometric problems. Our computational models reduce to nonlinear optimizations with various geometric constraints. Hence, we also need to explore effective optimization algorithms. The main contributions of this dissertation are three-folded. (1) We developed an effective progressive spherical parameterization algorithm, with an efficient nonlinear optimization scheme subject to the spherical constraint. Compared with the state-of-the-art spherical mapping algorithms, our algorithm demonstrates the advantages of great efficiency, lower distortion, and guaran-

teed bijectiveness, and we show its applications in spherical harmonics decomposition and shape analysis. (2) We propose a first topology-preserving polycube domain optimization algorithm that simultaneously optimizes polycube domain together with the parameterization to balance the mapping distortion and domain simplicity. We develop effective nonlinear geometric optimization algorithms dealing with variables with and without derivatives. This polycube parameterization algorithm can benefit the regular quadrilateral mesh generation and cross-surface parameterization. (3) We develop a novel quaternion-based optimization framework for 3D frame field construction and volumetric parameterization computation. We demonstrate our constructed 3D frame field has better smoothness, compared with state-of-the-art algorithms, and is effective in guiding low-distortion volumetric parameterization and high-quality hexahedral mesh generation.

1 Introduction

The effective construction of parametric representation of complicated geometric objects can facilitate many design, analysis, and simulation tasks in Computer-Aided Design (CAD), Computer-Aided Manufacturing (CAM), and Computer-Aided Engineering (CAE). Given a 3D shape, the procedure of finding such a parametric representation upon a canonical domain is called geometric parameterization. Regular geometric regions, such as polycubes and spheres, are desirable domains for parameterization. Parametric representations defined upon regular geometric domains have many desirable mathematical properties and can facilitate or simplify various surface/solid modeling and processing computation.

1.1 Motivation and Contribution

The parametric representations of 3D geometric shapes can be defined upon various geometric regions. For example, surfaces with non-trivial topology can be parameterized onto an atlas of local charts (topological disks) or another curved surface with the same geometry. In this dissertation, we study the choice of a type of canonical parametric domains that have the same topology with the given model and a (near-) homogeneous geometry. We call such geometric domain as a *regular domain* in shape parameterization. Regular domains are favorable parametric domains due to their simplicity and regularity, parametric representation defined upon which reduces the complexity to construct mathematical models and simulate the real-word processes in CAD, CAE and CAM, compared to the random curved complicated shapes. Furthermore, geometric modeling and processing over regular domains, e.g. parametrization, provides key-enabling technologies for many critical tasks. Unfortunately,

except for some rare theoretical cases, the parametrization in practice inevitably introduces angle distortion or area distortion or some combination of both. A good parametrization favored by the downstream applications is the one which minimizes these distortions to some extent. Generally, regular domains offer seamless parametrization with low distortion.

Overview of this dissertation. In this paper, we study the parameterizations over regular domains and propose algorithms to minimize the distortion of the parameterizations. First, We explore surface parameterization over the regular domains, i.e. sphere and polycube. We look for the parameterization with optimized distortion over the two regular domains and optimized domain shape for polycube domain. Then the research is generalized from 2D surfaces to 3D solids. Hence we study volumetric parameterization guided by a frame field over polycube domain. The smoothness of the frame field has a significant impact on the distortion of the parameterization. Therefore, we investigate frame field optimization to improve the smoothness.

1.1.1 Spherical Parametrization

A large portion of real-world 3D geometric models are bounded by closed genus-0 surfaces, for which the sphere is the most natural parametric domain. Parametrization over the planar domain has been studied for decades, but constructing the parameterization of a genus-0 surface via these planar domains requires to cut the surface into topological disks. The cutting seam will introduce large distortion in the resultant parametrization and hence artifacts in the subsequent modeling and simulation tasks. The spherical domain provides opportunity for seamless parametrization with low distortion. Spherical parametrization is key-enabling technology for a lot of applications including shape analysis using spherical harmonics, compression, morphing and etc.

However, obtaining good spherical parametrization is challenging in practice. First, the computation algorithm must be robust to guarantee the bijectivity. Second, computing optimal spherical parametrization is often formulated as a non-linear optimization problem,

whose solving is time-consuming. Moreover, local minima existing in the non-linear optimization are difficult to overcome.

In the first part of this thesis, we develop an effective hierarchical optimization scheme to compute spherical parametrization. Among all existing state-of-the-art spherical mapping methods, the main advantage of our method are two folded.

- It generates bijective and lowly distorted mapping results.
- The algorithm converges very efficiently, therefore it is suitable for handling huge geometric models.

We also demonstrate and analyze the effectiveness of our mapping in spherical harmonics decomposition and shape analysis.

1.1.2 Polycube Mapping

Besides the sphere, the polycube is another type of regular domains. Polycube is the surface of a solid that consists of a few solid cubes (see Figure 1.1). Polycube mapping was first introduced by Tarini et al [6]. It parameterizes a closed surface onto a polycube domain. A polycube has the same topology of the input surface, and it is usually constructed to approximate the geometry of the surface. Therefore, the surface parametrization on a polycube domain often allows much lower distortion than that over a planar domain. Meanwhile, the polycube domain still possesses great regularity: (a) each sub-patch is a rectangle; (b) transitions between adjacent patches are simple rotations and translations except on corner points. With these advantages, the polycube mapping has been used in many graphics and shape modeling applications such as texture mapping [6] and synthesis [7], shape morphing [8], spline construction [9, 10], volumetric matching [11, 12] and etc.

Intuitively, on one hand, the more cubes one uses to construct the polycube, the better the domain can approximate the original model, which brings the parametrization very small area and angle distortion. However, this introduces more corner points, which are singularity

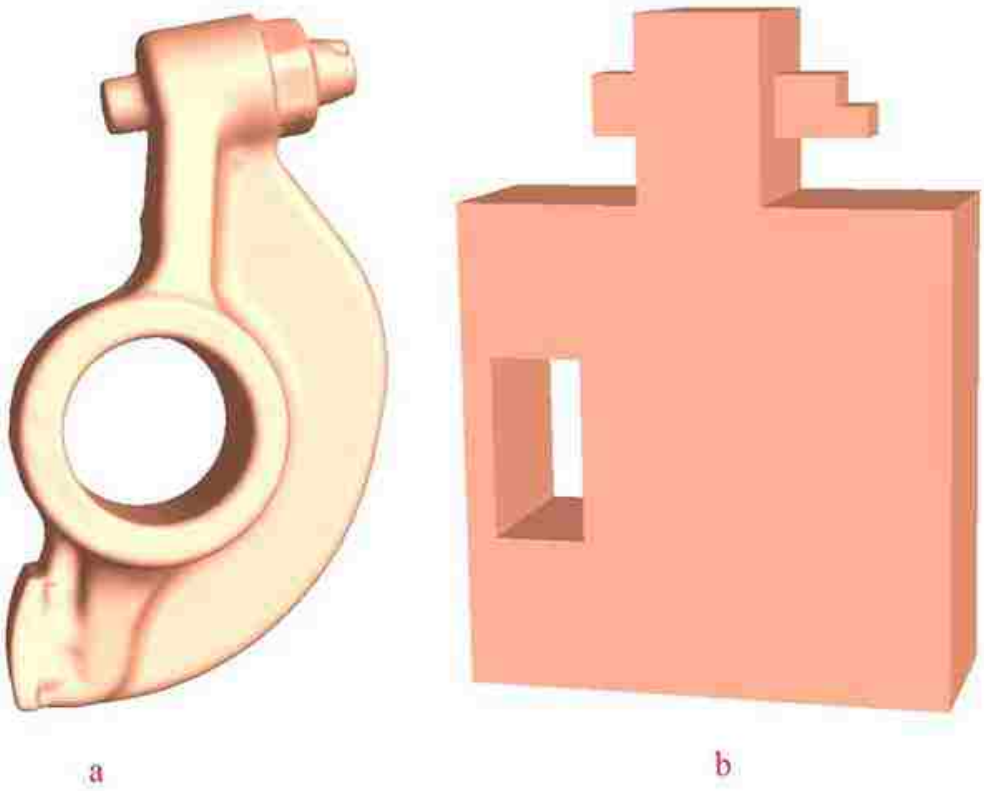


Figure 1.1. Example of A Polycube for Rocker Arm Model. (a) is model Rocker Arm of genus 1. (b) is a polycube for it with the same topology.

points of the parametrization. They are undesirable in many tasks such as spline construction, physics-based simulations and etc. On the other hand, if one uses fewer cubes to construct a simpler domain with fewer corner points, the parametrization will possess larger distortion due to the dissimilarity of geometric structures between the model and the domain shape. Therefore, when a fundamental question is asked: what is the optimal polycube domain? A reasonable answer can be an optimized balance between the singularity number and mapping distortion. More specifically, we try to solve the following problem: given a surface and a budget n of the singularity point number, what is the optimal shape of the polycube domain so that the parametrization has the least distortion and has no more than n corners?

In the second part of this dissertation, we present an effective optimization framework to compute polycube mapping. We develop an iterative algorithm to seek for the optimal polycube domain and mapping, with the constraint on using a restricted number of cubes

(therefore restricted number of corner points). We also use our polycube mapping framework to compute the optimal canonical polycube domain for multiple objects simultaneously for lowly-distorted consistent parametrization.

1.1.3 Volumetric Parameterization

Surface mapping techniques have been extensively studied in the past decade. However, parameterization over the solids is challenging. The rapid advancement of 3D scanning techniques makes it easier to acquire massive 3D data nowadays. Solid volumetric data have richer contents than those of the boundary surface. When the data processing or analysis are related to material, intensity, or any other structural information defined over the whole 3D region of the object (instead of on just its boundary shell), we need to consider the shape as a 3-manifold and study the volumetric mapping. Because of its importance, volumetric parameterization has gained greater interest in recent years, and a few related research work has been conducted towards various applications such as shape registration [13, 11, 14], volumetric deformation [15, 16, 17, 18], and trivariate spline construction [19], and so on.

In the third part of this dissertation, we develop volumetric polycube parameterization guided by a frame field and demonstrate its application on high-quality hexahedral mesh generation.

1.1.4 Frame Field Construction and Optimization

Nieser et al [20] introduce a volumetric parameterization framework guided by a given frame field. The smoothness of the frame field has a significant impact on the distortion of the parameterization. Recently how to find a smooth frame field has attracted much attention from the researchers in geometric modeling. A smooth frame field is desirable for many applications, such as parametrization and remeshing.

Ray et al [21] introduced global periodic parametrization, in which a surface frame field was utilized to guide the parametrization. The quality of the parametrization and the quality of

the resulting quadrilateral meshing results depends heavily on the smoothness of the guiding frame field. For example, a bad frame field will lead to degenerate parametrization and produce inferior quadrilateral meshing results.

In addition, the manual manipulation of the frame field is tedious in [21] and automation becomes necessary. Though surface frame field has been studied intensively and fruitful [22, 23, 24, 25, 26], the volumetric frame field has not been explored thoroughly.

In the fourth part of this dissertation, we propose an efficient quaternion-based frame field optimization algorithm. Compared to state-of-the-art algorithms, our method is compact and efficient, which is very suitable for handling large geometric models. We also provide a useful frame field smoothing tool that will benefit for broad applications, such as parametrization and remeshing.

1.2 Organization

The remainder of this dissertation is organized as follows. The related literature is reviewed in chapter 2. In chapter 3, we study the problem of optimizing spherical parametrization efficiently using hierarchical strategy. Furthermore, we also demonstrate the application of our method in spherical harmonics computation. In chapter 4, we focus on finding a good polycube mapping with the constraint to preserve the topology of the input surface. In addition, we also explore the application of polycube as the common domain for inter-surface mapping. In chapter 5, we develop volumetric polycube parameterization and demonstrate its application on high-quality mesh generation. In chapter 6, we propose a quaternion-based frame field smoothing algorithm and compare the smoothness of the optimized frame fields with existing methods. Finally, we conclude this dissertation with limitation and future work in chapter 7.

2 Related Work

2.1 Surface Parametrization Using Harmonic Functions

Theories and technologies in surface parametrization have been widely studied and they have been playing critical roles in many geometric processing tasks in graphics, CAGD, visualization, vision, medical imaging [13, 27], physical simulation, and etc. Many effective techniques have been developed to solve the parametrization under different distortion metrics with different boundary conditions. A thorough review is beyond the scope of this dissertation, and we refer the readers to three great surveys/tutorials of surface mapping and their applications in [28, 29, 30].

One widely used scalar function used for constructing lowly-distorted surface parametrization is the harmonic function. The discrete harmonic map was first proposed by Pinkall and Polthier [31] and introduced to the computer graphics field by Eck et al. [32]. By discretizing the energy defined in [31], Desbrun et al. [33] constructed free-boundary harmonic maps. Harmonic maps are preferable due to at least two important reasons: (1) it is meaningful from physics' point of view. A harmonic map minimizes the Dirichlet energy and leads to a minimal surface [31]; (2) it can be easily discretized and efficiently calculated from the computational aspect. A discrete harmonic map can be approximated either through Finite Element Method(FEM) analysis of the harmonic energy [32], or via mimicking the mean value property of harmonic functions [34]. The computation of discrete harmonic mapping can be written as the optimization of a quadratic energy and be efficiently solved as a sparse linear system.

2.2 Spherical Parameterization

Spherical mapping has been studied for many geometric modeling and processing applications since the past decade. Haker et al. [35] compute the conformal spherical parametrization through the stereographic projection. Gu and Yau [3] use the Gauss-Seidel iterative approach to approximate a harmonic map. Zayer et al. [1] solve the spherical parametrization in a curvilinear coordinate system. Distortion near the cutting curve is then reduced through a tangential Laplacian relaxation similar to [3]. An optimal cutting curve selection is proposed by Li et al. in [36]. Gotsman et al. [37] generalize the planar barycentric coordinates to spherical domain and prove its theoretical correctness. Saba et al [38] propose a more efficient numerical solution to these non-linear equations introduced in [37]. Sheffer et al. [39] extend their angle based flattening parametrization to spherical domain by adding spherical constraints. Asirvatham et al. [40] present a spherical parametrization algorithm that enforces feature constraints. Tian et al. [41] solved the model of [39] using progressive mesh and present a hybrid stretch metric minimization to obtain spherical parametrization with low area distortion. Li et al [42] propose a spherical parametrization algorithm using PHT-splines. Praun and Hoppe [2] compute the spherical parametrization by minimizing a stretch-based measure. Friedel et al [43] avoid the flip-over in the spherical parametrization by introducing a modification of planar parametrization quality measures.

Spherical harmonics were first introduced as a type of parametric representation for radial or stellar surfaces [44]. They are a natural and convenient choice of basis functions for representing any twice-differentiable spherical function [45].

Spherical harmonics have several favorable properties such as orthonormality, completeness, and coarse-to-fine hierarchy. These make spherical harmonics a favorable representation in many geometric shape analysis tasks, such as 3D surface filtering [46], shape representation [47], large-scale data modeling [48], data reconstruction [49], and shape retrieval [50, 51].

2.3 Polycube Mapping

As a useful parametric domain, polycube maps have been studied in many different shape modeling applications. Tarini et al [6] invented the concept of polycube map and applied it to the texture mapping and synthesis. Fan et al [8] extended it to generate cross parametrization and morphing by mapping surfaces to polycubes then composing the map by finding the correspondence between them. In these approaches, polycube maps are computed by extrinsic methods such as projections. Wang et al [9] introduced an intrinsic method for polycube maps and built splines representation on the polycube parametric domain. Compared with extrinsic methods, the intrinsic approach reduced the mapping distortion significantly. Later, Wang et al [10] developed user controllable polycube maps for manifold spline construction. Both approaches required much user involvement in polycube design. Lin et al [52] presented an automatic polycube mapping approach, but the bijectivity was not guaranteed. Recently, He et al [53] presented a divide-and-conquer approach for automatic polycube map construction. In that paper, the bijectivity was guaranteed and the mapping had shown low angle and area distortion. Han et al [54] applied volumetric polycube maps to construct hexahedral shell mesh. Wan et al [55] introduced a topology-preserving optimization framework for polycube mapping, and use the polycube mapping framework to compute an optimal common polycube domain for multiple objects simultaneously for lowly distorted consistent parameterization. Xia et al [56] introduced an editable polycube mapping, based on a divide-and-conquer strategy, which gives much more control over the quality of the induced subdivision surface and makes processing of large models with complex geometry and topology feasible. Gregson et al [57] developed a novel rotation- and position- driven deformation algorithm to construct polycubes. But to get rid of topologically erroneous wedges, a non-trivial post-processing is necessary. Yu et al [4] presented a computational framework for polycube construction and volumetric parameterization. The algorithm has three steps: pre-deformation, polycube construction and optimization, and mapping computation. This

algorithm could robustly generate a simple polycube domain shape, suitable for lowly distorted volumetric parameterization. This polycube parameterization can be used for high quality hexahedral mesh generation.

2.4 Volumetric Parameterization

Solid geometry and volumetric data have richer contents than that of surfaces. Compared to surface mapping, volumetric parameterization is more challenging and a lof of theory on the surface could not directly generalized to the solid. Wang et al [13] compute the discrete volumetric harmonic mapping over tetrahedral meshes for volumetric mappings on solid spheres. Li et al [11, 12, 58] develop meshless methods using the fundamental solution method in computing harmonic and biharmonic volumetric maps. Martin et al [19] parameterize volumetric models onto cylinders using the finite element method, and later generalize the algorithm to more complicated models with medial surfaces [59]. Nieser et al [20] introduce a CUBECOVER mapping algorithm for hexahedral meshing, and the mapping is guided by a user-designed frame field. Huang et al [60] present a boundary-aligned 3D frame field optimization algorithm that can automatically generate a smooth frame field from a given surface frame field. But the resultant frame field is not guaranteed to be valid (to induce valid mapping). Li et al [5] solve singularity-restricted frame fields to fix the singularity errors in the direct rotational-symmetry solving. However, the generation of valid cross frame-field (hence valid mapping) is not guaranteed. Wang et al [61] present a volumetric modeling framework to construct a novel spline scheme called restricted trivariate polycube splines, which develops a new trivariate hierarchical spline scheme for volumetric data representation. Unlike conventional spline formulations and techniques, their framework is built upon a novel parametric domain called Generalized PolyCube (GPC), comprising a set of regular cubes being glued together. Yu et al [4] present an algorithm for polycube construction

and volumetric parameterization. The algorithm has three steps: pre-deformation, polycube construction and optimization, and mapping computation.

2.5 Frame Field Construction and Optimization

Surface Frame Field Design and Applications. Ray et al [21] introduce periodic global parametrization. Given two orthogonal piecewise linear vector fields estimated and extrapolated from principal curvature direction, their method solves two parametric functions, aligned with the input vector fields by optimization, which defines a parametrization on the surface. They also propose an automatic procedure to detect and fix the singularities in the surface frame field. QuadCover is proposed by Kälberer et al [62], which converts a given frame field into a single vector field on a branched covering of the 2-manifold and generates an integrable vector field by a Hodge decomposition on the covering space. Their frame field smoothing process is optional and adapted from the method in [63], which yields a smooth vector field roughly aligned to the features of the surface mesh. Li et al [64] present a complete interpolation scheme of vector fields on triangulated surfaces, which enables arbitrary singularities to be represented at vertices. With their data structure, the singularity index of a vertex in a vector field can be determined combinatorially. N-symmetry direction field is formalized by Ray et al [24]. They demonstrate the Poincare-Hopf theorem in the case of N-symmetry direction fields on 2-manifolds. Moreover, they also derive an efficient algorithm to design a smooth frame field interpolating user defined singularities and directions. Ray et al [25] introduce an intermediate representation of the surface frame field allowing the intuitive design operations such as smoothing and setting directional constraints, and restate the objective function in a way avoiding the singularities yielded by small geometric details. Zhang et al [22] and Palacios et al [23] build systems to deign the directional field on surface, which provides control over the topology of rotational symmetry fields on surfaces such as removing or relocating the singularities.

Volumetric Frame Field Design and Applications. Based on QuadCover [62], Nieser et al [20] introduce a first approach for generating a hexahedral mesh of an input volume with boundary aligned cubes, which is guided by a frame field. First, a frame field is designed with manual input from the user, which guides the interior and boundary layout of the parametrization. Then the parametrization and the hexahedral mesh are computed in a way aligning with the given frame field. This paper lays theoretical foundation for 3D hexahedral parametrization and analyzes topological properties of the appropriate function space. Huang et al [60] represent 3D frames whose smoothness is measured using spherical harmonics representation, which is invariant to combinations of rotations around any axis by multiples of $\frac{\pi}{2}$. They construct a smooth 3D cross-frame field that is aligned with the surface normal at the boundary. However, due to the unsolved singularities, there exists degeneracy in the volumetric parametrization, which makes a hexahedron-dominant mesh but pure hexahedral mesh. The inadmissible singularities, which leads to degeneracy in the volumetric parametrization, are classified and treated in [65]. This paper provides a procedure for the treatment of the defects in the singularity graph. However, there is no theoretical guarantee that all the conflicting geometric and topological structure can be detected using this method. Inspired by the CubeCover method, Li et al [5] present an all-hex mesh generation framework based on Singularity-Restricted Field (SRF). In this approach, a boundary-aligned 3D frame field is computed for a given volume. Then the frame field is converted to be singualrity-restricted by a set of topological operations.

3 Efficient Hierarchical Optimization for Spherical Mapping

Spherical mapping is a key enabling technology in modeling and processing genus-0 close surfaces. Spherical parametrization seeks a bijective map $f : M \rightarrow S$ between a given closed genus-0 surface M and a unit spherical domain S. For a very wide category of solid models that do not have handles or voids, their boundary surfaces are closed and genus-0. The sphere is a natural parametric domain for them, because unlike planar parametrization which needs to slice the surface M open, a spherical map provides a seamless and continuous parametrization. Such a parametric representation could also be used to facilitate many geometric modeling and processing applications such as re-meshing, morphing animations, shape analysis, and so forth.

Among all bijective spherical maps, a map that introduces small metric distortion is desirable. Isometry (preserving both angle and area) is ideal but usually not possible for a generally given M. We therefore seek for a map that minimizes either angle distortion, or area distortion, or a balancing between both of them. Computing such a spherical parametrization, however, is often formulated as a non-linear optimization problem, and cannot be computed efficiently. For example, harmonic spherical map is conformal [3]. Such a map on triangle meshes can be computed by enforcing a vanishing Laplace-Beltrami operator on each vertex's tangent plane. The resultant parametrization is angle-preserving. However, its area distortion could be very large, especially in the long and thin protrusion regions (such as ears of the Stanford bunny). A parametrization that balances angle-distortion and

area-distortion is therefore often desirable. Zayer et al. [1] propose the Curvilinear Spherical Parametrization which better overcomes area-distortion and is very efficient. Another state-of-the-art spherical mapping algorithm is proposed by Praun and Hoppe [2]. They used the progressive mesh to iteratively optimize the $L2$ stretching energy [66] defined piecewise on the triangle mesh of M. Such a coarse-to-fine solving scheme can greatly overcome the local minima issue which exists in almost all spherical parametrization formulations that aim to minimize angle and area distortion together. This algorithm computes the spherical mapping with least angle and area distortion. Inspired by this work, we also adopt the progressive simplification and develop a hierarchical optimization scheme. But unlike [2], we utilize the distortion energy [43] which is shown converged to the continuous energy. Furthermore, we develop an effective hierarchical optimization scheme over the mesh (with different resolutions) from both local and global aspects, to improve the mapping efficiency and efficacy significantly [67].

The main contributions of this chapter include

- We present an effective hierarchical optimization framework for the spherical parametrization problem. Compared with other state-of-the-art spherical mapping computation algorithms, our method generates a bijective and lowly-distorted mapping, and converges efficiently. Therefore, our hierarchical algorithm can be applied on large geometric models with complex geometry (e.g. with long branches) robustly.
- We demonstrate the application of our mapping algorithm in computing spherical harmonics representations, and use it for a potential subsequent application in shape analysis of deformation sequences.

3.1 Hierarchical Spherical Parametrization

3.1.1 Mapping Distortion

The angle and area distortions of a triangle mesh can be evaluated piecewise on each triangle. The angle distortion per triangle can be measured [68] on the map of each triangle $f_T : T \to t$:

$$E_D(T) = \cot\alpha|a|^2 + \cot\beta|b|^2 + \cot\gamma|c|^2,$$

where T and t are the triangle of mesh M and its image on the parametric sphere S respectively; α, β, γ are the angles in T and a, b, c are the corresponding opposite edge lengths in t. The area distortion can be naturally measured by

$$E_A(T) = \frac{Area(t)}{Area(T)}.$$

The integrated (over the area of parameter triangle t) angle and area distortions of the entire spherical parametrization $f : M \to S$ are therefore:

$$\bar{E}_D(M) = \sum_{i=1}^{N_F} E_D(T_i)Area(t_i) \tag{3.1}$$

$$\bar{E}_A(M) = \sum_{i=1}^{N_F} E_A(T_i)Area(t_i) \tag{3.2}$$

where N_F is the number of faces in this mesh.

Following the modification proposed by [43], we use the following formulations on angle and area distortions, which provide upper bounds of the spherical integrals and avoid degeneracy during the optimization:

$$E_D(M) = \sum_{i=1}^{N_F} d_i^{-2} \cdot E_D(T_i)Area(t_i) \tag{3.3}$$

$$E_A(M) = \sum_{i=1}^{N_F} d_i^{-2} \cdot E_A(T_i)Area(t_i) \tag{3.4}$$

where d_i is the minimum distance from the origin to triangle t_i. The objective function is their weighted sum:

$$E = \lambda E_D(M) + \mu E_A(M) \tag{3.5}$$

where λ and μ are parameters balancing weights of angle and area distortion terms. Area distortion is a common problem for spherical parametrization leading to under-sampling, especially for those models with long and thin protrusions, which could cause undesirable artifacts in applications such as spherical harmonics computation. We found that a relatively large weight on area distortion usually provides stable and desirable parametrization; hence in our experiments, we set $\lambda = 0.1$ and $\mu = 1.0$ by default (also see Section 3.3.2 for how these distortions affect spherical harmonic reconstruction accuracy).

3.1.2 Algorithm Overview

The distortion energy introduced in Section 3.1.1 is nonlinear and nonconvex. For general models, directly optimizing the energy will get trapped in local minima inevitably. We therefore adopt the progressive mesh [69] to simplify the mesh into coarser resolutions and solve the optimization hierarchically while we gradually refine the mesh back to the original tessellation. The progressive scheme is similar to [2], but our optimization is developed differently and is more efficient and effective. Given a genus-0 mesh M with n vertices, we first progressively simplify it to a tetrahedron with 4 vertices, denoted as M^4. We then use M^k to denote the resolution of M with k vertices, and v_i^k to denote the vertex which will split during the inverse refinement process. v_i^k is a vertex on M^k and it splits into v_i^{k+1} and v_{k+1}^{k+1} (suppose the newly inserted vertex is always given the id $k+1$) in the new mesh M^{k+1}. Based on the above definition, the algorithm pipeline is as follows:

1. Simplify M^n to a tetrahedron M^4 using progressive mesh;

2. Map M^4 onto a unit sphere domain S^4,get $f^4 : M^4 \to S^4$;

3. Following the vertex split order that refines M^4 back to M^n, optimize the spherical mapping $f^k : M^k \to S^k$ hierarchically.

3.1.3 Global Hierarchical Optimization

We progressively refine M^k from $k = 4, \ldots, n$, and during each vertex split $v_i^k \to \{v_i^{k+1}, v_{k+1}^{k+1}\}$, we find a locally optimal spherical position as the image for each of these two vertices v_i^k, v_{k+1}^{k+1} while fixing images of all their one-ring neighboring vertices. After every η (a constant integer) vertex splits, we perform an optimization on all these newly placed vertices as well as their neighboring vertices. Ideally, after each split, we can perform a local optimization on images of v_i^k, v_{k+1}^{k+1}, and all their neighboring vertices until we get to a local optimum. However, this precise local optimization per every vertex split is relatively expensive and sometimes not necessary.

Therefore, we only conduct the optimization after a set of vertices are inserted. The parameter η controls how often such optimization should be performed. A larger η indicates fewer optimization iterations and thus better efficiency. In our experiments, η is simply set as 40 by default since a small value leads to longer optimization time. The whole optimization algorithm is formulated in Algorithm 1. $N_1(v)$ indicates the set of one-ring neighboring vertices of v. ϵ^η and ϵ^F control the convergence threshold, and $Iter_{max}^\eta$ and $Iter_{max}^F$ are the allowed maximal iterations.

3.1.4 Local Optimization on a Vertex

After the split of a vertex v_i^k, we need to embed the images of the two new vertices v_i^{k+1} and v_{k+1}^{k+1} on the sphere. Here we solve a simple local optimization to determine valid (non-flipped) spherical locations for them. Later, after each η vertex splits, we will perform such local optimizations on new vertices as well as their neighboring vertices together. When the mapping of a vertex is updated and the objective energy change is bigger than a threshold,

Algorithm 1: Global Hierarchical Optimization

In : Initial map $f^4 : M^4 \to S^4$ and progressive records for vertices to split $\{v^5, v^6, \ldots, v^n\}$

Out: Spherical map $f^n : M^n \to S^n$

$k \leftarrow 4$; Priority queue $Q = \emptyset$
repeat
  Perform vertex split $v_i^k \to \{v_i^{k+1}, v_{k+1}^{k+1}\}$;
  Add v_i^{k+1}, v_{k+1}^{k+1} into Q;
  Locally optimize once the images of v_i^{k+1} and v_{k+1}^{k+1} on the sphere, denoted as $p_i = f^{k+1}(v_i^{k+1})$ and $p_{k+1} = f^{k+1}(v_{k+1}^{k+1})$;
  if *(k mod η = 0)* **then**
    $j = 0$; Evaluate map distortion E_j.
    repeat
      $v_i = Pop(Q)$; optimize v_i;
      for $\forall v_k \in N_1(v_i)$, add v_k to Q if $v_k \notin Q$;
      $j = j + 1$.
    until $|E_i - E_{i-1}| < \epsilon^\eta$ *or* $j > Iter^\eta_{max}$;
    $Q \leftarrow \emptyset$
  end if
  $k \leftarrow k + 1$;
until *(k=n)*;
$j = 0$; Evaluate map distortion E_j.
Insert all vertices of M^n into Q
repeat
  $v_i = Pop(Q)$; optimize v_i;
  for $\forall v_k \in N_1(v_i)$, add v_k to Q if $v_k \notin Q$;
  $j = j + 1$.
until $|E_i - E_{i-1}| < \epsilon^F$ *or* $j > Iter^F_{max}$;

its one-ring vertices may need to be optimized again. We propagate this local refinement to larger regions using a priority queue (details will be given in Section 3.1.5).

We develop a local optimization to find the most suitable spherical embedding of each vertex through an efficient great-circle search, which is formulated in Algorithm 2. In this algorithm, we do not update a vertex's spherical embedding if the energy reduction is not significant. A line search mechanism is employed on the great circle of the spherical domain. The two cases in Algorithm 2 are illustrated in Figure 3.1. Note even if the initial position introduces flip-over, the energy minimization would guide the movement of vertex's spherical

Algorithm 2: Local Optimization of Spherical Position $p_i = f(v_i)$ through Great-circle Search.

In : Initial Position $p_i = f^k(v_i) \in S^k$

Out: Optimized Position $\tilde{p}_i = f^{k+1}(v_i) \in S^{k+1}$

$g = \nabla E(p_i)^T p_i$

if $\rho(v_i) \leq \epsilon$ **then**

  Exit.

end if

if $-g \geq 0$ *and* $\rho(v_i) > \epsilon$ **then**

  //Case 1 :

  $\bar{u} = \frac{-\nabla E(p_i)}{\|\nabla E(p_i)\|}$. Then we have a great circle centered at origin passing p_i and $\bar{u}$, denoted as c_{p_i}; denote the one-ring link on the mesh surrounding p_i as r_{p_i}. Let $\bar{u}_0$ be the intersection of c_{p_i} and r_{p_i}. $d_0 = \bar{u}_0 - p_i$.

  $k = 0$.

  while *true* **do**

    $k = k + 1$.

    if $E(\bar{u}_k) \leq E(p_i) + \delta d_k^T \nabla E(p_i)$ **then**

      update $\tilde{p}_i = \bar{u}_k$.

      Exit.

    else

      $\bar{u}_{k+1} = \frac{p_i + \bar{u}_k}{\|p_i + \bar{u}_k\|}$.

      $d_{k+1} = \bar{u}_{k+1} - p_i$.

    end if

  end while

end if

if $-g < 0$ *and* $\rho(p_i) > \epsilon$ **then**

  //Case 2 :

  $\bar{u}_0 = p_i - \frac{2g}{\|\nabla E(p_i)\|^2} \nabla E(p_i)$; (thus $\|\bar{u}_0\| = 1$)

  $d_0 = \bar{u}_0 - p_i$.

  $k = 0$.

  while *true* **do**

    $k = k + 1$.

    if $E(\bar{u}_k) \leq E(p_i) + \delta d_k^T \nabla E(p_i)$ **then**

      update $\tilde{p}_i = \bar{u}_k$.

      Exit.

    else

      $\bar{u}_{k+1} = \frac{p_i + \bar{u}_k}{\|p_i + \bar{u}_k\|}$.

      $d_{k+1} = \bar{u}_{k+1} - p_i$.

    end if

  end while

end if

image to a valid position free of flip-over. This local optimization is efficient and will converge within finite steps (see Section 3.1.7) for detailed analysis.

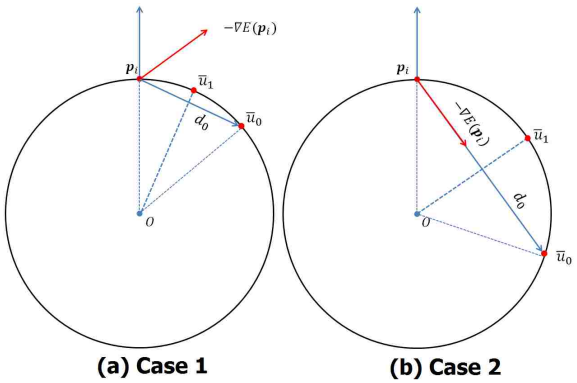


Figure 3.1. Two Cases In Great-circle Back Tracking Line Search. We have a great circle centered at origin O passing p_i and $\bar{u} = \frac{-\nabla E(p_i)}{\|\nabla E(p_i)\|}$, denoted as c_{p_i}; $\nabla E(p_i)$ is the gradient of E at p_i; denote the one-ring link on the mesh surrounding p_i as r_{p_i}. Let $\bar{u}_0$ be the intersection of c_{p_i} and r_{p_i}. $d_0 = \bar{u}_0 - p_i$. If $E(\bar{u}_0) \leq E(p_i) + \delta d_0^T \nabla E(p_i)$, we update p_i as $\tilde{p}_i = \bar{u}_0$ and exit. Otherwise, we choose a closer point on the great circle $\bar{u}_1 = \frac{p_i + \bar{u}_0}{\|p_i + \bar{u}_0\|}$ to continue the search for optimal position, which is analogous to the standard Armijo-type back tracking line search.

3.1.5 Priority Queue

When optimizing spherical images of the vertices, we iteratively pick a vertex to do its local optimization. The order of picking vertices is important and it could greatly affect the result and computation efficiency. Intuitively, we shall optimize the vertex whose movement potentially reduces the distortion energy most significantly. Both the magnitude of the first order KKT [70] violation and the distance the vertex can move are critical for the energy reduction. For example, in a region whose spherical mapping shrinks severely, KKT violations of the objective functions on vertices could be big, but spherical embedding of these vertices could not move much (since all these spherical triangles are already very small) before flipover appear. Then moving such vertices may not have high priority. We therefore use the first order KKT violation magnitude multiplying the potential moving distance as the key for this priority queue.

Therefore, for the priority queue in Algorithm 1, we adopt the following priority function τ defined on v_i's spherical image p_i:

$$\tau(v_i) = \rho(v_i) \cdot d \tag{3.6}$$

where d is the distance from p_i (v_i's image on sphere) to the boundary of its *spherical kernel* (see Section 3.1.6) along the negative gradient direction. And ρ is the magnitude of the first order KKT optimality violation:

$$\rho(v_i) = \|\nabla E(p_i)\| \sqrt{1 - (\frac{\nabla E(p_i)^T \cdot p_i}{\|\nabla E(p_i)\|})^2} \tag{3.7}$$

where $\nabla E(p_i)$ is the gradient of the objective function E of Eq 3.5 at vertex v_i. Note that the feasibility condition $\|p_i\| = 1$ is always guaranteed by the construction of the Algorithm 2. In our experiments, we simply use the average distance from p_i to its spherical one-ring to approximate d. $\tau(v_i)$ therefore estimates the aforementioned potential function reduction at vertex v_i, measured via the first order KKT optimality condition violation ρ at p_i multiplied by d.

3.1.6 Spherical Kernel and the Mapping Bijectivity

The spherical kernel can be defined on the spherical polygon formed by the one-ring neighboring vertices of a vertex v_i. It is defined and can be computed as the intersection of the open hemispheres defined by the spherical polygon edges. To avoid the flip-over on the spherical parameterization, we shall maintain a valid spherical embedding. This can be guaranteed if every vertex is inside its spherical kernel. We generalize the planar kernel computation algorithm [71] onto the spherical triangle mesh. The computation is efficient and takes $O(k)$, where k is the number of vertices on the spherical polygon.

The **bijectivity** of the spherical mapping can be shown. First, during local optimization, a non-flipped local region will not be converted into a flipped local region. Therefore, if we can guarantee the initial spherical embedding during the entire progressive refinement is

valid, then our final parameterization is non-flipped. Through induction, we can show that a valid initial spherical embedding can always be constructed during vertex split. (1) After the progressive simplification, the mesh is simplified to a tetrahedron M^4 with 4 vertices, which can be embedded on the sphere. (2) Suppose the mesh M^k with k vertices has a valid spherical embedding, and the next refinement is to do the vertex split from v_i^k to $(v_i^{k+1}, v_{k+1}^{k+1})$, then the spherical kernel for v_i^k is not empty. Then it can be shown that non-empty spherical kernel regions for v_i^{k+1}, v_{k+1}^{k+1} can always be constructed [2]. Therefore, a valid spherical embedding for the refined mesh M^{k+1} exists and can be used as the initial spherical positions for the next insertion and refinement. The mapping bijectivity is therefore guaranteed.

3.1.7 Analyzing Convergence of the Optimization

The first order KKT optimality condition of $\min E(p_i)$, subject to $\|p_i\| = 1$ can be written as

$$\nabla E(p_i) - \lambda p_i = 0, p_i^T p_i = 1. \tag{3.8}$$

where $\lambda \in R$ is Lagrange multiplier associated with the ball constraints. By considering $p^T p = 1$, which is guaranteed by the algorithm, we have $\lambda = \nabla E(p_i)^T p_i$. Then, the 2-norm residue of the left hand side of the first equation in Eq 3.8 can be written as

$$\rho(v_i) = \|\nabla E(p_i)\| \sqrt{1 - (\frac{\nabla E(p_i)^T \cdot p_i}{\|\nabla E(p_i)\|})^2} = 0 \tag{3.9}$$

which can be considered as the magnitude of the violation of KKT condition. When $\rho(v_i)$ is not small, p_i is not close to a local minimum. Then, because the angle between the asymptotic searching directions given in Algorithm 2 and the negative gradient direction of the energy function is an acute angle, the Armijo-type great-cirle back tracking line search described in Algorithm 2 will be successful within a finite number of steps and a sufficient energy value reduction relative to the KKT violation $\rho(v_i)$ will be obtained. This would force the first order KKT violation $\rho(v_i)$ goes to zero, since the energy function value is always bounded above from zero.

Globally, the objective function energy is also bounded below, actually nonnegative, and monotonically decreasing in each phase of the Algorithm 1. Furthermore, the great-circle back tracking line search conditions in Algorithm 2 will prevent the step size getting too small and the energy value will be reduced sufficiently when p_i is far away from local minimum. Therefore, globally, the total energy will decrease relatively rapidly to a minimum value.

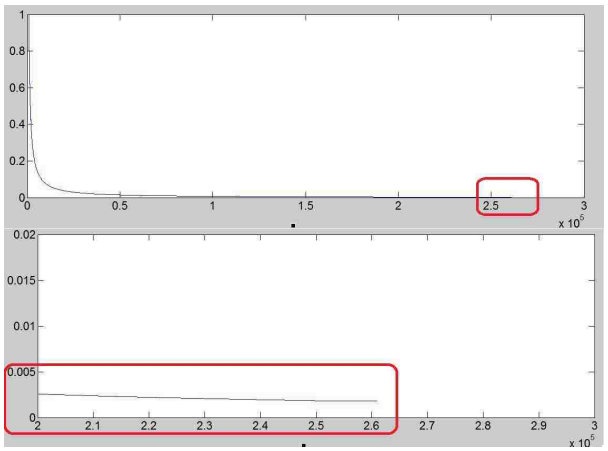
The graphs of the total distortion energy E per vertex in the optimization are depicted in Figure 3.2. In these figures we can observe that the energy drops severely in the beginning and the slope of the graph asymptotically goes towards zero with increasing number of iterations. This indicates that our approach finally converges.

3.2 Experimental Results and Discussions

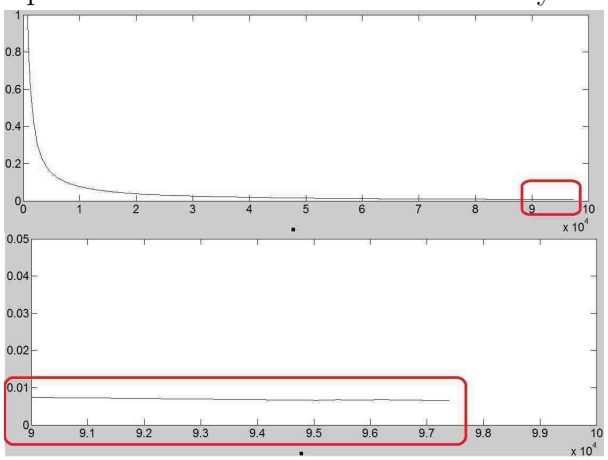
To perform side-by-side comparisons, we have implemented the harmonic spherical mapping [3], curvilinear spherical parameterization [1], and we obtained mapping results from the progressive spherical parameterization of [2]. We also parameterize various input models using our algorithm under different weights. In experiments demonstrated in this section, we use $\lambda = 0.1$ and $\mu = 1.0$.

Figure 3.3 shows the spherical images of the bunny using different parameterization methods. Like the spherical harmonic map, the curvilinear mapping [1] (a) could lead to a unnaturally stretched region around the tail region of Bunny; the progressive mapping computation result from [2] (b) relieves this artifact by dispersing the vertices in this region. Our mapping result, as shown in (c), provides a further less stretched parameterization in comparison with both (a) and (b).

Figure 3.4 and 3.5 demonstrate the effectiveness of our approach on Bunny(34K vertices) and Cow(11K vertices) by color-encoding angle and area distortions of the spherical mappings computed by [3], [1], and [2]. We can see that results of our method in the rightmost column are in cooler color, and therefore it has lower angle and area distortions.



(a) Spherical Parametrization of the Bunny Model



(b) Spherical Parametrization of the Cow Model

Figure 3.2. Energy per Vertex with respect to Iterations. The vertical axis shows energy per vertex and the horizontal axis shows the number of iterations. We do not plot iterations at the beginning that have large energies for more clear visualization.

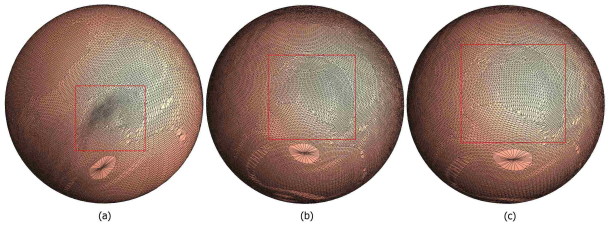
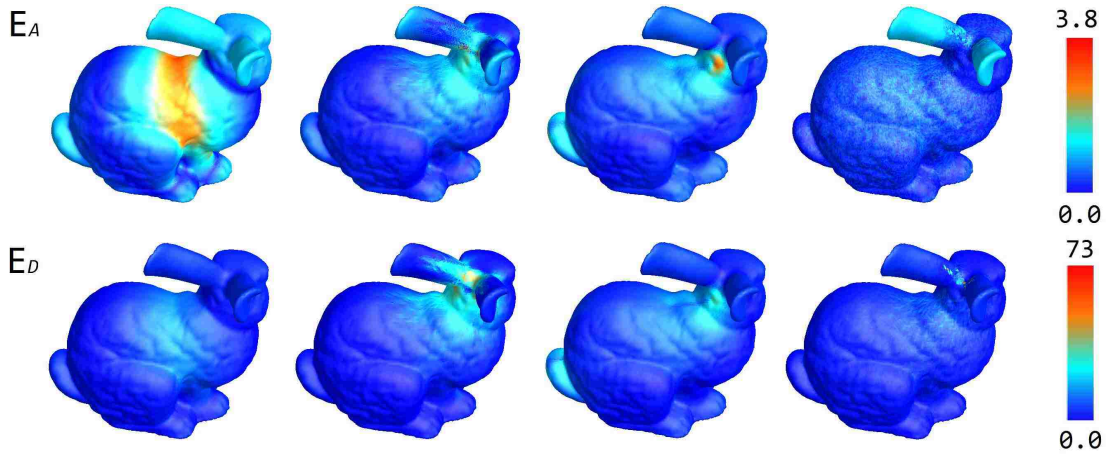


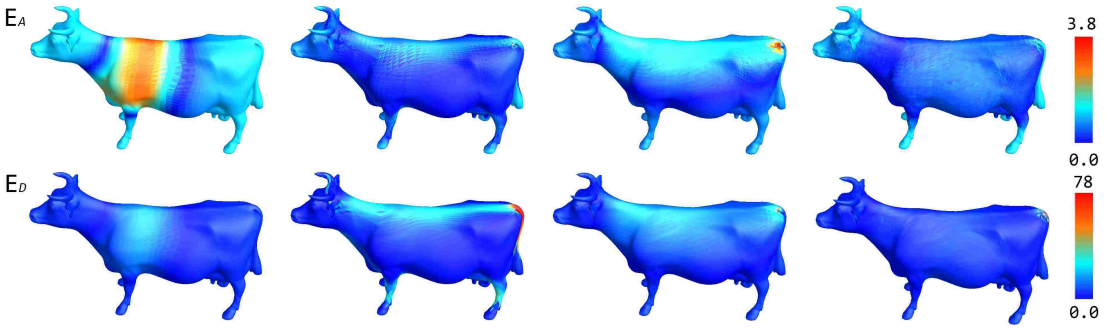
Figure 3.3. Comparison of Spherical Domain around the Tail Region on Bunny. (a)curvilinear spherical parameterization [1];(b) is from [2]; and (c) for Our approach. Our approach has slightly area distortion than [2] and much better than [1].

Figure 3.6 demonstrates the results of our approach on the Head (13K vertices) model side by side compared with [3] and [1]. Our approach introduces smaller angle and area



(a)[3] (b)[2] (c)[1] (d)Ours

Figure 3.4. Comparison of Other Spherical Parameterization Algorithms and Our Method on the Bunny model. (a) is from [3]; (b) is from (b)[2]; (c) is from [1] and (d) is from our method. E_A and E_D indicate area distortion and angle distortion. Warmer color, e.g red, indicates larger distortion; while cooler color, e.g. blue, indicates lower distortion. The rightmost column shows our results, which exhibits lower angle and area distortion.

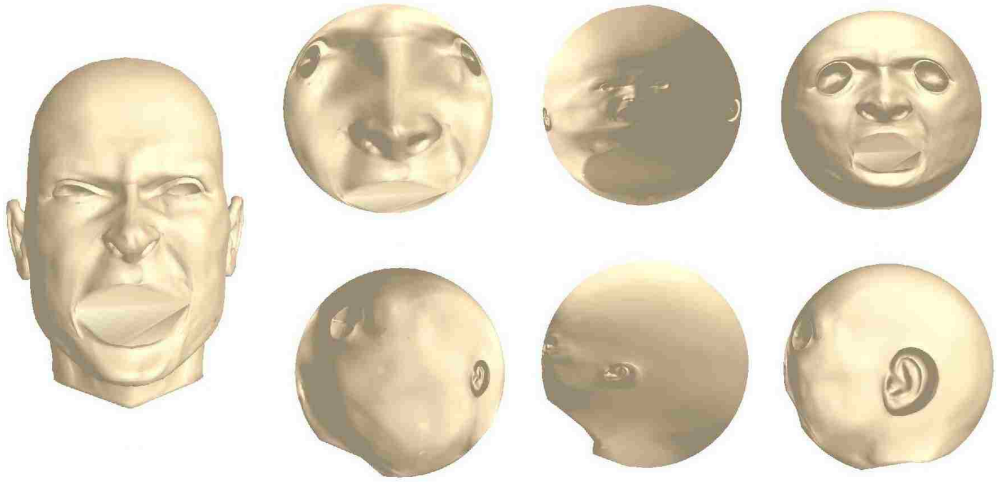


(a)[3] (b)[2] (c)[1] (d)Ours

Figure 3.5. Comparison of Other Spherical Parametrization Algorithms and Our Method on the Cow model. (a) is from [3]; (b) is from (b)[2]; (c) is from [1] and (d) is from our method. E_A and E_D indicate area distortion and angle distortion. Warmer color, e.g red, indicates larger distortion; while cooler color, e.g. blue, indicates lower distortion. The rightmost column shows our results, which exhibits lower angle and area distortion.

distortions, and hence better preserves the facial features like eyes, nose, mouth and ears on the sphere.

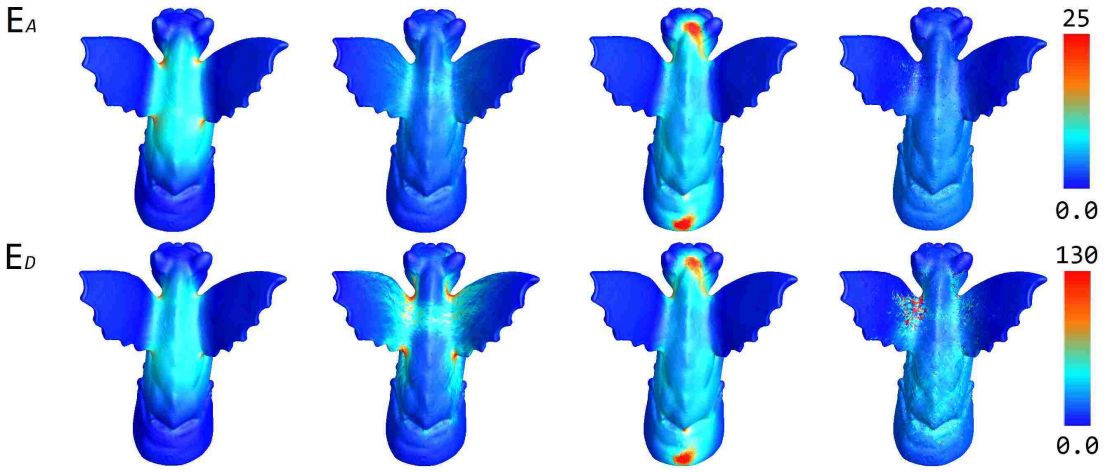
Numerically, the spherical mapping results of Bunny, Cow and Gargoyle, computed by [3], [2] and [1] are compared with our approach in Tables 3.1, 3.2 and 3.3. The visualization of



Input [3] [1] Ours

Figure 3.6. Comparison of Other Spherical Parameterization Algorithms and Our Method on the Head model. The leftmost column is the input model and the rightmost column is the result from our method. Our approach preserves the facial features like eyes, nose, mouth and ears more naturally.

E_D and E_A is placed in Figures 3.4, 3.5 and 3.7 respectively, where cooler color indicates less distorted triangle map and warmer color indicates more distorted triangle map.



(a)[3] (b)[2] (c)[1] (d)Ours

Figure 3.7. Comparison of Other Spherical Parametrization Algorithms and Our Method on the Gargoyle model. (a) is from [3]; (b) is from (b)[2]; (c) is from [1] and (d) is from our method. E_A and E_D indicate area distortion and angle distortion. Warmer color, e.g red, indicates larger distortion; while cooler color, e.g. blue, indicates lower distortion. The rightmost column shows our results, which exhibits lower angle distortion and comparable area distortion. And our approach is much more efficient on large-scale models like this one.

Table 3.1. Comparison of Distortions on Bunny between Our Method and Existing Algorithms.

#Vertices = 34K				
	[3]	[1]	[2]	Ours
#Flipover	2585	0	3	0
$E_D(M)$	50.79	63.57	78.07	61.37
$E_A(M)$	22.80	25.46	14.01	14.24
Time(s)	2397	91	600	58

Table 3.2. Comparison of Distortions on Cow between Our Method and Existing Algorithms.

#Vertices = 11K				
	[3]	[1]	[2]	Ours
#Flipover	2536	2	0	0
$E_D(M)$	51.19	73.15	117.32	69.92
$E_A(M)$	32.85	23.79	14.35	15.49
Time(s)	224	28	420	21

Table 3.3. Comparison of Distortions on Gargoyle between Our Method and Existing Algorithms.

#Vertices = 100K				
	[3]	[1]	[2]	Ours
#Flipover	6106	9	0	0
$E_D(M)$	51.66	78.80	81.17	81.79
$E_A(M)$	93.61	141.66	41.48	47.70
Time(s)	24393	1151.4	1380	193

Figure 3.8 illustrates some more mapping results computed by our algorithm. The execution times of our method on 9 models (whose vertex sizes vary from 11K to 400K) are reported in Table 3.4. Our implementation is unoptimized and the experiments are conducted on a desktop compute with AMD Athlon X2 2.9GHz CPU and 2GB RAM.

Our experiments and comparisons indicate:

- Our parametrization produces mappings with lower angle and area distortions;
- Our optimization is more efficient, especially for large-scale models;
- Our approach generates bijective spherical mappings.

Figure 3.8. More Results from Our Approach. There are three subfigures for each model: an original model subfigure, two spherical domain subfigures from different perspective.

Table 3.4. Execution Time of Our Approach on Various Models.

Models	Cow	Frog	Bunny	Horse	David	Venus	Gargoyle	Armadillo	Buddha
#Vert	11K	25K	34K	48K	50K	50K	100K	106K	400K
Time(s)	21	48	58	89	111	70	193	250	526

3.3 Application on Spherical Harmonics Representation

A function $f(\theta, \phi)$ defined on a sphere can be decomposed as the sum of its harmonics as follows.

$$f(\theta,\phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{m=l} a_{lm} Y_{lm}(\theta,\phi)$$

where Y_{lm} is spherical harmonics of degree l and order m, defined as follows.

$$Y_{lm}(\theta,\phi) = \sqrt{\frac{2l+1}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}} P_{lm}(cos\theta) e^{im\phi}$$

where P_{lm} is known as associated Legendre polynomial. the coefficients of spherical harmonics decomposition a_{lm} can be calculated as:

$$a_{lm} = \int_0^{2\pi} \int_0^{\pi} Y_{lm}^*(\theta,\phi) f(\theta,\phi) sin(\theta) d\theta d\phi$$

3.3.1 Decomposition and Reconstruction

To convert a given closed genus-0 triangle mesh M into a spherical harmonics representation, we adopt the following intrinsic computation:

1. Compute the spherical parametrization $f : S \to M$;

2. The x, y, and z component of M can be considered as three spherical height functions, $x(\theta, \phi)$, $y(\theta, \phi)$ and $z(\theta, \phi)$, respectively;

3. Compute the spherical harmonic coefficients of these three height functions.

After obtaining the spherical harmonics coefficients of a spherical function $f(\theta, \phi)$, we can reconstruct and approximate this function from its coefficients. Given a user-specified maximum degree L_{max}, we can calculate the approximating function on S:

$$\hat{f}(\theta, \phi) = \sum_{l=0}^{L_{max}} \sum_{m=-l}^{m=l} a_{lm} Y_{lm}(\theta, \phi)$$

Then surface M is reconstructed through reconstructing all its three spherical height functions $x(\theta, \phi)$, $y(\theta, \phi)$ and $z(\theta, \phi)$. Figure 3.9 illustrates the reconstruction results.

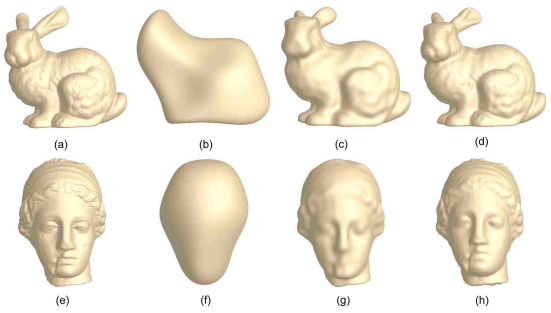


Figure 3.9. Reconstruction Results by Spherical Harmonics. Leftmost column contains original models. Middle left column contains reconstruction only using low frequencies, including 6×6 coefficients. Middle right column contains reconstruction using 16×16 coefficients. Rightmost column contains reconstructed results from low and higher frequencies, including 32×32 coefficients. We can approximate the input model better as more coefficients from higher frequencies are utilized.

Table 3.5. Spherical Harmonic Reconstruction Accuracy Using Different Parameterization Methods. L_{max} is the number of frequency utilized; the number in the right three columns indicates the reconstruction error $e(M, \hat{M})$.

	L_{max}	harmonic [3]	curvilinear [1]	ours
Bunny	6	0.1190	0.1028	0.0891
	16	0.0709	0.0730	0.0424
	32	0.0643	0.0669	0.0247
Venus	6	0.0299	0.0455	0.0279
	16	0.0117	0.0160	0.0113
	32	0.0088	0.0102	0.0083
Cow	6	0.3480	0.2424	0.1474
	16	0.2317	0.1547	0.0813
	32	0.1470	0.1161	0.0591
Horse	6	0.3236	0.2698	0.1653
	16	0.2440	0.1792	0.0740
	32	0.1976	0.1401	0.0563
Armadillo	6	0.3151	0.1962	0.1260
	16	0.2127	0.1106	0.0309
	32	0.1725	0.0900	0.0222

3.3.2 Analysis of Spherical Harmonic Reconstruction Accuracy

We can evaluate the reconstruction accuracy by calculating an error term $e(A, B) = max\{dis_{AB}, dis_{BA}\}$ according to the Hausdorff distance [72]: dis_{AB} is defined as the average value of $D(A_i)$, where $D(A_i)$ is the minimum distance from point $A_i \in A$ to another set B. Therefore, we use $e(M, \hat{M})$ to measure the distance from the original mesh M to its spherical harmonic reconstruction $\hat{M}$.

We conduct the reconstruction experiments on a set of models and show their reconstruction accuracy in Table 3.5. As number of utilized frequencies L_{max} increases, the reconstruction error $e(A, B)$ decreases. We can therefore obtain a progressive surface reconstruction. For models with long and thin protrusions, e.g. cow and horse, curvilinear map [1] is better than harmonic map [3]; when models are smooth, such as Venus, harmonic map exhibits lower reconstruction error. Meanwhile, our results are better than both [1] and [3], which

demonstrates that our proposed parameterization is ideal for spherical harmonic computation and analysis.

Furthermore, we also check how angle versus area distortions, i.e. the ratio $\frac{\lambda}{\mu}$ in Eq 3.5, influences the reconstruction accuracy is also explored. Figure 3.10 and 3.11 illustrate this influence on Venus and Bunny. In both of the two figures, (a) 6×6 coefficients are used; (b) 16×16 coefficients are used.

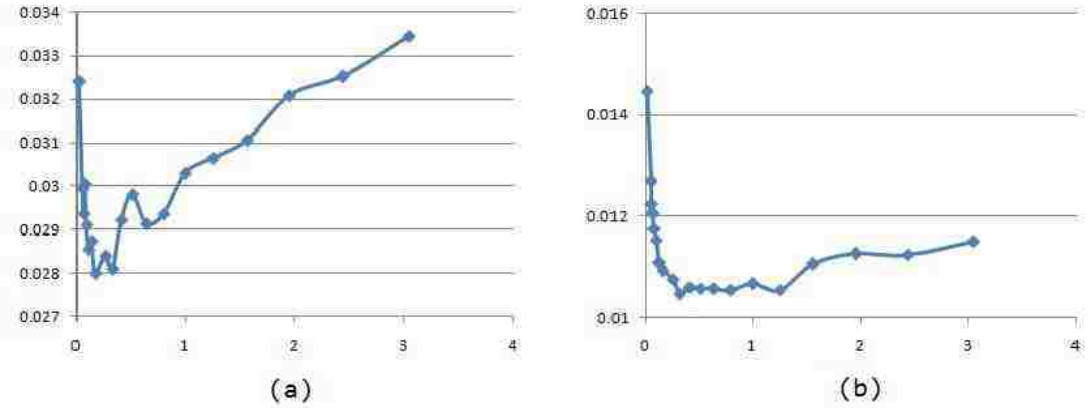


Figure 3.10. Effect on Reconstruction Accuracy from $\frac{\lambda}{\mu}$ in Eq 3.5 on Venus. (a) 6×6 cofficients are used; (b) 16×16 coefficients are used. The optimal ratio is about $(0.2, 0.3)$ for this model.

Figure 3.10 indicates the optimal ratio for spherical harmonics is roughly within $(0.2, 0.3)$ for Venus. This indicates that the angle and area distortion should be balanced together. Either the dominance of angle distortion or area distortion will yield large reconstruction error. In Figure 3.11, it is a bit different. The reconstruction has smaller error when λ is very small. This indicates that the attention to area distortion should be paid much more than to angle distortion. This situation might be due to the long protrusion region near the ears. It is prone to have huge area distortion near the ears, since it is usually mapped to a very tiny region on the sphere and the parametric triangle area is close to zero. These will severely increase the spherical harmonics reconstruction error. Based on these experiments, we observe that for rounded objects we shall choose a balanced ratio $\frac{\lambda}{\mu}$ while for objects

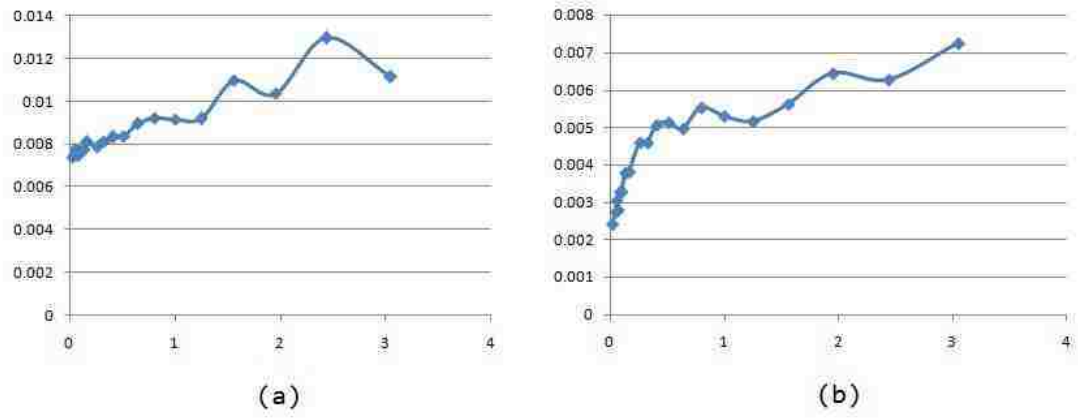


Figure 3.11. Effect on Reconstruction Accuracy from $\frac{\lambda}{\mu}$ in Eq 3.5 on Bunny. The optimal ratio is when $\lambda = 0$, which means area distortion should be put much more attention to. This might due to the long protrusion region near the ears, which easily causes large area distortion and leads to undersampling in this region.

with long protrusions, area-preserving is more important for effective spherical harmonic decomposition/reconstruction.

3.3.3 Deformation Analysis Using Spherical Harmonics

With lowly distorted spherical mapping, closed genus-zero surfaces can be parameterized onto a canonical sphere domain and converted to spherical harmonics effectively. Then we can analyze the shape using its spherical harmonics representations. We develop a simple experiment on a set of head models with different expressions (therefore, different geometry) [73] to demonstrate this.

For a spherical function f, upon a given frequency l:

$$f_l(\theta, \phi) = \sum_{m=-l}^{m=l} a_{lm} Y_{lm}(\theta, \phi),$$

we can define the energy of this frequency using the following L_2-norm:

$$\|f_l(\theta, \phi)\| = \sqrt{\int_0^{2\pi} \int_0^{\pi} f_l(\theta, \phi) f_l^*(\theta, \phi) sin\phi d\phi d\theta}.$$

An important property of spherical harmonics is that the energy in each frequency of given signal is rotation invariant [50, 74]. Hence, this spherical harmonics representation is a

rotation-invariant descriptor for shape analysis. In our experiment, we analyze a deformation sequence, and measure the difference between different frames (i.e. their geometric shapes) by matching their spherical harmonic energies of various frequencies. The difference between two spherical functions f and g can be calculated as

$$D_n(f,g) = \sum_{l=0}^{n} (\|f_l(\theta,\phi)\| - \|g_l(\theta,\phi)\|)^2.$$

Figure 3.12 shows a set of deformed head models with different expressions. In terms of shape differences, we can consider facial expressions introduces the geometric deviations on the face especially around the eyes and mouth region. These deformed heads are all genus-0 models. So we can simply fill holes on them to make them a topological sphere. Then we compute their spherical harmonics and measure differences between different expressions using the above function $D_n()$. From the chart shown in (j), we can analyze their similarity. For example, from the first row of (j), we see a is very similar to b and c, and is different from g and h.

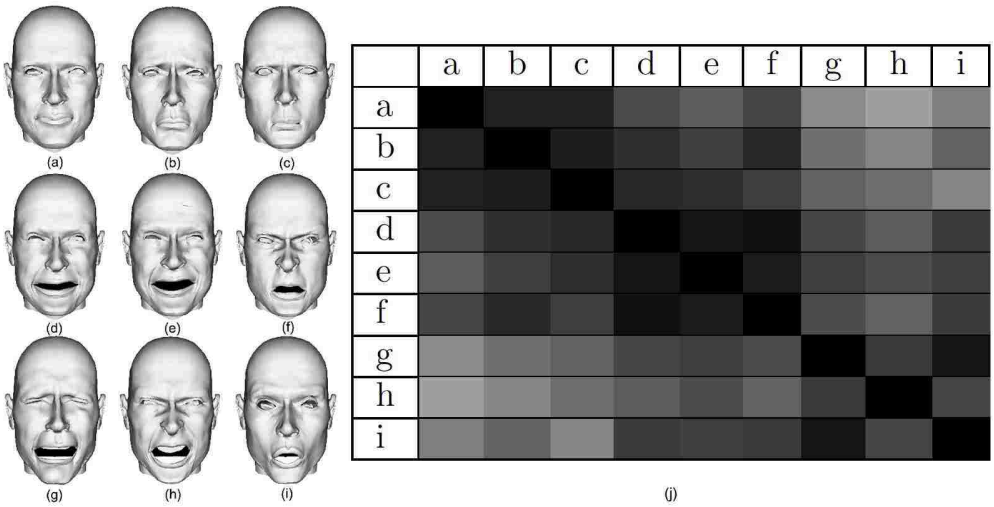


Figure 3.12. Comparison of Head Models with Different Expressions (a-i). The matching results are illustrated in (j), where black indicates better similarity and white indicates bigger difference. For example, following the first row of (j), (a) is very similar to (b) and (c), and is different from (g) and (h). For example, muscle geometry differ significantly between (a) and (h) in the mouth and eyes regions, while their variation is less between (a) and (b,c).

Spherical harmonics computation and spherical harmonic analysis just show a simple example of applications of our spherical parametrization. Our lowly distorted spherical mapping provides an effective intrinsic parametric representation of genus-0 surfaces. While spherical harmonics only encode the global property of 3D models, the spherical parametrization provides an enabling tool to analyze the shape's local differential properties. In the future, we will explore deformation analysis by combining both global property and local property.

3.4 Summary

In this chapter, an effective spherical parametrization algorithm that minimizes angle and area distortions is presented. An effective hierarchical spherical optimization scheme to solve it is developed. Compared with other state-of-the-art spherical mapping computation algorithms, this method generates a bijective and lowly-distorted mapping, and converges efficiently. Finally the effectiveness of this spherical parametrization in spherical harmonics computation and shape analysis is also demonstrated.

4 Topology-preserving Optimization for Polycube Mapping

Computing the parametrization of 3D shapes (surfaces/solids) on specific canonical domains is an important problem in shape modeling, and it can facilitate many computer graphics and geometric processing tasks. Although the sphere is a natural domain for genus-0 surfaces, there are a lot of models whose genus is higher than 0. In that case, the sphere might not be a suitable domain for them. Instead, polycube can serve as the parametric domain for surfaces of arbitrary genus.

Composed of a set of small cubes, a polycube well approximates the geometry of the free-form model yet possesses great regularity; therefore, it can serve as a nice parametric domain for free-form shape modeling and analysis. Polycube mapping was first introduced by [6]. It parameterizes a closed surface onto a polycube domain, which is composed of a set of small cubes. A polycube has the same topology of the given surface, and it is usually constructed to approximate the geometry of the surface. Therefore, the surface parametrization on a polycube domain often has much smaller distortion than that on a planar domain. Meanwhile, the polycube domain still possesses great regularity; each sub-patch is a rectangle; transitions between adjacent patches are simple rotation and translation except on corner points. Due to many of these advantages, the polycube mapping has been used in many graphics and shape modeling applications such as texture mapping [6] and synthesis [7], shape morphing [8], spline construction [9, 10], and volumetric matching [11].

Intuitively, the more cubes one uses to construct the polycube, the better the domain can approximate the original model, which brings parametrization very small area and angle distortion. However, corner points are singularity points of the parametrization. They are undesirable in many tasks such as spline construction [9, 10], physics-based simulations [75], etc. On the other hand, if one uses fewer cubes to construct a simpler domain with fewer corner points, the parametrization will possess larger distortion due to the dissimilarity of geometric structures between the model and the domain shape. Therefore, when a fundamental question is asked : what is the optimal polycube domain? A reasonable answer can be an optimized balancing between the singularity number and mapping distortion. More specifically, we try to solve the following problem: given a surface S and a budget n of the singularity point number, what is the optimal shape of the polycube domain P so that the parametrization $f : S \to P$ has the least distortion and P has no more than n corners?

Depending on applications, different metrics (angle distortion, area distortion, isometry distortion, etc) have been studied and used to measure the mapping quality. *Harmonic functions* are most widely used in constructing lowly distorted mapping. With a fixed boundary condition, a function $\phi(x, y)$ is harmonic if it is a solution of the Laplace's equation. When a boundary condition is given, ϕ is a minimizer of the Dirichlet energy [31, 32] and it possesses great smoothness. For example, conformal parametrization can be constructed by two conjugate harmonic functions [76, 77]. In this chapter, we use harmonic functions to construct polycube mapping, minimizing a metric energy composed by shape-preserving and area-preserving terms. The framework is general and can be used for other metrics. A similar idea, proposed by Pietroni et al [78], considered the trade off between the mapping distortion and the simplicity of the domain, solves the surface parametrization over abstract domains by locally optimizing the mapping on subregions then globally smoothing it [55].

Now the optimal polycube maps can be formulated as solving $argminE(P, f)$ for a given shape S, where energy function E is defined on any mapping $f : S \to P$ and P is a

polycube with n corners. Since the domain P is part of the optimization, it is extremely difficult. We restrict our optimization to a subspace of this problem, which we call a *topology-preserving polycube mapping*. Specifically, given an initial polycube domain $P = \{P_i\}$, the *topology* of the polycube P is defined by its dual graph (see Figure 4.1) $DM = \{DV, DE\}$. $DV = \{dv_1, \ldots, dv_n\}$ are nodes corresponding to rectangle subpatches $\{P_i\}$. DE is a set of edges: an edge $[dv_i, dv_j]$ is in DE, if P_i and P_j are adjacent to each other. We say two polycubes $P = \{P_1, \ldots, P_n\}$ and $Q = \{Q_1, \ldots, Q_m\}$ are *topologically equivalent*, if their dual graphs DP and DQ are isomorphic. Therefore, given an initial polycube P, our goal is to find the optimal polycube P' and the mapping f that minimizes distortion $E(P, f)$, in the same topological equivalence class (without changing the structure of its dual graph).

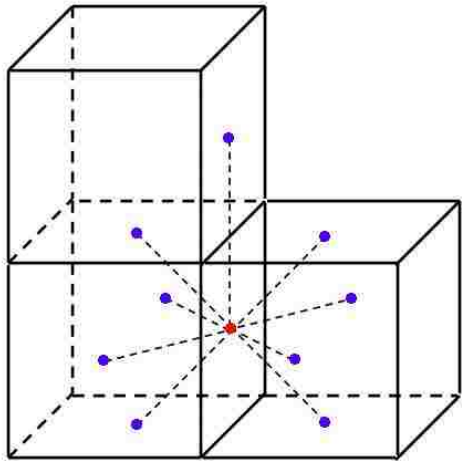


Figure 4.1. Part of the dual graph corresponding to one facet (red node) and its neighboring facets(blue node).

This chapter has three main contributions.

- We formulate the above optimal polycube mapping problem, and present a polycube mapping computation framework based on the given restricted complexity of polycube domain;

- We develop efficient optimization solvers to seek the topology-preserving optimal polycube domain and mapping iteratively.

- We extend the polycube optimization algorithm to multiple objects, for the construction of the common optimal domain for multiple models.

4.1 Algorithms Overview

A polycube domain P is composed of a set of rectangular patches P_i. A polycube map is therefore composed of a set of rectangle maps. We use the harmonicity and area distortion to measure the mapping quality and optimize the domain shape as well as the mapping.

Ideally, given a metric, we shall simultaneously optimize the polycube domain P as well as the mapping $f : S \to P$ to minimize the distortion $E(f)$. We can formulate this as minimizing $E(\mathbf{x}, \mathbf{y}) = E(x_1, x_2, \ldots, x_{3n}, y_1, y_2, \ldots, y_{3n})$, with the constraints that $(x_{3i-2}, x_{3i-1}, x_{3i})$ is a point on S, and $(y_{3i-2}, y_{3i-1}, y_{3i})$ is the corresponding corner point on the polycube P, for $i = 1, \ldots, n$.

Directly solving this nonlinear optimization is highly expensive. As will be discussed shortly (Section 4.3 and 4.4), the derivatives of E over $\mathbf{y}$ can be computed efficiently, but the derivatives of E over $\mathbf{x}$ could not be computed in practice. Without derivatives of the object function, this optimization with complicated constraints is difficult even for moderately large n. To make full use of the partial derivative information of the objective function, we iteratively do the optimization over $\mathbf{x}$ (for optimal polycube corner mapping) and $\mathbf{y}$ (for optimal polycube domain shape) separately. Hence, gradient based nonlinear optimization methods using the derivatives of $\partial E/\partial \mathbf{y}$ can be developed to efficiently optimize the subproblem $E(\mathbf{x}, \mathbf{y})$ for fixed $\mathbf{x}$. Meanwhile, a derivative-free optimization algorithm is developed to optimize the subproblem $E(\mathbf{x}, \mathbf{y})$ for fixed $\mathbf{y}$. During each iteration, when the shape of every rectangle and the mappings of its four corner points are determined, we can compute/update the mapping efficiently (see Section 4.2 and Section 4.4). The proposed

iterative polycube mapping optimization framework therefore has the following three steps (illustrated in Figure 4.2).

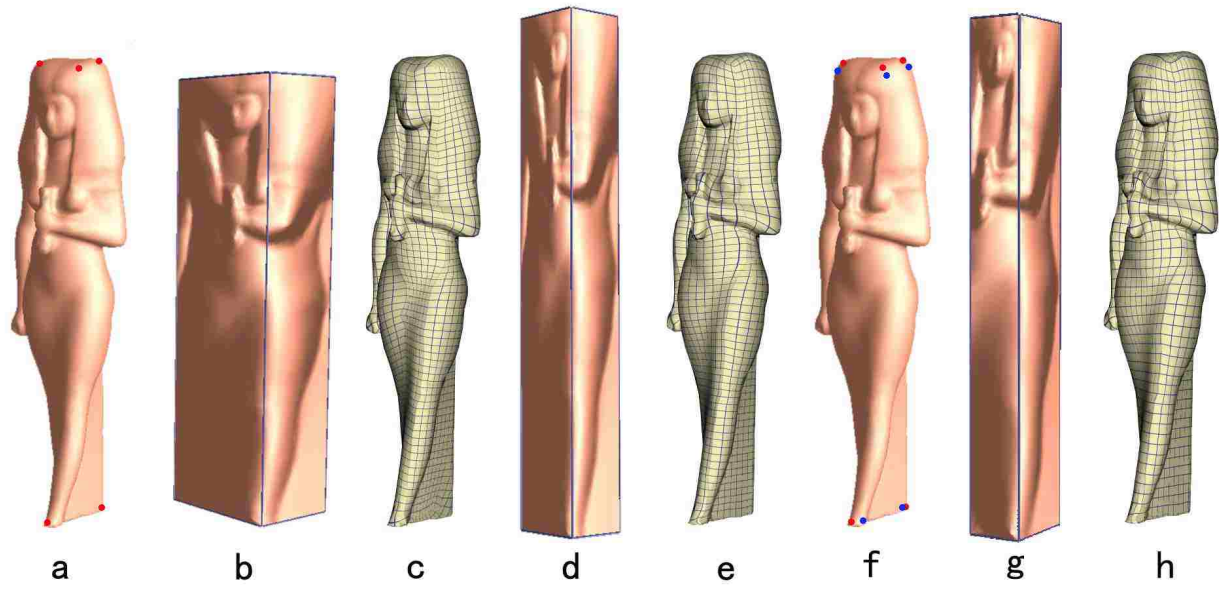


Figure 4.2. Algorithm Overview. (a) original surface with eight corner points(red). (b)(c) initial polycube domain and mapping. (d)(e) optimized polycube domain and mapping The harmonic energy with area distortion term is reduced from 5.4414 to 4.7812. (f) the optimized polycube mapping with eight new corner points(blue) with a lower harmonic energy of 4.5961. (g)(h) final optimized domain and optimized mapping after two iterations. The grid quality is improved.

1. *Initial Polycube Domain Construction* (Section 4.2). Given a budget number of corner points, an initial polycube domain is constructed either automatically or manually, meeting the corner point budgets; then the corner point mapping and the initial polycube mapping are computed.

2. *Optimizing Polycube Domain Shapes* (Section 4.3). Preserving the topology of the polycube, the scaling of sub-patches is optimized so that mapping energy is minimized.

3. *Optimizing Polycube Mapping* (Section 4.4). Without changing shape of the polycube, the surface-polycube mapping is optimized by searching the optimal corner point mapping.

Algorithm 3: Optimal Polycube Mapping.

input : surface S, corner point number n;
output: polycube mapping $f : S \to P$;

Construct an initial polycube P_0, whose corner point number $\leq n$;
Compute an initial mapping $f_i : S \to P_i; i = 0$;
repeat
  $i \leftarrow i + 1$;
  Optimize the polycube domain P_i, s.t. distortion of mapping f_{i-1} is minimized;
  Optimize the polycube map $f_i : S \to P_i$;
until $|P_i - P_{i-1}| < \epsilon$;
Perform a global smoothing.

The framework is formulated in Algorithm 3. Note that in our iterative process, we keep on optimizing scaling factors of sub patches and the corner points. Then (1)*polycube domain optimization* takes corner points decided by the current mapping f_i as the input and solve scaling of subpatches to reduce mapping distortion; and (2) *polycube mapping optimization* uses the scaled polycube P_{i+1} as the target domain and optimizes the location of corner points. This iterative refinement converges when the polycube domain shape P_i does not change any longer.

4.2 Constructing Initial Polycube and Mapping

The initial polycube can be constructed manually [6, 9], or automatically [52, 53]. We also use a simple voxelization algorithm (Section 4.2.1) to generate the polycube. Since this initial polycube and maps (Section 4.2.2) will be optimized to minimize the distortion, a simple, efficient, and adaptive (to different corner budgets) scheme such as this voxelization algorithm is sometimes enough. The following optimization framework is general, and can optimize an initial polycube mapping constructed via different methods.

4.2.1 Polycube Construction via Voxelization

Given a solid object M, supposing its boundary surface is represented by a triangle mesh $S = (\mathcal{V}_s, \mathcal{E}_s, \mathcal{F}_s)$ where $\mathcal{V}_s, \mathcal{E}_s, \mathcal{F}_s$ are vertex, edge, and face sets, we construct a polycube domain $P = (\mathcal{V}_p, \mathcal{E}_p, \mathcal{F}_P)$ and corresponding corner points mapping using a voxelization

algorithm. Figure 4.3 illustrates a polycube construction example of a Buddha model through voxelization.

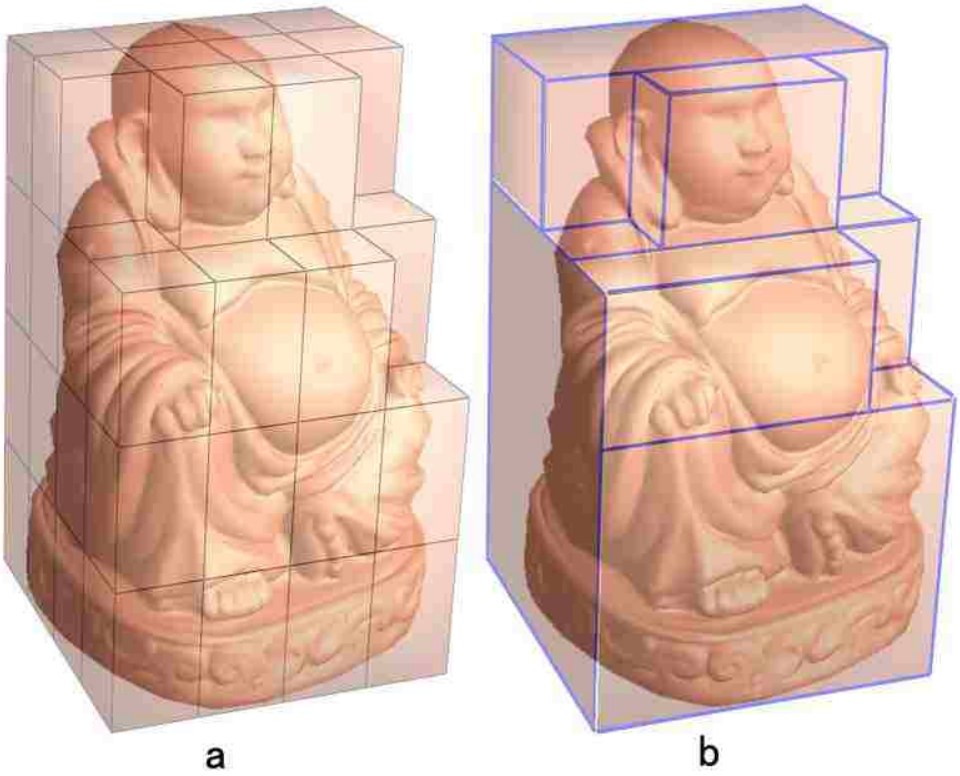


Figure 4.3. Voxelization For Polycube Construction.

We use an octree to represent the object. The subdivision starts from a rectangular bounding box. Each cell (rectangular cuboid) can be labeled as *inside* or *outside*. We remove all interior faces that are shared by two inside cells, and finally merge all inside cells to one polycube P. The remaining faces form the boundary surface of P. We further merge these remaining faces to a set of big rectangle facets of the polycube. Iteratively, we merge two adjacent faces if the result remains a planar convex polygon. After merging, only rectangle facets are left. The vertices of these rectangles are called corner points, denoted as $\mathcal{V}_{CP}$. And the edges of the rectangles form the connectivity of the corner points $\mathcal{E}_{CP}$. For each corner $v \in \mathcal{V}_{CP}$, we use the simple projection method [6] to find its corresponding points on S. Without ambiguity, we also call these corresponding points *corner points* on S, denoted as $\mathcal{V}_{CS}$; they will be mapped to corners in the initial polycube mapping. The voxelization

algorithm is simple, automatic and efficient. Moreover, the octree's depth can be adaptively decided by the number of corner points.

Voxelization approaches sometimes provide unnecessary zigzagged domain shapes when the geometry of the object is not well aligned with principal axes, which can be undesirable. Then other polycube domain construction algorithms (e.g. [6, 9, 52, 53]) may be used to construct the initial mapping, and our subsequent optimization paradigm can still be applied to refine the domain shape and improve mapping quality.

4.2.2 Initial Polycube Mapping

Given the initial polycube P, corner point correspondences $\mathcal{V}_{CS}$, $\mathcal{V}_{CP}$, and cube edges $\mathcal{E}_{CP}$, we compute an initial polycube mapping $f : S \to P$ as follows. Denote the position of each vertex v on S as $X = (x^0, x^1, x^2)$ and its image on the polycube as $U = f(X) = (u^0, u^1, u^2) \in P$; also denote three components of the vector function f as f^0, f^1, and f^2.

A discrete harmonic parametrization [32] is a bijective map from S to a 2D (u,v)-domain, $h : S \to D, S \subset \mathcal{R}^3, D \subset \mathcal{R}^2$ such that the discrete harmonic energies of both u and v components are minimized. When the target planar domain D is convex, and a diffeomorphic boundary mapping is given, the harmonic mapping h is bijective. Therefore, we decompose S to multiple patches, each of which will be mapped to a rectangle facet P_i on P. The harmonic energy of a mapping function on k-th (k=1,2,3) component is defined as

$$H^k = \frac{1}{2} \sum_i \sum_{v_j \in N(v_i)} w_{ij} (f^k(X_i) - f^k(X_j))^2, \tag{4.1}$$

where $N(v_i)$ is the set of all 1-ring neighboring vertices of v_i. $w_{ij} = \frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij})$ is the well known cotangent weight [32] defined on the edge $[v_i, v_j] \in \mathcal{E}_S$, where α_{ij} and β_{ij} are two angles opposite to the edge $[v_i, v_j]$.

For each polycube edge in $[v_{pi}, v_{pj}] \in \mathcal{E}_{CP}, v_{pi}, v_{pj} \in \mathcal{V}_{CP}$, we trace curves to connect their corresponding points $v_{si}, v_{sj} \in \mathcal{V}_{CS}$ using shortest paths following algorithms introduced in

[79]. After this, the harmonic mapping computation is straightforward. We parameterize these traced paths to polycube edges using the arc-length parametrization. On each facet of the polycube, corner and edge mapping decides the boundary condition and the interior mapping can be computed by solving two sparse linear systems [32].

4.3 Optimizing Polycube Domain

Given a polycube mapping $f : S \to P = \{f_i : S_i \to P_i\}$ defined on a set of topological rectangle patches on S. We want to find the optimal re-scaled P_i so that mapping distortion is minimized. We use a distortion energy E composed of the harmonic energies $H^t(f), t = 0, 1, 2$ and an area-stretching term $A(f)$.

$$H^t = \sum_{P_k} H^t_k = \sum_{P_k} \left(\sum_{e_{i,j} \in P_k} \frac{1}{2} w_{ij} (f^t(X_i) - f^t(X_j))^2 \right); \tag{4.2}$$

$$A = \sum_{P_k} \sum_{F_{i,j,h} \in P_k} \frac{(\Delta(U_i, U_j, U_h))^2}{\Delta(X_i, X_j, X_h)}; \tag{4.3}$$

$$E = H^0 + H^1 + H^2 + \alpha A; \tag{4.4}$$

where $\Delta(X_i, X_j, X_h)$ and $\Delta(U_i, U_j, U_h)$ denote the original area of triangle (v_i, v_j, v_h) and the area of its image under the mapping; P_k is a facet of polycube and $F_{i,j,h}$ is a triangle on this facet; α is a weighting factor balancing the harmonic and area-stretching terms.

When optimizing the polycube shape, we restrict our re-scaling on P_i such that (1) it preserves the total area of the polycube, and (2) it doesn't increase the number of corner points. Specifically, we divide the polycube P into different rectangular facets in each coordinate plane (see Figure 4.4).

First, we sort the coordinates of all corner points in three axes, and denote them as $\{\alpha^i_j\}, i = 0, 1, 2, j = 0, \ldots, N_i$. We translate the left-bottom of the polycube to the origin, so that any $\alpha^i_0 = 0$.

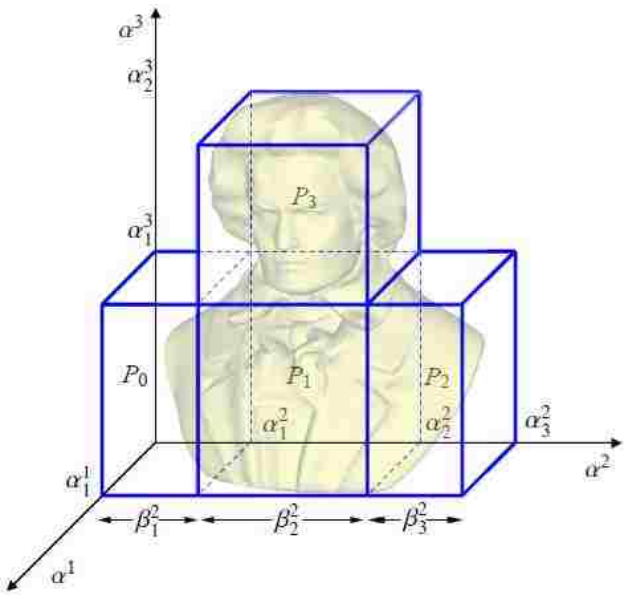


Figure 4.4. Definition of Polycube Coordinates and Parameters.

Then supposing a facet P_k is perpendicular to the u^t coordinate axis, we (1) denote the coordinate of P_k in u^t axis as α_j^t, and (2) on each patch perpendicular to u^t, denote its corresponding coordinates as $[\alpha_j^{t+1}, \alpha_{j+1}^{t+1}]$ and $[\alpha_j^{t+2}, \alpha_{j+1}^{t+2}]$. The superscript indicates the corresponding axis (u^0, u^1, or u^2), so $t+1$ actually denotes $(t+1)$ *mod* 3. In our following derivations, the addition of superscripts denotes their addition modulo 3.

Now we can denote the length of each segment in u^t-axis as $\beta_{j+1}^t = \alpha_{j+1}^t - \alpha_j^t$; and adjacent facets (faces connected by a same polycube edge) should share a same corresponding scaling factor β, to prevent the increase of corner points.

Therefore, supposing a rectangle domain P_k is perpendicular to the axis $u^i, (i = 0, 1, 2)$, we denote the two corresponding segment lengths of the rectangle as $\beta^{i+1}(P_k), \beta^{i+2}(P_k)$, their initial lengths as $\widetilde{\beta}^{i+1}(P_k), \widetilde{\beta}^{i+2}(P_k)$, initial harmonic energies as $\widetilde{H}_{P_k}^{i+1}, \widetilde{H}_{P_k}^{i+2}$, and initial area stretching energy as $\widetilde{A}_{P_k}$. These constants $\widetilde{\beta}^{i+1}(P_k), \widetilde{\beta}^{i+2}(P_k), \widetilde{H}_{P_k}^{i+1}, \widetilde{H}_{P_k}^{i+2}, \widetilde{A}_{P_k}$ are determined by the initial mapping. Then the harmonic energy of all sub-patches that are perpendicular to u^i, with respect to the their scalings can be written as:

$$\begin{aligned} &E_H^i(\alpha_1^i, \ldots, \alpha_{N_i-1}^i, \beta_1^i, \ldots, \beta_{N_i-1}^i) \\ &= \textstyle\sum_{P_k} \left(\beta^{i+1}(P_k)\right)^2 \widetilde{C}_k^{i+1} + \left(\beta^{i+2}(P_k)\right)^2 \widetilde{C}_k^{i+2}, \end{aligned} \tag{4.5}$$

where $\widetilde{C}_k^{i+1}$ and $\widetilde{C}_k^{i+2}$ are constants decided by the initial mapping:

$$\widetilde{C}_k^{i+1} = \left(\frac{\widetilde{H}_{P_k}^{i+1}}{\left(\widetilde{\beta}^{i+1}(P_k)\right)^2} \right), \widetilde{C}_k^{i+2} = \left(\frac{\widetilde{H}_{P_k}^{i+2}}{\left(\widetilde{\beta}^{i+2}(P_k)\right)^2} \right);$$

Considering all three axes, the global harmonic energy of the polycube mapping is:

$$\begin{array}{l} E_H\left(\{\alpha_j^i, \beta_j^i\}, \forall i = 0,1,2, j = 1, \ldots, N_i - 1\right) \\ = E_H^1(\alpha_1^0, \ldots, \alpha_{N_1-1}^0, \beta_1^0, \ldots, \beta_{N_1-1}^0) \\ +E_H^2(\alpha_1^1, \ldots, \alpha_{N_2-1}^1, \beta_1^1, \ldots, \beta_{N_2-1}^1) \\ +E_H^3(\alpha_1^2, \ldots, \alpha_{N_3-1}^2, \beta_1^2, \ldots, \beta_{N_3-1}^2); \end{array} \tag{4.6}$$

The area stretching term of the mapping is :

$$\begin{array}{l} E_A\left(\{\alpha_j^i, \beta_j^i\}, \forall i = 0,1,2, j = 1, \ldots, N_i - 1\right) \\ = \sum_{P_k} \left(\beta^{i+1}(P_k)\beta^{i+2}(P_k)\right)^2 \widetilde{C}_k; \end{array} \tag{4.7}$$

where $\widetilde{C}_k$ is a constant decided by the initial mapping:

$$\widetilde{C}_k = \left(\frac{\widetilde{A}_{P_k}}{\left(\widetilde{\beta}^{i+1}(P_k)\widetilde{\beta}^{i+2}(P_k)\right)^2} \right).$$

Finally, we have the entire distortion energy:

$$E(\{\alpha_j^i, \beta_j^i\}) = E_H + E_A; \tag{4.8}$$

subject to the constraints:

$$\begin{cases} \alpha_1^i = \beta_1^i, \\ \alpha_1^i + \beta_2^i = \alpha_2^i, \\ \alpha_2^i + \beta_3^i = \alpha_3^i, \\ \ldots \\ \beta_j^i > 0, \forall j = 1, \ldots, N_i, i = 0,1,2, \\ \sum_{P_k} \beta^{i+1}(P_k)\beta^{i+2}(P_k) = \widetilde{Area}, \end{cases} \tag{4.9}$$

where $\widetilde{Area} = \sum_{P_k} \widetilde{\beta}^{i_1}(P_k)\widetilde{\beta}^{i_2}(P_k)$; the last equation preserves the total area of the polycube domain. Figure 4.5 shows an example of an optimized polycube for the Beethoven model

based on the initial polycube mapping. The original polycube (b) is re-scaled to (d); as the grid texture mapping visualized, the distortion of the original mapping (a) reduces when the polycube shape changes (f); as in the zoom-in view (e), the angle distortion is smaller than that in (c).

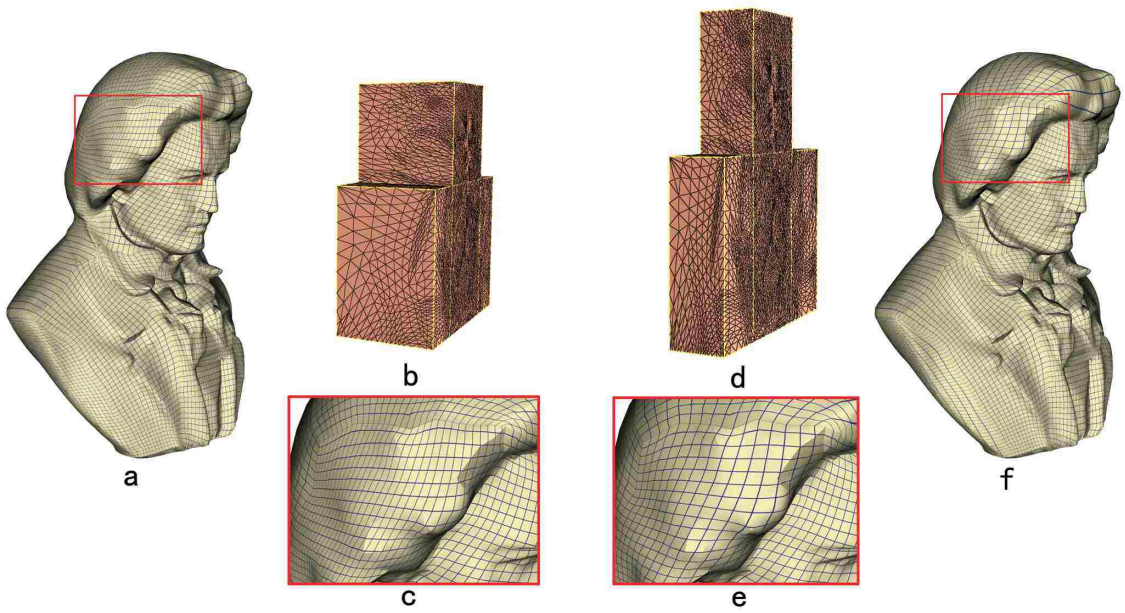


Figure 4.5. Polycube domain optimization. (a)-(c) shows the initial polycube domain and mapping. (d)-(f) shows the optimized polycube domains. Note the improvement of the checkerboard texture mapping between (c) and (e).

4.3.1 Barzilai-Borwein Gradient Projection Optimization Algorithm

In order to solve the energy $E(\{\alpha_j^i, \beta_j^i\})$ in equation (4.8) subject to constraints in equation (4.9), we will strictly enforce all the bound and linear constraints, and put the last nonlinear constraint as a penalty term $\lambda \left(\sum_{P_k} \beta^{i+1}(P_k)\beta^{i+2}(P_k) - \widetilde{Area} \right)^2$ in the objective function. As a result, this optimization problem could be formulated as minimization of a nonlinear function with bound and linear constraints, i.e.,

$$\begin{aligned} \min \quad & F \quad E(\mathbf{x}) \\ s.t. \quad & \mathbf{x} \in \Omega := \{\mathbf{x} : A\mathbf{x} = \mathbf{b},\ \mathbf{b_l} \leq \mathbf{x} \leq \mathbf{b_u}\}, \end{aligned} \tag{4.10}$$

where $\mathbf{x} \in \mathcal{R}^n$ is the vector of variables $\{\alpha_j^i, \beta_j^i\}$, $n = 2(N_1 + N_2 + N_3)$, $\mathbf{b_l}$ and $\mathbf{b_u} \in \mathcal{R}^n$ are the bound constraints, and A is an m by n matrix with $\mathbf{b} \in \mathcal{R}^m$ denoting the linear constraints. Although the objective function is continuously differentiable, the dimension n of our reformulated problem generally can be large, and the explicit computation of the Hessian is difficult. Hence, first order method, which only requires gradient information, is preferred. To solve (4.10), we develop the following nonmonotone gradient projection algorithm, which is also an iterative algorithm: given the starting $\mathbf{x}_0$, our algorithm takes the following iterations

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k, \tag{4.11}$$

where k is the iteration number, α_k is a stepsize and $\mathbf{d}_k$ is the searching direction defined as

$$\mathbf{d}_k = P_\Omega(\mathbf{x}_k - \frac{1}{\lambda_k^{BB}}\mathbf{g}_k) - \mathbf{x}_k.$$

Here, P_Ω is the projection on the feasible set Ω, $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ and λ_k^{BB}, $k \geq 1$ is the so called Barzilai-Borwein [80] stepsize parameter generated by satisfying a quasi-Newton property, i.e.

$$\lambda_k^{BB} = \arg\min_{\lambda \geq \lambda_0} \ \|\mathbf{\Lambda}(\lambda)\mathbf{s}_{k-1} - \mathbf{y}_{k-1}\|_2, \tag{4.12}$$

where $\mathbf{s}_{k-1} = \mathbf{x}_k - \mathbf{x}_{k-1}$, $\mathbf{y}_{k-1} = \mathbf{g}_k - \mathbf{g}_{k-1}$, $\mathbf{\Lambda}(\lambda_k) = \lambda_k \mathbf{I}$, and λ_0 is a positive constant. Hence, the proposed λ_k^{BB}, when $k \geq 1$, obtained from (4.12), is

$$\lambda_k^{BB} = \max\left\{\frac{\mathbf{s}_{k-1}^\mathsf{T}\mathbf{y}_{k-1}}{\mathbf{s}_{k-1}^\mathsf{T}\mathbf{s}_{k-1}}, \lambda_0\right\}, \tag{4.13}$$

and λ_0^{BB} can be arbitrarily defined as a positive number and we set $\lambda_0^{BB} = \|\mathbf{g}(\mathbf{x}_0)\|_\infty$ and $\lambda_0 = 10^{-10}$ in practice. This BB initial stepsize (4.13) has been extensively studied recently and been shown to perform much better than steepest descent type gradient projection methods [81, 82]. However, to maintain the efficiency, the stepsize α_k in (4.11) must be obtained by a nonmonotone line search. In our experiments, we use the non-monotone line search developed in [83, 84].

4.4 Optimizing Polycube Mapping

In Section 4.3, we fix the corner point mapping $f(\mathcal{V}_{CS}) \rightarrow \mathcal{V}_{CP}$ to optimize the shape of polycube domain. We further reduce the mapping distortion by moving vertices $\mathcal{V}_{CS}$ (without ambiguity, we also call them corner points) over S. Any 2-dimensional manifold S can be parameterized to an atlas $\Omega = \{\Omega_i\}$, and locally any point on S: $(x^1, x^2, x^3) \in S$ can be represented as a 2D coordinate (r^1, r^2) on a local planar chart. We construct local parametrization $g_i : S_i \rightarrow \Omega_i$ by mapping the C-ring neighboring regions (in our experiments, we set $C = 20$) of each initial corner point $\in \mathcal{V}_{CS}$ to a unit disc Ω_i. Any neighboring points on the domain Ω_i are continuously parameterized. Let N be the number of the corner points $N = |\mathcal{V}_{CP}|$. The optimization will be conducted on all charts $\{\Omega_1, \ldots, \Omega_N\}$ simultaneously by searching the optimal N corner points, represented as coordinates $(\{r_1, r_2, \ldots, r_{2N-1}, r_{2N}\})$, where (r_{2k-1}, r_{2k}) corresponds to (r^1, r^2) on chart Ω_k.

This problem is formulated as minimizing the distortion energy E of the map f decided by the corner maps:

$$E(r_1, r_2, \ldots, r_{2N}) = H^1(f) + H^2(f) + H^3(f) + A(f), \tag{4.14}$$

the harmonic energies and area stretching of function f are defined following equation (4.2) and equation (4.3).

For polycube mapping with N corner points, the dimension of this optimization problem is $2N$. f is determined by these $2N$ parameters, and can be efficiently computed (Section 4.4.1), but since we need to retrace the shortest paths as the sub-patch boundaries, we do not have the closed form for f or its derivative. Therefore, we use a derivative-free optimizer (Section 4.4.2) to solve this problem.

As indicated in Algorithm 3, we iteratively perform domain optimization (Section 4.3) and mapping optimization (this section) until the polycube domain does not change. Despite the optimization of both the domain shape and the corner mapping, the angle distortion near

the subregion boundary (e.g. polycube corners, edges) can be large due to the usage of harmonic mapping with fixed boundary. We perform a smoothing process to further reduce the distortion. Smooth transition functions [85] can be easily computed between adjacent polycube faces, then parameterization/smoothing can be computed on a flattened domain covering this boundary region. We adopt the smoothing algorithm of [86] to refine the map near polycube corner/edge regions.

Figure 4.6 illustrates an iteration of domain mapping optimization on a Beethoven model. Corners in (a) are adjusted to new positions (f). Meanwhile, the mapping distortion energy reduces, which can also be visualized in the zoom-in regions (d,e vs b,c). If we perform an aforementioned smoothing, the distortion near the boundary region can be further reduced (f,g).

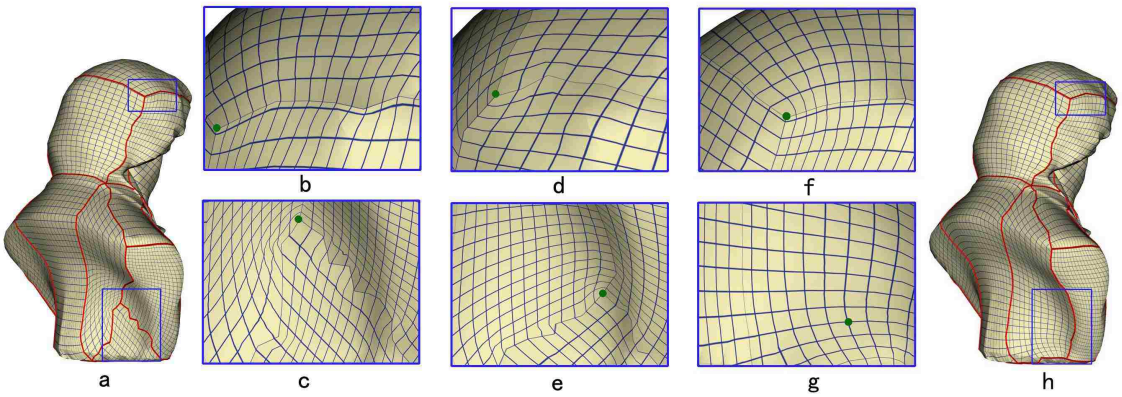


Figure 4.6. Polycube Mapping Optimization. (a) is the model before mapping optimization. (b,c) zoom in to show the distortion before this step. (d,e) illustrate the distortion after mapping optimization. (g,f) show distortion after the smoothing postprocess. (h) is model after smoothing. The corner points are shown in green. With the smoothing, distortion and discontinuity across sub-region boundaries significantly reduces.

Figure 4.7 shows an iteration of our polycube map optimization on the horse model; the initial horse mapping (a) on a polycube with 60 corner points is optimized; the resultant mapping (b) has smaller angular and area distortion.

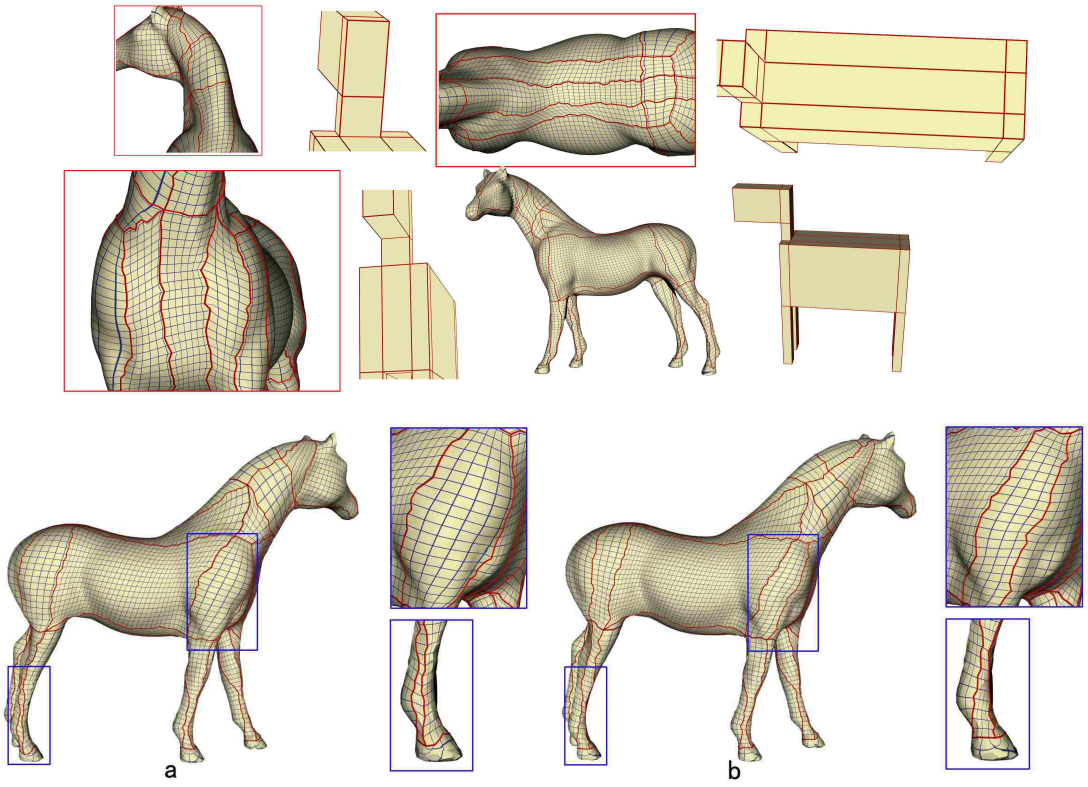


Figure 4.7. Mapping Optimization of The Horse Model on a Polycube (upper row) with 60 Corner Points. The lower row shows the moving of corner points: (a) before optimization, (b) after optimization.

4.4.1 Efficient Mapping Recomputation

The typical computation for harmonic surface mapping on each rectangle sub-patch involves solving two systems of linear equations. This can be time consuming when we need to recompute it and re-evaluate its distortion in every step during the optimization. Since the boundary condition of the mapping always changes gradually, we can utilize a more efficient linear equation updating algorithm CHOLMOD [87] to accelerate the mapping recomputation.

Mapping on each sub-patch is harmonic, so the coefficient matrix is sparse, symmetric and positive definite. This special property makes it feasible to utilize Cholesky decomposition to solve and update the linear systems very quickly. Initially, we precompute the shortest paths between all pairs of vertices using the Floyd-Warshall algorithm and store predecessor

matrices on shortest paths. This takes $O(n^3)$ preprocessing time, where n is the number of vertices. During each iteration, when corner points are replaced by some of their neighboring points, between each pair of corners, we retrace corresponding shortest paths in $O(k)$ time where k is the number of vertices on this path. The coefficient matrix only changes slightly (a few rows and columns proportional to the number of mutable boundary conditions due to the change of corner points). This infers an efficient solution-update algorithm. Davis and Hager [87] proposed an approach of dynamic supernodal sparse Cholesky update and downdate, which produces a solution for the newly update linear system without repeatedly computing the coefficient matrix and solving the system. After an initial Cholesky decomposition at a cost of $O(n^3)$, the decomposition can be updated in $O(N)$, where N is the number of changed entries in Cholesky factor, which is typically much smaller than the size of the mesh, leading to efficient harmonic mapping update. The similar approach was introduced to graphics and shape modeling [88] for dynamically updating harmonic fields design.

With this efficient mapping update technique, we can re-evaluate the objective function for a given new planar coordinates for corner points on S. Since the parameterization (and therefore the corner selection) is continuous, we dynamically split each corresponding triangle (where each parametric corner point locates) into three and update the accumulated energy accordingly.

4.4.2 Derivative-free Optimization Algorithm

The objective function (4.14) can be reformulated in the following format

$$\begin{aligned} \textbf{min} \quad & \Phi(\mathbf{x}) = \textstyle\sum_i^m \mathrm{sgn}(i)\ f_i^2(\mathbf{x}), \\ \textbf{s.t.} \quad & \mathbf{b_l} \leq \mathbf{x} \leq \mathbf{b_u}, \end{aligned} \tag{4.15}$$

where $\mathbf{x} \in \mathcal{R}^n$, $\mathbf{b_l}$ and $\mathbf{b_u} \in \mathcal{R}^n$ are the bound constraints, and $\mathrm{sgn}(i) = \pm 1$ is the sign in front of the squares of f_i, $i = 1, \ldots, m$. The main difficulty of solving this problem is that

the explicit derivatives are not available. We develop a trust region based derivative-free algorithm in spirit similar to the approach proposed in [89]. Our algorithm does not require the derivative information of the objective function, nor does it explicitly approximate the derivative. Instead, at each iteration it builds a local quadratic model of the objective function by multivariate interpolation in combination with trust region techniques. More specifically, at each iteration, the algorithm adaptively chooses a set of interpolation points $\mathbf{Y}_k$, with $(n+1) \leq |\mathbf{Y}_k| \leq (n+1)(n+2)/2$, where k is the iteration number and $|\mathbf{Y}_k|$ is the cardinality of $\mathbf{Y}_k$. Our algorithm takes the following major steps:

Step 0 **(Initialization)** Set up initial starting guess $\mathbf{x}_0$, trust region radius Δ_0 and sampling points $\mathbf{Y}_0$. Build initial trust region model on $\mathbf{Y}_0$ and set $k = 0$.

Step 1 **(Criticality step)** Choose a base point $\mathbf{y}_k \in \mathbf{Y}_k$ and calculate the gradient of our model. If the gradient is sufficiently small, stop. Otherwise, make sure the model is well-posed [89] in a trust region with radius proportional to the norm of model gradient.

Step 2 **(Step calculation)** Solve the following trust region subproblem :

$$\begin{aligned} \textbf{min} \quad & \phi_k(\mathbf{d}), \\ \textbf{s.t.} \quad & \|\mathbf{d}\| \leq \Delta_k \\ & \mathbf{l} \leq \mathbf{x}_k + \mathbf{d} \leq \mathbf{u}, \end{aligned} \tag{4.16}$$

where $\phi_k(\mathbf{d})$ is a local quadratic model of $\Phi(\mathbf{x})$ in a trust region with radius Δ_k. Here, $\|\cdot\|$ is the 2-norm.

Step 3 **(Acceptance of the trial step)** Compute the ratio of actual and predicted function reduction

$$r_k = \frac{\Phi(\mathbf{x}_k) - \Phi(\mathbf{x}_k + \mathbf{d}_k)}{\phi_k(\mathbf{0}) - \phi_k(\mathbf{d}_k)},$$

where $\mathbf{d}_k$ is the minimizer of (4.16). If $r_k > 0$, then $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$; otherwise, $\mathbf{x}_{k+1} = \mathbf{x}_k$.

Step 4 **(Trust region radius update)** Update trust region radius by

$$\Delta_{k+1} = \begin{cases} \frac{1}{2}\,\|\mathbf{d}_k\| & \text{if } r_k < 0.1, \\ \max\{\frac{1}{2}\Delta_k,\ \|\mathbf{d}_k\|\} & \text{if } 0.1 \le r_k < 0.7, \\ \max\{\Delta_k,\ 2\|\mathbf{d}_k\|\} & \text{if } r_k \ge 0.7, \end{cases}$$

If $r_k \ge 0.1$, form $\mathbf{Y}_{k+1}$ from $\mathbf{Y}_k$ by merging new point $\mathbf{x}_{k+1}$. Set $k = k + 1$, go to Step 1.

Step 5 **(Model improvement)** This step applies only when $r_k < 0.1$. In this case, before shrinking the trust region radius, make sure the model is well-posed [89] in the current trust region. Set $k = k + 1$, go to Step 1.

One critical advantage of this algorithm is using the least Frobenius norm updating strategy [90] to update the quadratic model (4.16). Hence, to build our quadratic model, we only need $\mathcal{O}(n)$ (in our experiments, $2n + 1$) function evaluations, while normally $(n + 1)(n + 2)/2 = \mathcal{O}(n^2)$ number of valuations are required for building a fully quadratic model (Note, $(n+1)(n+2)/2$ could be much bigger than $2n+1$ for relatively large n). In addition, at each iteration, only one new function evaluation is required to update the local quadratic model.

Therefore, our approach is usually more efficient [91, 89] than other widely used strategies in derivative-free optimization, such as using finite-difference to approximate derivatives [92] or some direct search methods [93]. Global convergence of the algorithm as well as the good local geometry of the set of interpolation points are guaranteed by trust region techniques [89][90].

4.5 Polycube Mapping for Multiple Objects

We also demonstrate an application of our polycube mapping framework in multiple objects mapping. Polycube can be used as a canonical base domain for multiple objects (preferably, these objects have the same topology and similar geometry). Our framework can be used to generate such a common regular domain, and multiple objects are parameterized onto this single polycube with low distortion. Multiple shapes can be analyzed, processed, and

integrated over this single domain. Supposing we have a set of models $\{S_1, \ldots, S_n\}$ to be integrated. We construct a common polycube P using S_1. We also compute initial mapping f_i between P and each $S_i, i = 1, \ldots, n$. Then simultaneously, we optimize P and the mapping $f_i : S_i \to P$ using the above proposed framework. The final polycube domain P is the one that minimizes the total distortion of multiple polycube parametrization $E = \sum_i E(f_i)$. The final polycube is an optimal domain for all these models. Inter-surface mapping between two models S_i and S_j can be composed and optimized over the this domain as $f_{i,j} : S_i \to S_j = f_j^{-1} \circ f_i$. We visualize our optimal polycube and the mapping results using inter-object morphing by linearly interpolating them over the common polycube domain.

Specifically, we construct initial polycube P for S_1 and use projection to determine corner points mapping. However, this simple projection approach does not work well when we map P to other models $S_2, S_3, \ldots, S_n$, especially when S_i is not geometrically similar to P. Especially for this situation (when we want to map a surface to a dissimilar polycube), we compute the initial polycube mapping in the following more robust way (Note that any other suitable polycube mapping approach can also be used to generate initial f_i). We partition P and each S_i consistently (i.e. the segmentation of P and S_i has the isomorphic dual graph); then compute the mapping $f_i : S_i \to P$ by merging all individual sub-region mappings. Such an approach based on canonical pants decomposition is introduced in [94]. We briefly recap the basic idea, and refer readers to [94] for details. The pants patch is a genus-0 surface with 3 boundaries. Any surface (except for a few trivial cases) can be decomposed into a set of pants patches, including g handle patches and a base patch, where g is the genus. The base patch is then further iteratively partitioned into a set of pants patches. Finally, every pants patch is decomposed into two sub-patches, each of which can be parameterized on a regular planar hexagon. Therefore, the global surface mapping between two objects is composed by parameterizations of sub-patches on these hexagonal domains. This approach can easily and robustly handle the surface mapping between two objects with arbitrary topology and feature

points, therefore it is suitable here for generating initial mapping $f_i : S_i \to P$. Figure 4.8 shows an example of using the above approach to construct optimal common polycube for the horse and the cow. Individually optimal polycubes for the horse and cow are shown in (b) and (c), and initial polycube maps are visualized in (a) and (d); the optimal common polycube is shown in (f). Specifically, a compromise can be seen in the neck region. The final common polycube mappings are visualized in (e) and (g).

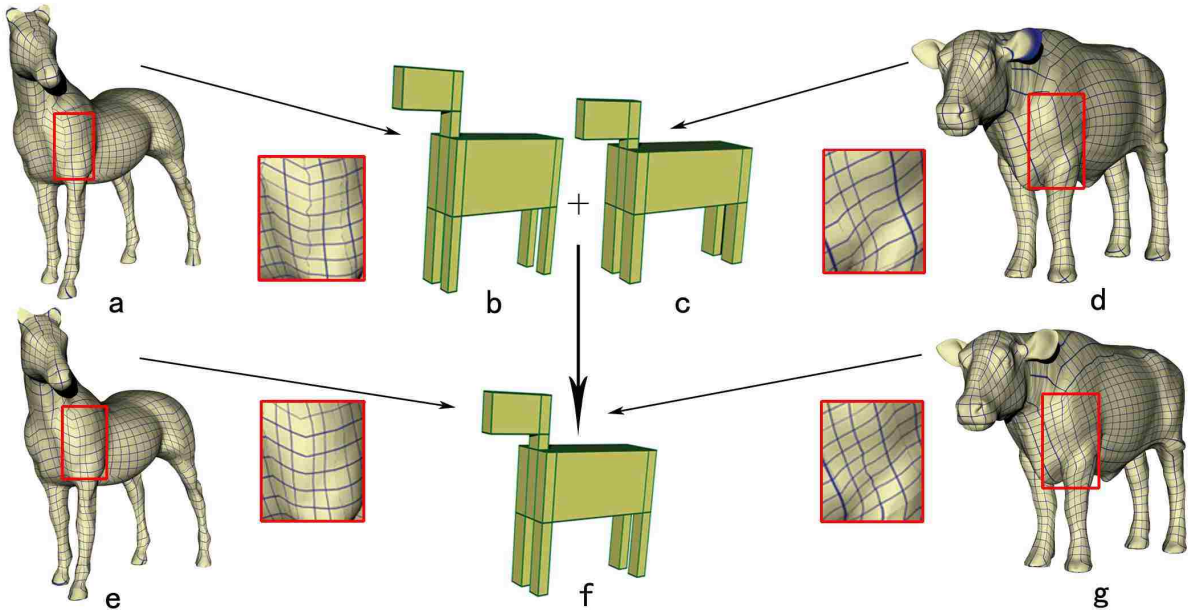


Figure 4.8. Common Polycube Mapping for Multiple Models. Initial polycube maps of the horse and cow are as (a) and (d); individually-optimal polycube domains are shown in (b) and (c); the common optimal polycube domain is shown in (f); and the final common optimal polycube mapping of both models are as (e) and (g). Note: the common polycube balances both individually-optimal polycubes, see the neck region.

4.6 Experimental Results and Discussions

We compare the properties of our polycube mapping framework with existing methods and list them in Table 4.1. Our method generates the optimal polycube within the same topological class, and the complexity of the polycube is flexibly bounded by the given number of singularities. We test our optimization framework on a few 3D shapes. Figure 4.9 shows

the optimization on Bimba and Max-planck. The texture-mapped rectangular grids become closer to squares, indicating the reducing of angle distortion.

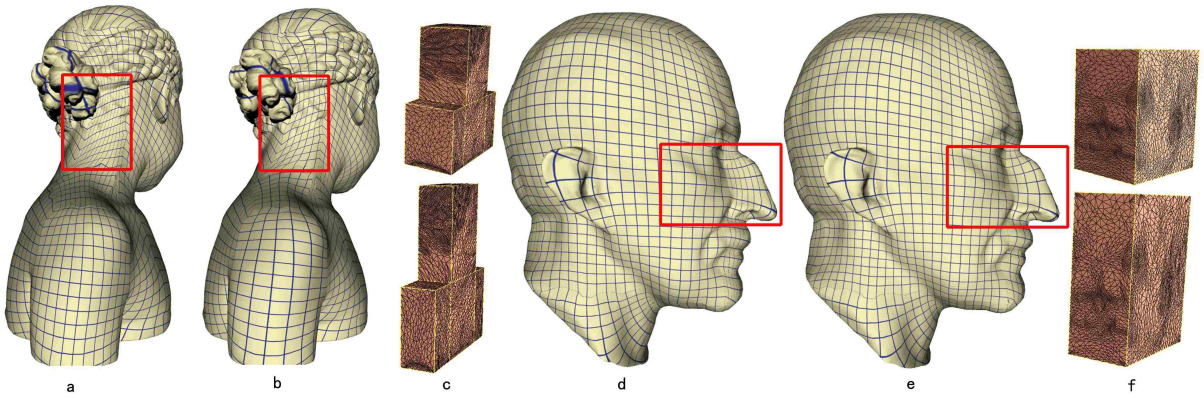


Figure 4.9. Polycube Mapping of Bimba and Max-Planck. (a,d) initial mapping, (b,e) optimized mapping. The texture mappings of grids show the reduction of angle distortions after the optimization. (c,f) initial polycube (in upper row) and optimized polycube (in lower row) domains.

Table 4.1. Comparisons of Different Polycube Mapping Methods. *PC Constr.*, *Opt. PC*, *Sing. Control*, *Common PC* indicate whether polycube construction can be automatic, whether polycube shape is optimal, whether polycube complexity can be controlled by the given restriction on singularity number, and whether it can be used to construct a canonical domain for multiple objects, respectively.

Methods	PC Constr.	Opt. PC	Sing. Control	Common PC
Tarini[6]	manual	no	manual	no
Wang[9]	manual	no	manual	no
Wang[10]	manual	no	manual	no
Lin[52]	auto.	no	no	no
He[53]	auto.	no	yes	no
Ours	auto.	yes	yes	yes

Figure 4.10 shows a common polycube parameterization for multiple objects. We parameterize the horse, cow, and goat onto an optimized common polycube domain. (a-c) visualize the geometry represented on the polycube parameterization (using the connectivity of the polycube), then we can easily interpolate them and generate a "mixed creature". (d) shows

an interpolated shape with 20%-horse, 50%-goat, and 30%-cow. Features of horse, goat, and cow can be seen on the final interpolated shape.

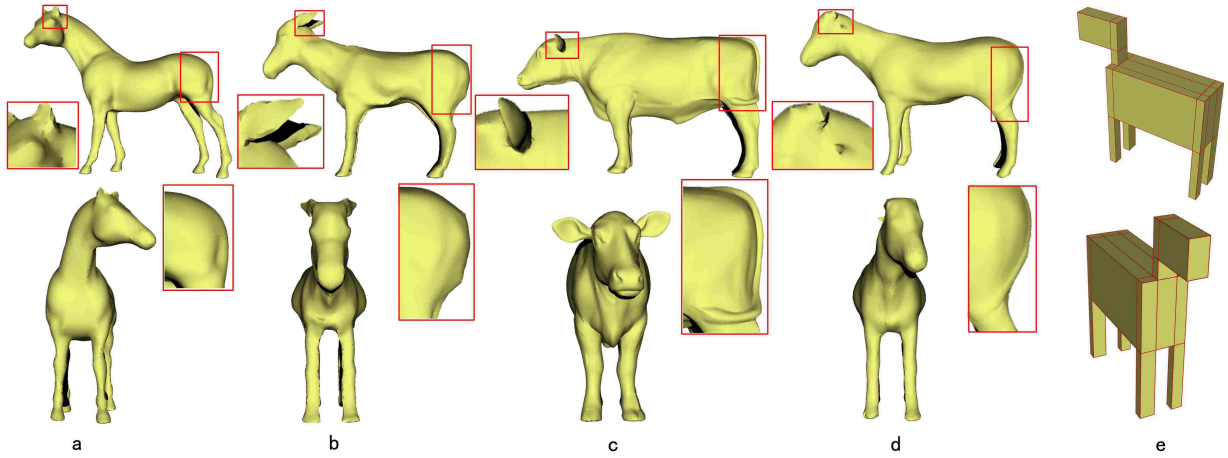


Figure 4.10. Integration of Multiple Objects over a Common Polycube Domain. The horse (a), goat (b), and cow (c) are blended in a this polycube domain. Features from the original models can still be seen in the interpolated shape (e.g. the mouth and neck of the horse, ears of the goat, and the tail of the cow).

The quality of polycube mapping can be measured by area distortion ϵ_{area} and angle distortion ϵ_{angle} [95].

$$\epsilon_{area}(T) = \frac{area(\Delta_{M'}(T))}{area(\Delta_M(T))} + \frac{area(\Delta_M(T))}{area(\Delta_{M'}(T))}$$

$$\epsilon_{angle}(T) = \frac{cot\alpha|a^2| + cot\beta|b^2| + cot\gamma|c^2|}{2area(\Delta_M(T))}$$

The closer the values of ϵ_{area} and ϵ_{angle} is to 1, the better the quality of polycube mapping we get. The statistics and performance of our test cases are reported in Table 4.2.

Intuitively, the more complicated the polycube domain is used, the more freedom we have to optimize its shape. Moreover, generally when the polycube is closer to the original model, we can get a less distorted/stretched polycube mapping. Figure 4.11 illustrates an example on the Beethoven model. When only one cube is used as the parameterization domain, the distortion is larger (a,b), compared with the mapping constructed on a more complicated polycube domain (c,d). On the other hand, a more complicated polycube domain indicates

Table 4.2. Runtime Table: $\#\Delta$ (number of triangles); $\#C$ number of corner points, ϵ^0_{angle} and ϵ^0_{area} are angle and area distortions before optimization; ϵ_{angle} and ϵ_{area} are distortions after optimization; T_1 and T_2 is the execution time for domain optimization and mapping optimization (in seconds).

Models	$\#\Delta$	$\#C$	ϵ^0_{angle}	ϵ^0_{area}	ϵ_{angle}	ϵ_{area}	T1	T2
Isis	5K	8	1.261	1.429	1.134	1.385	0.52	112
Beethoven	21K	20	1.387	1.563	1.215	1.236	7.74	504
Max-Planck	10K	8	1.104	1.477	1.060	1.395	1.36	33
Bimba	30K	20	1.292	1.243	1.283	1.209	10.62	744
horse	16K	60	1.352	1.302	1.258	1.229	11.72	1842
cow	39K	60	1.198	1.210	1.191	1.161	21.21	2898
goat	21K	60	1.359	1.304	1.241	1.190	10.83	2032

more corner points (singularities) [9] and potentially more-distorted parameterization across sub-region boundaries.

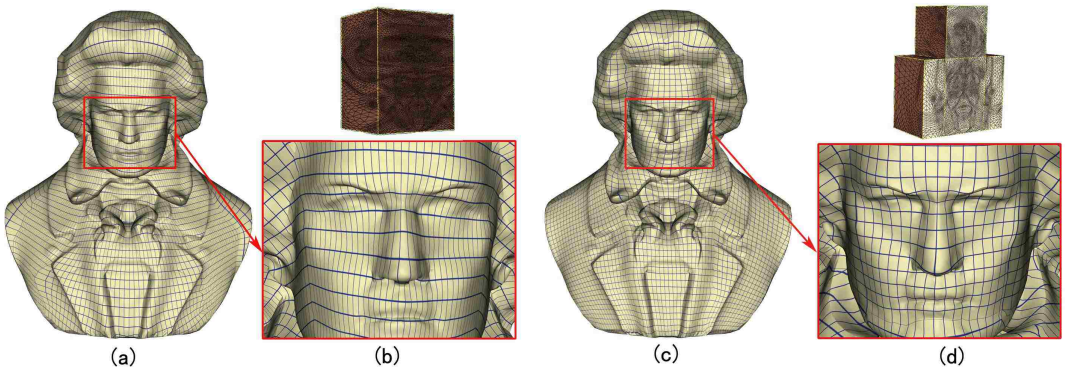


Figure 4.11. Different Initial Corner Budgets. With increase of the initial budget (from 8 to 20), the mapping quality is improved (from a,b to c,d).

We also adjust the weighting factor α in Eq 4.4 to see different mapping results. Table 4.3 shows the different angle and area distortion under different settings. $\alpha = 1.0$ was used when we perform our other experiments. Figure 4.12 illustrates this mapping result. When the area term is emphasized, a more uniform but less conformal mapping is obtained (a,b); when α is small, the angle distortion is reduced (c,d).

Moreover, our approach also applies to high-genus models. In Figure 4.13, an example is provided on model Torus, where the polycube domain is optimized. (a) is the unoptimized

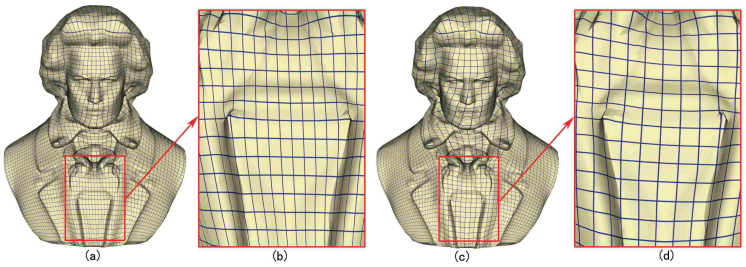


Figure 4.12. Different Weighting Factors. (a,b) Area-stretching term $\alpha = 1000$, (c,d) $\alpha = 0.01$.

Table 4.3. Testing different weighting on the area-stretching term (α in equation (4.4)), on the polycube-Beethoven mapping. ϵ_{angle} and ϵ_{area} are the corresponding angle and area distortion.

α	0.1	0.5	1.0	1.5
ϵ_{angle}	1.219	1.235	1.253	1.264
ϵ_{area}	1.380	1.316	1.292	1.281

polycube mapping result; (b) is the optimized polycuybe mapping result, where the domain is updated. The optimized polycube mapping provides better remeshing quality.

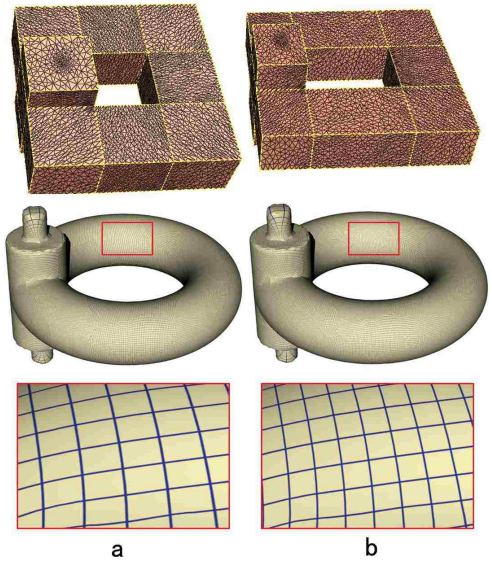


Figure 4.13. Domain Optimization on High Genus Model. (a) is the unoptimized polycube mapping result; (b) is the optimized polycuybe mapping result, where the domain is updated. The optimized polycube mapping provides better remeshing quality.

4.7 Summary

In this chapter, an interactive optimization framework to solve the optimal polycube mapping problem is proposed. Because directly solving optimal polycube domain and mapping together is too expensive, polycube domain shape and polycube mapping are iteratively optimized separatively, to make full use of the available partial derivative information of the objective function. An efficient non-linear optimization algorithm with linear bound constraints for the first sub-problem is developed to find a optimized polycube domain. For the second sub-problem, an efficient derivative-free solver is developed, making use of the summation-of-square structure of the objective function; and a fast mapping re-computation algorithm to accelerate the evaluation in the optimization process is proposed. The polycube mapping framework has been demonstrated effective in several experiments, and can be used to construct common polycube domains for multiple objects.

5 Volumetric Polycube Parameterization Guided By Frame Field

Solid volumetric data have richer contents than those of the boundary surface. When the data processing or analysis are related to material, intensity, or any other structural information defined over the whole 3D region of the object (instead of on just its boundary shell), we need to consider the shape as a 3-manifold and study the volumetric parameterization. Computing volumetric parameterization is a fundamental problem and is very important for geometric modeling and processing of solid data in scientific and engineering fields. It serves as an important preprocessing step in many tasks of CAD, CAE, CAM, medical image analysis and etc. Therefore, we would like to generalize the parameterization from the surfaces to the solids. In this chapter, volumetric polycube parameterization guided by frame field is presented. A frame is defined as a collection of three coupled unit orthogonal vectors $\mathbf{X} = (\mathbf{v}^0, \mathbf{v}^1, \mathbf{v}^2)$, which is usually represented by a 3×3 unitary matrix. A frame field contains a set of frames $\{X_i\}$ defined over a manifold.

5.1 Definitions

Given a solid M, a volumetric parameterization is defined as

$$f : M \to \Omega, \tag{5.1}$$

where $\Omega \subset R^3$. f is establishing three scalar parametric functions on the given model M. In the discrete representation, the solid M is given as a tetrahedral mesh containing a set of tetrahedra (or tets). In that case, f is piecewise linear function. The gradient or Jacobian

J of this vector function $f(x, y, z) = (f^u, f^v, f^w)$, $(x, y, z) \in M$ and $(f^u, f^v, f^w) \in \Omega$ is a matrix containing three column vectors

$$J = \nabla f = \begin{bmatrix} \nabla f^u & \nabla f^v & \nabla f^w \end{bmatrix} \tag{5.2}$$

where $\nabla f^u = [\frac{\partial f^u}{\partial x} \ \frac{\partial f^u}{\partial y} \ \frac{\partial f^u}{\partial z}]^T$, $\nabla f^v = [\frac{\partial f^v}{\partial x} \ \frac{\partial f^v}{\partial y} \ \frac{\partial f^v}{\partial z}]^T$ and $\nabla f^w = [\frac{\partial f^w}{\partial x} \ \frac{\partial f^w}{\partial y} \ \frac{\partial f^w}{\partial z}]^T$ are 3×1 vectors. Discretely, the gradient defined on tet-i is a constant denoted by ∇f_i. When the determinant on tet-i $\|\nabla f_i\| = 0$, we call it a **singularity**, since ∇f_i is singular. Generally, there exist singularities in the volumetric parameterization.

5.1.1 Objective Energy

As in Eq 5.2, the gradient of the parametric function and the frame field can be both represented by matrices. The parameterization can be controlled by a frame field through minimizing the different between the gradient and the frame field. Based on this idea, Nieser et al [20] propose CUBECOVER algorithm to compute a volumetric mapping by minimizing the functional:

$$E = \sum_i \|\nabla f_i - X_i\|^2 \cdot t_i \tag{5.3}$$

where ∇f_i indicates the gradient of the parametric function to be solved, and X_i indicates the frame defined in tet-i, and t_i is the volume of tet-i. In other words, the difference between the given frame field X_i and the gradient of the parametric function is to be minimized. The Frobenius norm is used here.

According to Wang et al [13], the gradient of a parametric function f defined on tet-i with respect to u, v, w coordinates can be discretized as 3×1 vectors, i.e. ∇f_i^u, ∇f_i^v and ∇f_i^w.

$$\nabla f_i^u = \frac{1}{3t_i} S\mathbf{f_i^u} \tag{5.4}$$

$$\nabla f_i^v = \frac{1}{3t_i} S\mathbf{f_i^v} \tag{5.5}$$

$$\nabla f_i^w = \frac{1}{3t_i} S\mathbf{f_i^w} \tag{5.6}$$

where $\nabla f_i = [\nabla f_i^u \quad \nabla f_i^v \quad \nabla f_i^w]$ is a 3×3 matrix containing three 3×1 vectors, and S is a 3×4 constant matrix resulting from signed triangle face area and face normal vectors. $\mathbf{f_i^u}$, $\mathbf{f_i^v}$ and $\mathbf{f_i^w}$ are three 4×1 vectors, indicating the parametric values of 4 vertices in tet-i.

5.1.2 Linear Constraints

The solid M is usually parameterized onto an atlas of local charts (topological disks). Each local chart is represented by a tet. A piece of the parametric function is defined for each tet. The volumetric parameterization is then defined by local mapping and the transition among them. Intuitively, given two adjacent tet-i and tet-j, and the local pieces of the parametric function are f_i and f_j respectively, the transition from f_i to f_j is usually described by rigid transformation, namely rotation and translation. Formally, they are related by

$$f_j = \Pi_{ij} f_i + \mathbf{g}_{ij} \tag{5.7}$$

where Π_{ij} is called matching matrix that can be any 3D rotation and $\mathbf{g}_{ij}$ is called gap vector that can be any 3D translation.

Therefore, on a face triangle Δpqr shared by tet-i, and tet-j, the parametric function f to be solved is required to satisfy the following transition functions:

$$\mathbf{f(p)_j} = \Pi_{ij} \mathbf{f(p)_i} + \mathbf{g_{ij}} \tag{5.8}$$

$$\mathbf{f(q)_j} = \Pi_{ij} \mathbf{f(q)_i} + \mathbf{g_{ij}} \tag{5.9}$$

$$\mathbf{f(r)_j} = \Pi_{ij} \mathbf{f(r)_i} + \mathbf{g_{ij}} \tag{5.10}$$

where $\mathbf{f(p)_j}$, $\mathbf{f(q)_j}$, and $\mathbf{f(r)_j}$ are three 3×1 vectors indicating the parametric values of three corners on Δpqr, respectively. They are unknown variables.

Singularity. Given an interior edge e surrounded by cyclically ordered tets $(t_0, ..., t_\beta)$, and any point p on e, the parametric function f should respect the transition functions in

Eq 5.8, 5.9, and 5.10, i.e.

$$f|_{t_1}(p) = \Pi_{t_0 t_1} f|_{t_0}(p) + g_{t_0 t_1},$$
$$f|_{t_2}(p) = \Pi_{t_1 t_2} f|_{t_1}(p) + g_{t_1 t_2}, \ldots$$

and after plugging each equation into its successor we get:

$$\begin{aligned} & f|_{t_0}(p) = type(e, t_0) \cdot f|_{t_0}(p) + \bar{\mathbf{g}} \\ \Leftrightarrow \quad & (Id - type(e, t_0)) f|_{t_0}(p) = \bar{\mathbf{g}} \end{aligned} \tag{5.11}$$

for some constant vector $\bar{\mathbf{g}} = [\bar{g}_0 \ \bar{g}_1 \ \bar{g}_2]^T$, which depends on the gap $g_{t_i t_{i+1}}$ where $i = 0, 1, \ldots, \beta - 1$. This equation is true for any point p on the edge. If $type(e, t0) = \Pi_{t_\beta t_0} \cdot \Pi_{t_{\beta-1} t_\beta} \cdots \Pi_{t_1 t_0}$ is not identity, the edge e is called a singularity. If Π_{ij} is chosen as identity and $\mathbf{g}_{ij}$ is chosen as zero vector, there is no interior singularity within the volume. However, other types may be introduced and details are discussed in chapter 6.

5.2 Experimental Results and Discussions

We compute volumetric parameterization for various solid models, in which polycube domain is constructed for given geometric models and used as the boundary constraints [4]. The optimization of the objective energy in Eq 5.3 leads to a system of linear equations. The volumetric parameterization is obtained by solving the linear system. The experiments are conducted on a workstation with 2.27 GHz Xeon CPU and 4GB memory.

Hexahedral Remeshing. Regular hex structure Ω_H can be generated on the polycube domain Ω. After the parameterization $f : M \to \Omega$ computed on the tetrahedral mesh of M, we simply resample all the vertices of Ω_H on M by f^{-1} using barycentric interpolation. Figures 5.1-5.3 illustrate our hex meshing results generated from the volumetric parameterization for various models.

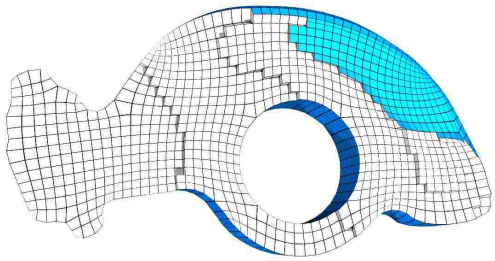
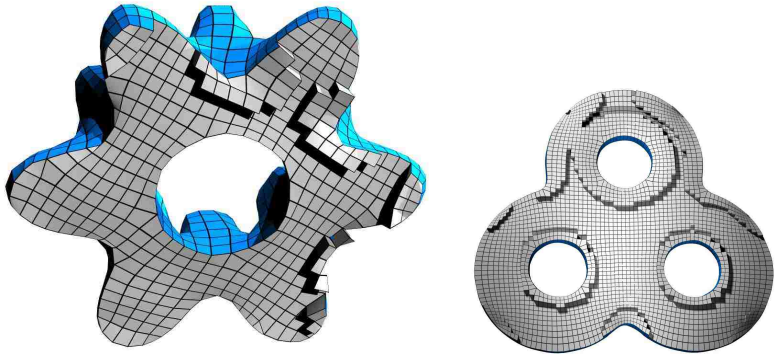
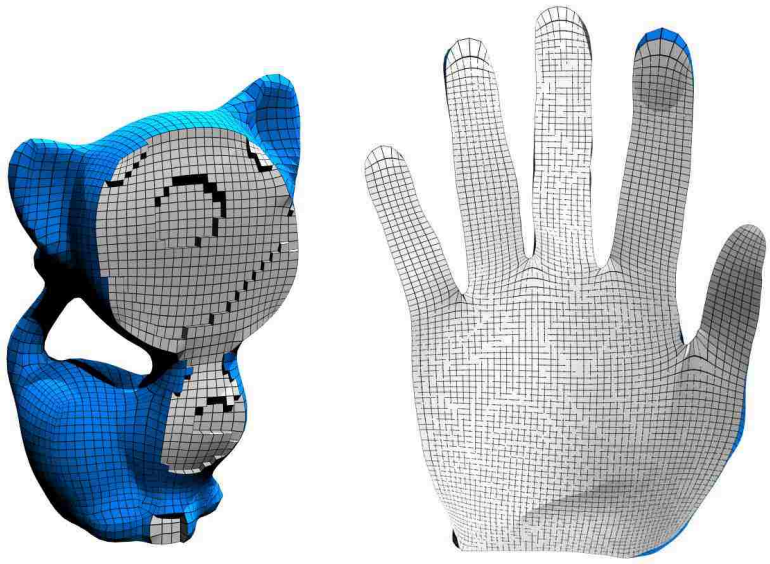


Figure 5.1. Hex Meshing Results Using Constructed Polycube Domain for Rocker Arm [4].



(a) Bump Torus (b) 3-Torus

Figure 5.2. Hex Meshing Results Using Constructed Polycube Domains for 3-Torus and Bump Torus [4].



(a) Bunny (b) Hand

Figure 5.3. Hex Meshing Results Using Constructed Polycube Domains for Model Bunny and Hand [4].

5.3 Summary

In this section, we present an effective volumetric polycube parameterization guided by frame field for given solids. This polycube parameterization usually does not have interior singularities and hence is desirable for many computer-aided design/engineering tasks such as spline construction. We demonstrate this parameterization's application in high-quality hexahedral mesh generation for 3D solid geometric models.

6 Frame Field Optimization Using Quaternions

In chapter 5, we have demonstrated the volumetric parameterization guided by a predefined frame field. The smoothness of the frame field significantly affects the quality of the parameterization and resultant hexahedral mesh generation. In this chapter, we study the automatic construction and optimization of a 3D frame field for volumetric parameterization. Besides re-meshing, the frame field is also utilized in many applications in computer graphics and geometric modeling, such as texture synthesis, non-photorealistic rendering, pen-and-ink sketching, fluid simulation, parametrization and spline construction.

We recap the definition of 3D frame field: a frame is defined as a collection of three coupled unit orthogonal vectors $\mathbf{X} = (\mathbf{v}^0, \mathbf{v}^1, \mathbf{v}^2)$, which is usually represented by a 3×3 unitary matrix. A frame field contains a set of frames $\{X_i\}$ defined over a manifold. For example, given a tetrahedral mesh M, a frame is assigned to each tetrahedron. The volumetric frame field is a 3D extension of the 2D frame field on surfaces.

The quality of a parametrization and resulting quadrilateral/hexahedral mesh depends heavily on the guiding frame field, which describes the trend of the parametric lines. The directions of the parametric lines are related to the distortion of the meshing element from a square/cube and encode the topological structure of the quadrilateral/hexahedral mesh.

Hence a good frame field is essential. Manually setting the frame field is tedious and error-prone. Therefore automatic generation of the frame field is necessary. Compared with volumetric frame field, the manipulation of the surface frame field has been studied more

thoroughly [21, 62, 64, 24, 25, 26, 57, 60, 23, 22]. The surface frame field optimization or smoothing utilizes the property that a frame on a triangle in the surface mesh can be represented by an angle, which simplifies the formulation and solution of the optimization problem.

However, the generalization from surface frame field to volumetric frame field is not straightforward. There is not enough work on volumetric frame field smoothing [20, 60] compared with the surface frame field. The representation of 3D frame field is complicated, and the singularity structure is quite different from the one in surface frame field, which makes optimization of the frame field more difficult.

In this chapter, we propose an effective frame field optimization framework based on quaternion representation which is very efficient. Our contribution is that we introduce a compact representation of the frame field based on quaternions, which accelerates the optimization. Compared with the previous work on volumetric frame field smoothing [60, 65], our representation yields fewer number of variables in the optimization solving process, which improves the efficiency of the optimization.

6.1 Frame Field Construction and Optimization

6.1.1 Definitions

Formally, a frame is defined as a collection of three ordered unit orthogonal vectors $\mathbf{X} = (\mathbf{v}^0, \mathbf{v}^1, \mathbf{v}^2)$. Therefore a frame in 3D can be represented by a 3×3 unitary matrix. A frame field is a set of frames, each of which is defined in a tetrahedron. For any tetrahedron t_i, there is a frame X_i associated with t_i. Considering a reference frame, the frame defined on a tetrahedron t_i is represented by a relative rotation matrix indicating the rotation from the reference. Without loss of generality, the reference frame is simply selected as the identity matrix.

The quaternion is a number system extending the complex number. A quaternion is a four dimensional complex number defined as $q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$, indicating a rotation about the axis $\overrightarrow{\mathbf{d}} = (d^0, d^1, d^2)$ for angle α with respect to the right hand rule. Hence there are two equivalent notations for a quaternion in Eq 6.1 and 6.2.

$$q = (w, x, y, z) \tag{6.1}$$

$$q = (\cos\frac{\alpha}{2}, \overrightarrow{\mathbf{d}} \cdot \sin\frac{\alpha}{2}) \tag{6.2}$$

Here $w = \cos\frac{\alpha}{2}$, $x = d^0 \sin\frac{\alpha}{2}$, $y = d^1 \sin\frac{\alpha}{2}$ and $z = d^2 \sin\frac{\alpha}{2}$ where $\alpha \in [0, \pi]$.

Important Rules of Quaternion Calculations. Given quaternions $q = (w, x, y, z)$, $q_i = (w_i, x_i, y_i, z_i)$ and $q_j = (w_j, x_j, y_j, z_j)$, some important rules of quaternion calculations are given as follows.

- Conjugate: $\bar{q} = (w, -x, -y, -z)$.
- Norm: $\|q\| = \sqrt{w^2 + x^2 + y^2 + z^2}$
- Reciprocal: $q^{-1} = \frac{\bar{q}}{\|q\|^2}$.
- Product: $q_i \cdot q_j = (w_i w_j - x_i x_j - y_i y_j - z_i z_j,\ w_i x_j + x_i w_j + y_i z_j - z_i y_j,\ w_i y_j - x_i z_j + y_i w_j + z_i x_j,\ w_i z_j + x_i y_j - y_i x_j + z_i w_j)$.

Conversion between Quaternion and Rotation Matrix in 3D. A quaternion $q = (w, x, y, z)$ and a 3×3 rotation matrix $A = [a_{ij}]$ can be converted to each other. For example, a quaternion q can be converted to a matrix A by

$$A = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy + 2wz & 2xz - 2wy \\ 2xy - 2wz & 1 - 2x^2 - 2z^2 & 2yz + 2wx \\ 2xz + 2wy & 2yz - 2wx & 1 - 2x^2 - 2y^2 \end{bmatrix}.$$

Converting a rotation matrix to a quaternion is a bit more challenging and requires some tricks to avoid numerical instability.

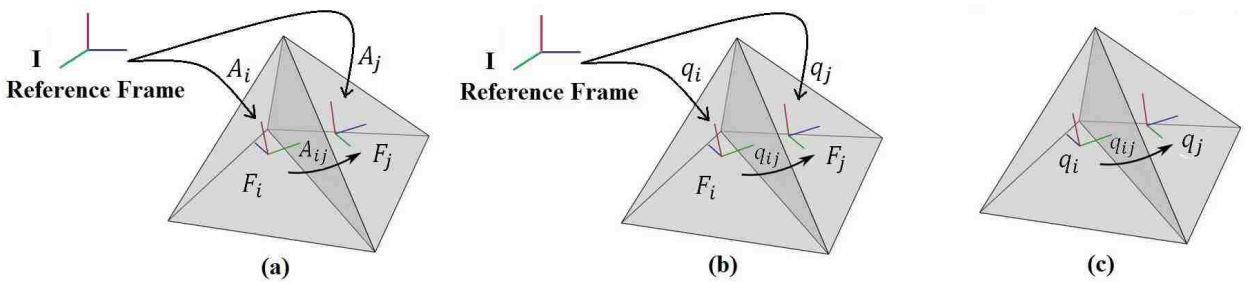


Figure 6.1. Quaternion Representation of Two Frames. (a) A frame F_i can be denoted by $F_i = A_i \cdot I$, given I is a identity reference frame and A_i is a rotation. Note that translation is ignored here. (b) Quaternions q_i, q_j and q_{ij} are equivalent to rotations A_i, A_j, and A_{ij}, respectively. (c) Frames and rotations are represented by quaternions.

6.1.2 Quaternion Representation of Frames

Given a frames F_i defined on a tetrahedra as shown in Figure 6.1-a, it can be denoted by

$$F_i = A_i \cdot I \tag{6.3}$$

where A_i is a rotation from a global reference frame I. Note that the translation is ignored here. Furthermore, the rotation from frame F_i to F_i can be denoted by

$$A_{ij} = F_j \cdot F_i^{-1} \tag{6.4}$$

If I is chosen as identity transformation, a frame F_i is equivalent to A_i according to Eq 6.3. Moreover, rotations A_i, A_j and A_{ij} can be naturally represented by quaternions q_i, q_j and q_{ij}, as shown in Figure 6.1-b. Therefore, a frame can be represented by a quaternion equivalently. If symmetry is not considered(defined in Sec 6.1.3), the transitional rotation between two frames q_i and q_j is measured by another quaternion q_{ij} in Eq 6.5, illustrated in Figure 6.1-c.

$$q_{ij} = q_j \cdot q_i^{-1} \tag{6.5}$$

6.1.3 Rotational Symmetry in Measuring the Smoothness of Two Frames

The formulation in Eq 6.5 is evaluating the magnitude of rotation from three vectors in one frame to the corresponding vectors in an adjacent frame. That evaluation method is for a simple frame field. A wide class of applications in computer graphics, such as texture

synthesis [96] or re-meshing [97], requires to use objects of higher symmetry than simple frame field, i.e. objects invariant by rotations of $90 \cdot k$ degrees about single $u-, v-$ or $w-$axis where $k \in \mathbb{Z}$ [24]. In that case, one vector, e.g. u-axis in a frame can correspond to a different vector, e.g. $v-$axis in an adjacent frame. This concept is called *rotational symmetry* in the frame field and can be used to introduce interior singularities in the parameterization and reduce the distortion [24]. Specifically, in a "perfectly smooth" rotational symmetric frame field (there is no deviation),the transformations rotating a frame to another one without distinguishing the specific types of vectors contain 24 rotations, namely identity, rotation about single $u-, v-$ and $w-$axis for $90, 180$ and 270 degrees, and the composite of the aforementioned rotations. The aforementioned 24 rotations form a group called chiral cubical symmetry group , denoted by $\mathcal{G}$ in this thesis.

Figure 6.2 illustrates the local effects of allowing rotational symmetry in parameterization and subsequent mesh generation. In these three examples, the parametric lines are connected in different ways. Without rotational symmetry, a parametric line (e.g. iso-u) in a local chart connects to another local chart's parametric line of the same type as shown in Figure 6.2-a. After the parameterization, hexes could be extracted by intersecting the traced parametric lines. Then a hex-mesh is generated by gluing generated adjacent hexes. If rotational symmetry is allowed, in the generated hex-mesh, different parametric lines are not distinguished. For example, as shown in Figure 6.2-b positive iso-u line can seamlessly connect to negative iso-u line. Figure 6.2-c shows another example, iso-u line is connected with iso-v line. When using these parameterization to generate hexahedral meshes, all these three meshing results are valid.

Once the symmetry is considered, the smoothness of two frames q_0 and q_1 should be the minimum rotation angle, after applying any rotation in $\mathcal{G}$ on the transitional rotation q_{01}. For example, the energy of 120-degree rotation should be equivalent to the energy of 30-degree

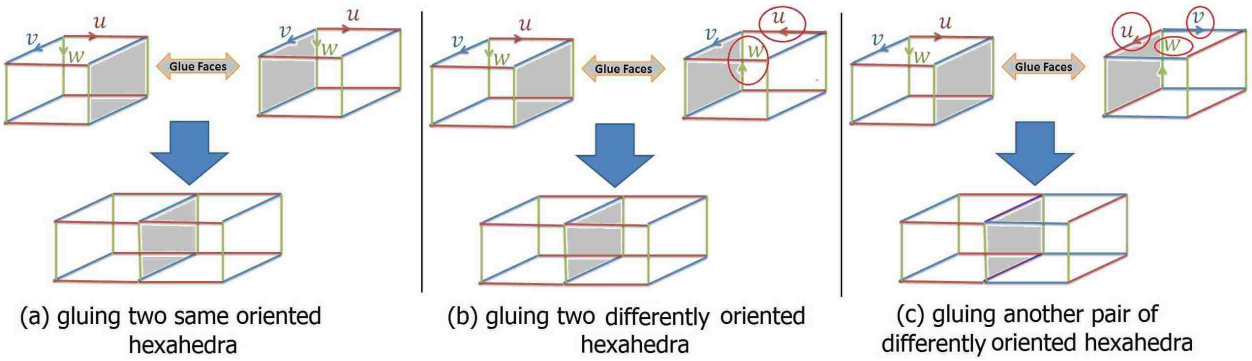


Figure 6.2. Different Types of Parametric Lines Can Connect to Each Other in Generated Hexahedral Mesh. (a) Normally a parametric line (e.g. iso-u) in a hex connects to another one's parametric line of the same type. Hex-mesh is generated by gluing adjacent hexes. In the generated hex-mesh, different parametric lines are not distinguished. For example, in (b) iso-u line in positive direction can seamlessly connect to iso-u line in negative direction. In (c) iso-u line can connect to iso-v line. The glued hexes in (a), (b) and (c) are considered the same.

rotation. Formally, the smoothness is defined as

$$E_{smooth}(q_0, q_1) = \min_{\pi_{01}} \quad \theta_{01} \tag{6.6}$$

where θ_{01} is the rotation angle in the quaternion $s = \pi_{01} \cdot q_{01}$ and $\pi_{01} \in \mathcal{G}$ is represented by a quaternion. According to Eq 6.2, $s = (\cos\frac{\theta}{2}, \overrightarrow{\mathbf{d}} \cdot \sin\frac{\theta}{2})$. Here θ is the actual rotational angle between the two frames.

In surface parameterization, rotational symmetry was studied in designing smooth surface frame field [24, 23]. It is non-trivial to generalize such symmetry in volumetric frame field smoothing, since representing the symmetry continuously in optimization is very challenging. First, previous volumetric frame field smoothing methods [60, 5] use softly constrained energy which represents the symmetry through an indirect way, where the symmetry is denoted by matrices. Hence orthonormalization is necessary to round the approximated solution to a valid rotation matrix. Second, more importantly, unless manually constructed or adjusted, frame fields do not usually lead to non-degenerate volumetric parameterization. Finally, in contrast to the surface frame field, symmetry for 3D frame fields cannot be formulated by simple one-parameter 2D rotations in the tangent planes [60].

Quaternion is a natural and direct way to indicate the symmetry in rotation. One of the benefits of using this representation is that quaternions are always valid rotations, where complicated orthonormalization is not required. Normalization of quaternions is usually required, which is very simple.

Moreover, since $\mathcal{G}$ contains 24 equivalent classes of rotations, the condition $\pi_{01} \in \mathcal{G}$ requires rounding rotation π_{01} to one of the 24 rotations. If π_{01} were represented by a matrix, the geometric meaning of the rounding process is obscure. However, if π_{01} is represented by a quaternion, the geometry meaning is intuitive: rounding π_{01} to a rotation in $\mathcal{G}$ because their rotation angles and axes are close to each other respectively. This rounding is necessary since it is very difficult to solve the problem if π_{01} changes discretely.

6.1.4 Definition of Objective Energy

Given two frames defined on adjacent tetrahedra t_i and t_j, denoted by q_i and q_j, the rotation from q_i to q_j is denoted by $q_{ij} = q_j \cdot q_i^{-1}$. When symmetry is considered, the rotation between them is denoted by $s_{ij} = (s_{ij}^0, s_{ij}^1, s_{ij}^2, s_{ij}^3)$ as defined in Eq 6.7.

$$s_{ij} = \pi_{ij} \cdot q_{ij} \tag{6.7}$$

According to the definition of quaternions in Eq 6.2, the first component of s_{ij} is related to the rotation angle θ_{ij}, formally in Eq 6.8.

$$s_{ij}^0 = \cos \frac{\theta_{ij}}{2} \tag{6.8}$$

where generally $\theta_{ij} \in [0, \pi]$, or $\theta_{ij} \in [0, \frac{\pi}{2}]$ considering the symmetry. A straightforward way to measure the smoothness is using the rotation angles as shown in Eq 6.9.

$$E_{simple}(f_{ij}) = \theta_{ij} = 2 \arccos s_{ij}^0 \tag{6.9}$$

where t_j and t_j share a face f_{ij}. This involves inverse trigonometric function $\arccos x$, which is non-linear and not differentiable at $x = k\pi, k \in \mathbf{Z}$.

To get rid of such arccos function, an approximated objective energy can be defined using $\cos\frac{\theta_{ij}}{2}$, as shown in Eq 6.10.

$$E(f_{ij}) = -(s_{ij}^0)^2 = -\cos^2\frac{\theta_{ij}}{2} \tag{6.10}$$

This formulation is desirable approximation and simplification because $y = \cos\frac{x}{2}$ function is **monotonically decreasing** in the interval $x \in [0, \pi]$. Consequently, E and E_{simple} achieve maxima and minima at the same time, i.e. when $\theta_{ij} = 0$ (minimum) or $\theta_{ij} = \frac{\pi}{2}$ (maximum). According to the properties of quaternions, s_{ij}^0 is related to q_i, q_j defined on the two adjacent tetrahedra, as shown in Eq 6.11.

$$\begin{aligned} s_{ij}^0 = \quad & m_{ij}^0(w_iw_j - x_ix_j - y_iy_j - z_iz_j) \\ & -m_{ij}^1(w_ix_j + x_iw_j + y_iz_j - z_iy_j) \\ & -m_{ij}^2(w_iy_j - x_iz_j + y_iw_j + z_ix_j) \\ & -m_{ij}^3(w_iz_j + x_iy_j - y_ix_j + z_iw_j) \end{aligned} \tag{6.11}$$

where $\pi_{ij} = (m_{ij}^0, m_{ij}^1, m_{ij}^2, m_{ij}^3)$, $q_i = (w_i, x_i, y_i, z_i)$, and $q_j = (w_j, x_j, y_j, z_j)$ will be the variables in later optimization. The overall objective energy can be formulated as the summation of $E(f_{ij})$ over all the interior faces in a volumetric mesh, as shown in Eq 6.12.

$$E = \sum_{f_{ij}} E(f_{ij}) \tag{6.12}$$

where f_{ij} are shared by two adjacent tetrahedra t_i and t_j.

This objective energy is to be minimized subject to the spherical constraints, i.e. the quaternions, such as q_i, q_j and π_{ij}, should be normalized, as follows.

$$\begin{aligned} w_i^2 + x_i^2 + y_i^2 + z_i^2 &= 1 \\ w_j^2 + x_j^2 + y_j^2 + z_j^2 &= 1 \\ (m_{ij}^0)^2 + (m_{ij}^1)^2 + (m_{ij}^2)^2 + (m_{ij}^3)^2 &= 1 \end{aligned}$$

6.1.5 Optimization

The problem of volumetric frame field smoothing has been formulated as a 6-degree polynomial objective function subject to 4D spherical constraint. We apply an efficient great-circle search optimization algorithm [67] which minimizes the objective energy subject to the spherical constraints. We avoid repetition in this chapter and refer the readers to [67] for details about the optimization algorithm.

6.2 Experimental Results and Discussions

The experiments are run on a laptop with 2.1GHz Intel Core i3-2310M CPU and 12GB RAM. The results are compared with a state-of-the-art method [5] in the rotation angles between two adjacent frames in the field. In order to have same boundary frame field for fair comparison on one model, we extract and adopt the boundary frames of the data provided by [5], which is generated from principal-dominant cross fields on boundary surfaces.

Numerical results and timings for various models are reported in Tables 6.1-6.6, where $\overline{\theta}$, δ, and θ_{max} indicate the mean rotation angle between two adjacent frames, the standard deviation of the rotation angles, and the maximum rotation angle, respectively. The initial guess of the interior frames are set as identity. Compared with [5], our method generates smoother frame field whose rotation angle between two adjacent frames is smaller statistically. In addition, our method is more efficient, since there are fewer number of variables and the constraints are less complicated when quaternion representation is used.

The number of singularities in the optimized frame fields is reported in Table 6.7. N_s is the number of singularities in the model. Here the number of singularities is counted as the summation of the number of simple singularity curves and the number of intersection nodes of the singularity curves.

Figure 6.3 illustrates the distribution of rotation angles between adjacent frames in the smoothed frame field generated by our methods and [5], where red curves represent our

Table 6.1. Comparison of Frame Field Smoothness and Optimization Time on Rod. $\overline{\theta}$, δ, and θ_{max} indicate the mean rotation angle between two adjacent frames, the standard deviation of the rotation angles, and the maximum rotation angle, respectively.

#Tets= 41K				
	$\overline{\theta}$	δ	θ_{max}	Time(s)
SRF [5]	59.95	57.49	180.00	7.9
Ours	11.52	21.56	102.27	3.2

Table 6.2. Comparison of Frame Field Smoothness and Optimization Time on Bunny.

#Tets= 153K				
	$\overline{\theta}$	δ	θ_{max}	Time(s)
SRF [5]	96.79	61.07	180.00	33.0
Ours	7.26	20.09	108.70	13.4

results and blue curves represent results in [5]. In the subfigures, horizontal axis indicates the rotation angle in degrees, and vertical axis indicates the accumulated percentage of rotation angles below a specific degree. Note that the closer is the curve to shape '⌜'(i.e. closer to left side and top side of the bounding box of the subfigure), the smoother is the frame field. Compared with [5], our method generates smoother frame fields. An interesting phenomenon can be noticed that the smoothing of the largest model Rocker Arm consumes surprisingly less time compared with other models. The reason for this is the objective energy moves to a local minima soon after the starting point.

Remarks. Though the Poincarë-Hopf index theorem [24] provides theoretical guarantee to find a frame field which is valid for surface parameterization, given a 3D frame field, whether it leads to a valid volumetric parameterization and the existence of resultant all-hex re-meshing is an open problem. Therefore, we focus on the smoothing of a volumetric frame field, while dealing with the singularity of a generated frame field is beyond the scope of this dissertation. There exists work trying to propose various heuristic postprocessing methods

Table 6.3. Comparison of Frame Field Smoothness and Optimization Time on Fertility.

#Tets= 179K				
	$\overline{\theta}$	δ	θ_{max}	Time(s)
SRF [5]	100.30	57.00	180.00	35.0
Ours	10.49	23.10	109.80	14.1

Table 6.4. Comparison of Frame Field Smoothness and Optimization Time on Joint.

#Tets= 187K				
	$\overline{\theta}$	δ	θ_{max}	Time(s)
SRF [5]	87.92	70.32	180.00	34.4
Ours	7.47	23.30	100.21	12.71

Table 6.5. Comparison of Frame Field Smoothness and Optimization Time on Hanger.

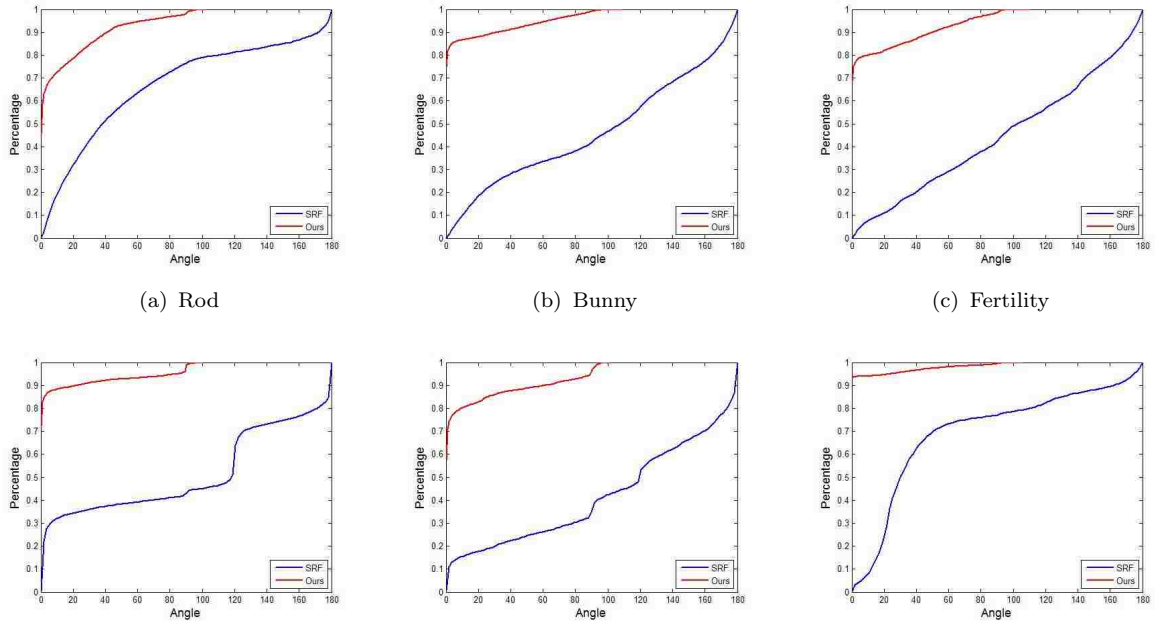
#Tets= 215K				
	$\overline{\theta}$	δ	θ_{max}	Time(s)
SRF [5]	105.73	63.11	180.00	40.7
Ours	11.75	26.32	101.13	28.7

Table 6.6. Comparison of Frame Field Smoothness and Optimization Time on Rocker Arm.

#Tets= 254K				
	$\overline{\theta}$	δ	θ_{max}	Time(s)
SRF [5]	54.97	53.70	180.00	46.9
Ours	2.97	13.24	105.88	4.2

Table 6.7. Number of Singularities in Optimized Frame Fields. N_s is the number of singularities in the model.

Model	Rod	Bunny	Fertility	Joint	Hanger	Rocker Arm
N_s	946	2563	3552	397	1250	663



(a) Rod (b) Bunny (c) Fertility

(d) Joint (e) Hanger (f) Rocker Arm

Figure 6.3. Distribution of Rotation Angles in Various Models. Red curves represents our results and blue curves represent results of SRF method [5]. Horizontal axis indicates the rotation angle in degrees. Vertical axis indicates the accumulated percentage of rotation angles below a specific value. The closer is the curve to shape 'T'(i.e. closer to left side and top side of the bounding box the subfigure), the smoother is the frame field. Our method generates smoother frame fields. As for the time complexity, please refer to Tables 6.1-6.6.

to remedy an incorrect frame field that yields degenerated parameterization, but none has guaranteed success [65, 5]. Such postprocessing techniques includes modifying the matching matrices [65] and/or improper singular edge collapse [5]. Here improper singularity is the singular edge that yields degenerated parameterization.

6.3 Summary

In this chapter, quaternions are used to represent the frames for fast and effective volumetric frame field smoothing. Compared with existing representations of the frames, the dimension of the optimization variables is reduced, and the formulation of objective energy as well as the constraints on the variables is simplified. Thanks to this compact representation, a previously proposed great-circle search optimization algorithm in chapter 3 is able to be

applied which deals with spherical constraints. The experimental results indicate that this proposed approach generates smoother frame field and is more efficient.

7 Conclusions

To summarize, in this dissertation we have studied the construction of parameterization on regular geometric domains and explored their applications in shape modeling and computer-aided design.

In chapter 3, we develop an effective progressive spherical parameterization algorithm, with an efficient nonlinear optimization scheme subject to the spherical constraint. Compared with state-of-the-art spherical mapping algorithms, our method demonstrates the advantages of great efficiency, lower distortion, and guaranteed bijectiveness, and we show its applications in spherical harmonics decomposition and shape analysis.

Spherical mapping is not suitable for high-genus surfaces but polycube domain can handle this problem well. Therefore, in chapter 4, we propose a first topology-preserving polycube domain optimization algorithm that optimizes polycube domain together with the parameterization to balance the mapping distortion and domain simplicity. We develop effective nonlinear geometric optimization algorithms dealing with variables with and without derivatives. This polycube parameterization algorithm can benefit the regular quadrilateral mesh generation and cross-surface parameterization.

Surface parameterization has been widely investigated but the research for solids is not enough. Hence in chapter 5, we develop volumetric parameterization guided by frame field and demonstrate its application on high-quality mesh generation. A limitation is that such local chart based algorithm would introduce more variables and lead to larger linear systems,

which requires a large-scale solver for sparse linear systems. We plan to extend the application of the volumetric parameterization for heterogeneous volumes [58] and large-scale models.

In chapter 6, we develop a novel quaternion-based optimization framework for 3D frame field construction and volumetric parameterization computation. We demonstrate our constructed 3D frame field has better smoothness, compared with state-of-the-art algorithms, and it is effective in guiding low-distortion volumetric parameterization and high-quality hexahedral mesh generation.

The limitations of the studied algorithms in this dissertation include: (1) Obtaining a lowly distorted mapping for models with the long branch regions is still difficult. Besides, for an extension of this work, we would like to exploit the parallelism in the global optimization scheme by decomposing the given surface into a few individual regions [98, 99, 94] on which local optimization of different vertices can be executed simultaneously without interrupting each other. (2) Feature alignment in the polycube mapping can benefit many graphics applications such as morphing and registration. However, this is challenging and has not been well discussed/solved in existing polycube mapping literature. Within our current framework, on a subpatch, directly enforcing the harmonic mapping to map an interior feature point to a specific position on the polycube domain may cause local flip-over around the feature point. One possible approach is to simply add feature alignment as a soft constraint in the mapping optimization step, such that feature matching errors are penalized like the angle-distortion and area-distortion terms. To enforce a hard constraint on feature matching, additional domain partitioning [94] to make the features on the subpatch boundary can be another solution. Surface decomposition has been widely studied (see surveys [100, 101]). (3) The algorithm in chapter 4 can not change the topology of the polycube domain. A possible polycube construction with topological operations is studied in [4]. (4) Though the Poincarë-Hopf index theorem [24] provides theoretical guarantee to find a frame field which is valid for surface parameterization, whether a given 3D frame field can lead to valid volumetric

parameterization and resultant all-hex re-meshing is an open problem. It seems plausible that we could start from a trivial but valid frame field such as identities, and constrain the searching of frame field within a subspace that would not violate the sufficient conditions discussed in [4, 65] such as only allowing identities and rotations about single axis and avoiding two different types of matching matrices in a triangle face. We will study this problem in the near future.

References

[1] Rhaleb Zayer, Christian Rossl, and Hans-Peter Seidel. Curvilinear spherical parameterization. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006*, Washington, DC, USA, 2006. IEEE Computer Society.

[2] Emil Praun and Hugues Hoppe. Spherical parametrization and remeshing. *ACM Trans. Graph.*, 22:340–349, July 2003.

[3] Xianfeng Gu and Shing-Tung Yau. Global conformal surface parameterization. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 127–137, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[4] Wuyi Yu, Kang Zhang, Shenghua Wan, and Xin Li. Optimizing polycube domain construction for hexahedral remeshing. *Computer-Aided Design*, in press, 2013.

[5] Yufei Li, Yang Liu, Weiwei Xu, Wenping Wang, and Baining Guo. All-hex meshing using singularity-restricted field. *ACM Trans. Graph.*, 31(6):177:1–177:11, November 2012.

[6] M. Tarini, K. Hormann, P. Cignoni, and C. Montani. Polycube-maps. In *ACM SIGGRAPH*, pages 853–860, 2004.

[7] Hongwei Li, Kui-Yip Lo, Man-Kang Leung, and Chi-Wing Fu. Dual poisson-disk tiling: An efficient method for distributing features on arbitrary surfaces. *IEEE TVCG*, 14(5):982–998, 2008.

[8] Zhengwen Fan, Xiaogang Jin, Jieqing Feng, and Hanqiu Sun. Mesh morphing using polycube-based cross-parameterization: Animating geometrical models. *Comput. Animat. Virtual Worlds*, 16(3-4):499–508, 2005.

[9] Hongyu Wang, Ying He, Xin Li, Xianfeng Gu, and Hong Qin. Polycube splines. In *SPM*, pages 241–251, New York, NY, USA, 2007. ACM.

[10] Hongyu Wang, Miao Jin, Ying He, Xianfeng Gu, and Hong Qin. User-controllable polycube map for manifold spline construction. In *SPM*, pages 397–404, New York, NY, USA, 2008. ACM.

[11] Xin Li, Xiaohu Guo, Hongyu Wang, Ying He, Xianfeng Gu, and Hong Qin. Harmonic volumetric mapping for solid modeling applications. In *SPM*, pages 109–120, New York, NY, USA, 2007. ACM.

[12] Xin Li, Huanhuan Xu, Shenghua Wan, Zhao Yin, and Wuyi Yu. Feature-aligned harmonic volumetric mapping using mfs. *Computers & Graphics*, 34(3):242 – 251, 2010.

[13] Yalin Wang, Xianfeng Gu, Tony F. Chan, Paul M. Thompson, and Shing tung Yau. Volumetric harmonic brain mapping. In *ISBI 04: IEEE International Symposium on Biomedical Imaging: Macro to Nano*, pages 1275–1278, 2004.

[14] X. Li, X. Guo, H. Wang, Y. He, X. Gu, and H. Qin. Meshless harmonic volumetric mapping using fundamental solution methods. *IEEE Trans. on Automation Science and Engineering*, 6, 2009.

[15] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.*, 24(3):561–566, July 2005.

[16] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. *ACM Trans. Graph.*, 26(3), July 2007.

[17] Yaron Lipman, David Levin, and Daniel Cohen-Or. Green coordinates. *ACM Trans. Graph.*, 27(3):78:1–78:10, August 2008.

[18] Nicu D. Cornea, Deborah Silver, and Patrick Min. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, May 2007.

[19] T. Martin, E. Cohen, and R. M. Kirby. Volumetric parameterization and trivariate b-spline fitting using harmonic functions. *Computer Aided Geometry Design*, 26(6):648–664, August 2009.

[20] M. Nieser, U. Reitebuch, and K. Polthier. Cubecover parameterization of 3d volumes. *Computer Graphics Forum*, 30(5), 2011.

[21] Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. Periodic global parameterization. *ACM Trans. Graph.*, 25(4):1460–1485, 2006.

[22] Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Vector field design on surfaces. *ACM Trans. Graph.*, 25(4):1294–1326, October 2006.

[23] Jonathan Palacios and Eugene Zhang. Rotational symmetry field design on surfaces. *ACM Trans. Graph.*, 26(3), July 2007.

[24] Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. N-symmetry direction field design. *ACM Trans. Graph.*, 27(2):10:1–10:13, 2008.

[25] Nicolas Ray, Bruno Vallet, Laurent Alonso, and Bruno Levy. Geometry-aware direction field processing. *ACM Trans. Graph.*, 29(1):1:1–1:11, 2009.

[26] David Bommes, Henrik Zimmer, and Leif Kobbelt. Mixed-integer quadrangulation. *ACM Trans. Graph.*, 28(3):77:1–77:10, July 2009.

[27] Xin Li, Zhao Yin, Li Wei, Shenghua Wan, Wei Yu, and Maoqing Li. Symmetry and template guided completion of damaged skulls. *Computers and Graphics*, 35(4):885–893, 2011.

[28] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. *Advances in Multiresolution for Geometric Modelling*, pages 157–186, 2005.

[29] A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.*, 2(2):105–171, 2006.

[30] K. Hormann, B. Lévy, and A. Sheffer. Mesh parameterization: Theory and practice. In *ACM Siggraph 2007 Course*, volume 11, pages 1–87, 2007.

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

[32] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH*, pages 173–182, 1995.

[33] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21(3):209–218, 2002.

[34] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.

[35] Steven Haker, Sigurd Angenent, Allen Tannenbaum, Ron Kikinis, Guillermo Sapiro, and Michael Halle. Conformal surface parameterization for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 6:181–189, April 2000.

[36] Zhong Li, Yao Jin, Xiaogang Jin, and Lizhuang Ma. Approximate straightest path computation and its application in parameterization. *The Visual Computer*, 28:63–74, 2012. 10.1007/s00371-011-0600-0.

[37] Craig Gotsman, Xianfeng Gu, and Alla Sheffer. Fundamentals of spherical parameterization for 3d meshes. *ACM Trans. Graph.*, 22:358–363, July 2003.

[38] Shadi Saba, Irad Yavneh, Craig Gotsman, and Alla Sheffer. Practical spherical embedding of manifold triangle meshes. In *Proceedings of the International Conference on Shape Modeling and Applications 2005*, pages 258–267, Washington, DC, USA, 2005. IEEE Computer Society.

[39] A. Sheffer, C. Gotsman, and N. Dyn. Robust spherical parameterization of triangular meshes. *Computing*, 72:185–193, April 2004.

[40] Arul Asirvatham and Emil Praun. Consistent spherical parameterization. In *in: International Conference on Computational Science*, pages 265–272, 2005.

[41] Haishan Tian, Yuanjun He, and Yong Wu. A new approach of progressive spherical parameterization. In *9th International Conference on Computer Aided Design and Computer Graphics*, page 5 pp., dec. 2005.

[42] Ying Li, Zhouwang Yang, and Jiansong Deng. Spherical parameterization of genus-zero meshes using the lagrange-newton method. In *10th International Conference on Computer Aided Design and Computer Graphics*, page 32, 2007.

[43] Ilja Friedel, Peter Schröder, and Mathieu Desbrun. Unconstrained spherical parameterization. In *ACM SIGGRAPH 2005 Sketches*, New York, NY, USA, 2005. ACM.

[44] R.B.Schudy and D.Ballard. Towards an anatomical model of heart motion as seen in 4d cardiac ultrasound data. In *in 6 th Conf on Computer Applications in radiology ans Computer Aided Analysis of Radiological Images*, pages 366–376, 1979.

[45] G. B. Arfken and H. J. Weber. *Mathematical Methods for Physicists*. Academic Press, Orlando, 1985.

[46] Kun Zhou, Hujun Bao, and Jiaoying Shi. 3d surface filtering using spherical harmonics. *Computer-Aided Design*, 36(4):363–375, February 2004.

[47] M.Mousa, R.Chaine, and S.Akkouche. Frequency-based representation of 3d point-based surfaces using the spherical harmonics. *ICCVG'06, International Conference on Computer Vision and Graphics*, September 2006.

[48] Li Shen and Moo K. Chung. Large-scale modeling of parametric surfaces using spherical harmonics. *IEEE 3DPVT 2006: Third International Symposium on 3D Data Processing, Visualization and Transmission*, pages 294–301, June 2006.

[49] Chengming Zou, Guanghui Zhao, and Edwin R.Hancock. Reconstructing 3d facial shape using spherical harmonics. In *ICIAP'09 Proceedings of the 15th International Conference on Image Analysis and Processing*, pages 949–957, 2009.

[50] M.Kazhdan, T.Funkhouser, and S.Rusinkiewicz. Rotation invariant spherical harmonic represetation of 3d shape descriptors. *Symposium on Geometry Processing*, pages 167–175, June 2003.

[51] Panagiotis Papadakis, Ioannis Pratikakis, Stavros Perantonis, and Theoharis Theoharis. Efficient 3d shape matching and retrieval using a concrete radialized spherical projection representation. *Pattern Recognition*, 40(9):2437–2452, 2007.

[52] J. Lin, X. Jin, Z. Fan, and C. C. L. Wang. Automatic polycube-maps. In *GMP*, pages 3–16, 2008.

[53] 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.

[54] Shuchu Han, Jiazhi Xia, and Ying He. Hexahedral shell mesh construction via volumetric polycube map. In *SPM*, SPM '10, pages 127–136, New York, NY, USA, 2010. ACM.

[55] Shenghua Wan, Zhao Yin, Kang Zhang, Hongchao Zhang, and Xin Li. A topology-preserving optimization algorithm for polycube mapping. *Computer & Graphics*, 35(3):639–649, June 2011.

[56] Jiazhil Xia, Ismael Garcia, Ying He, Shi-Qing Xin, and Gustavo Patow. Editable polycube map for gpu-based subdivision surfaces. In *Symposium on Interactive 3D Graphics and Games*, pages 151–158, 2011.

[57] James Gregson, Alla Sheffer, and Eugene Zhang. All-hex mesh generation via volumetric polycube deformation. *Computer Graphics Forum*, 30(5):1407–1416, 2011.

[58] Huanhuan Xu, Wuyi Yu, Shiyuan Gu, and Xin Li. Biharmonic volumetric mapping using fundamental solutions. *IEEE Transactions on Visualization and Computer Graphics*, 19(5):787–798, 2013.

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

[60] Jin Huang, Yiying Tong, Hongyu Wei, and Hujun Bao. Boundary aligned smooth 3d cross-frame field. *ACM Trans. Graph.*, 30(6):143:1–143:8, 2011.

[61] Kexiang Wang, Xin Li, Bo Li, Huanhuan Xu, and Hong Qin. Restricted trivariate polycube splines for volumetric data modeling. *IEEE Transactions on Visualization and Computer Graphics*, 18(5):703–716, 2012.

[62] Felix Kälberer, Matthias Nieser, and Konrad Polthier. Quadcover - surface parameterization using branched coverings. *Computer Graphics Forum*, 26(3):375–384, 2007.

[63] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 517–526, New York, NY, USA, 2000.

[64] Wan-Chiu Li, Bruno Vallet, Nicolas Ray, and Bruno Levy. Representing higher-order singularities in vector fields on piecewise linear surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1315–1322, 2006.

[65] Jin Huang, Tengfei Jiang, Yuanzhen Wang, Yiying Tong, and Hujun Bao. Automatic frame field guided hexahedral mesh generation. Technical Report MSU-CSE-12-9, Department of Computer Science, Michigan State University, East Lansing, Michigan, August 2012.

[66] Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 409–416, New York, NY, USA, 2001. ACM.

[67] Shenghua Wan, Tengfei Ye, Maoqing Li, Hongchao Zhang, and Xin Li. An efficient spherical mapping algorithm and its application on spherical harmonics. *SCIENCE CHINA Information Sciences, Special Issue of Computational Visual Media Conference 2012.*, in press, 2013.

[68] K. Hormann and G. Greiner. Mips: An efficient global parametrization method. *Curve and Surface Design: Saint-Malo 1999*, pages 153–162, 2000.

[69] Hugues Hoppe. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 99–108, New York, NY, USA, 1996. ACM.

[70] J. Nocedal and S. J. Wright. *Numerical Optimization*, 2006.

[71] D. T. Lee and F. P. Preparata. An optimal algorithm for finding the kernel of a polygon. *J. ACM*, 26(3):415–421, July 1979.

[72] Gnter Rote. Computing the minimum hausdorff distance between two point sets on a line under translation. *Information Processing Letters*, 38:123–127, 1991.

[73] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23:399–405, August 2004.

[74] Martin J.Mohlenkamp. A user's guide to spherical harmonics. `http://www.ohio.edu/people/mohlenka/research/uguide.pdf`, 2011.

[75] X. Guo, X. Li, Y. Bao, X. Gu, and H. Qin. Meshless thin-shell simulation based on global conformal parameterization. *IEEE TVCG*, 12(3):375–385, 2006.

[76] X. Gu and S.-T. Yau. Global conformal surface parameterization. In *Proc. Symp. Geometry Processing*, pages 127–137, 2003.

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

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

[79] E. Praun, W. Sweldens, and P. Schröder. Consistent mesh parameterizations. In *ACM SIGGRAPH*, pages 179–184, 2001.

[80] Jonathan Barzilai and Jonathan M. Borwein. Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[81] Ernesto G. Birgin, Mario Mart Jos, and Marcos Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, pages 1196–1211, 2000.

[82] Yu-Hong Dai, William W. Hager, Klaus Schittkowski, and Hongchao Zhang. The cyclic barzilai-borwein method for unconstrained optimization. *IMA Journal of Numerical Analysis*, 26:604–627, 2006.

[83] Yu-Hong Dai and Hongchao Zhang. Adaptive two-point stepsize gradient algorithm. *Numerical Algorithms*, 27:377–385, 2001.

[84] William W. Hager and Hongchao Zhang. A new active set algorithm for box constrained optimization. *SIAM J. on Optimization*, 17:526–557, August 2006.

[85] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. Spectral surface quadrangulation. *ACM Trans. Graph.*, 25:1057–1066, July 2006.

[86] V. Kraevoy and A. Sheffer. Cross-parameterization and compatible remeshing of 3D models. *ACM Trans. Graph.*, 23(3):861–869, 2004.

[87] Timothy A. Davis and William W. Hager. Dynamic supernodes in sparse cholesky update/downdate and triangular solves. *ACM Trans. Math. Softw.*, 35:27:1–27:23, 2009.

[88] Kai Xu, Hao Zhang, Daniel Cohen-Or, and Yueshan Xiong. Technical section: Dynamic harmonic fields for surface processing. *Comput. Graph.*, 33:391–398, June 2009.

[89] H. Zhang, A. R. Conn, and K. Scheinberg. A derivative-free algorithm for the least-squares minimization. *SIAM Journal on Optimization.*, 20:3555–3576, 2010.

[90] M. J. D. Powell. Least frobenius norm updating of quadratic models that satisfy interpolation conditions. *Math. Program.*, 100:183–215, May 2004.

[91] Jorge J. Moré and Stefan M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM J. Optimization*, 20(1):172–191, 2009.

[92] Jorge Moré. The levenberg-marquardt algorithm: Implementation and theory. In G. Watson, editor, *Numerical Analysis*, volume 630 of *Lecture Notes in Mathematics*, pages 105–116. Springer Berlin / Heidelberg, 1978.

[93] Charles Audet and J. E. Dennis, Jr. A pattern search filter method for nonlinear programming without derivatives. *SIAM J. on Optimization*, 14:980–1010, 2004.

[94] X. Li, X. Gu, and H. Qin. Surface mapping using consistent pants decomposition. *IEEE TVCG*, 15(4):558–571, 2009.

[95] P. Degener, J. Meseth, and R. Klein. An adaptable surface parameterization method. In *In IMR*, pages 201–213, 2003.

[96] Greg Turk. Texture synthesis on surfaces. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 347–354, New York, NY, USA, 2001. ACM.

[97] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic polygonal remeshing. *ACM Trans. Graph.*, 22(3):485–493, July 2003.

[98] W. Yu and X. Li. Computing 3d shape guarding and star decomposition. *Computer Graphics Forum*, 30:2087–2096, 2011.

[99] Qi-Xing Huang, Martin Wicke, Bart Adams, and Leonidas J. Guibas. Shape decomposition using modal analysis. *Computer Graphics Forum*, 28:407–416, 2009.

[100] Alexander Agathos, Ioannis Pratikakis, Stavros Perantonis, Nikolaos Sapidis, and Philip Azariadis. 3D mesh segmentation methodologies for CAD applications. *Computer-Aided Design & Applications*, 4(6):827–841, 2007.

[101] Ariel Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27:1539–1556, 2008.

Vita

Shenghua Wan was born in the city of Nanchang of Jiangxi Province in China in 1987. He graduated from Harbin Institute of Technology, Heilongjiang Province, China in July 2009 with the bachelor's degree in Computer Science with the award of excellent thesis. One month later, He joined the doctoral program in the department of Electrical and Computer Engineering in Louisiana State University, Baton Rouge, USA. During the pursuit of PhD degree, he obtained a master's degree in Electrical Engineering from the School of Electrical Engineering and Computer Science in Louisiana State University in 2011.