# Stony Brook University

# A Spline-Based Data Modeling Framework Over Regular Domains

A Dissertation Presented
by
**Hongyu Wang**

to
The Graduate School
in Partial Fulfillment of the
Requirements
for the Degree of
**Doctor of Philosophy**
in
**Computer Science**
Stony Brook University

**December 2009**

**Stony Brook University**
The Graduate School

**Hongyu Wang**

We, the dissertation committee for the above candidate for
the degree of Doctor of Philosophy, hereby recommend
acceptance of this dissertation.

**Hong Qin, Dissertation Advisor**
Professor, Computer Science Department

**Xianfeng Gu**
Associate Professor, Computer Science Department

**Xiangmin Jiao**
Assistant Professor, Department of Applied Mathematics and Statistics

**Joseph Mitchell, Chairperson of Defense**
Professor, Department of Applied Mathematics and Statistics

**Ye Duan**
Associate Professor, Computer Science Department
University of Missouri at Columbia

This dissertation is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

**A Spline-Based Data Modeling Framework Over Regular Domains**

by

**Hongyu Wang**

**Doctor of Philosophy**

in

**Computer Science**

Stony Brook University

**2009**

With the rapid advancement of modern 3D scanning technologies, CAD-based digital prototypes are routinely acquired in forms of raw points and/or triangular meshes. In order to enable geometric design and downstream product development processes (e.g., accurate shape analysis, finite element simulation, and e-manufacturing) in CAE environments, discrete data inputs must be converted into continuous, compact representations for scientific computing and engineering applications. In this dissertation, we present a novel spline-based data modeling framework to directly define tensor-product splines over any manifolds (serving as parametric domains). Since tensor-product B-splines and NURBS are current standards in CAD software industry, our entire mesh-to-spline data transformation pipeline enables and expedites the manifold surface design over existing CAD software platform industry (without any trimming), and thus, has great potential in shape modeling and reverse engineering applications of complicated real-world objects.

Tensor-product spline schemes require the parametric domains have the regular (rectangular) structures, and constructing the domain manifold with regular structures in an efficient way still remains a challenge. In this dissertation, we study and present efficient regular domain construction methods, and demonstrate their applications in modeling 3D objects of arbitrary topology.

First, we propose the novel concept of polycube splines by defining splines directly upon the polycube map, serving as its parametric domain. We present a

systematic way to construct polycube maps for surfaces of arbitrary topology based on global conformal parameterization, and demonstrate the modeling efficacy of the proposed polycube splines in solid modeling and shape computing.

We then further improve the stage of the polycube map construction by introducing the user-controllable polycube map, which allows users to directly select the corner points of the polycubes on the original 3D surfaces, then construct the polycube maps by using the discrete Euclidean Ricci flow. The location of singularities can be interactively placed where no important geometric features exist, which makes the entire hole-filling process much easier to accomplish.

We also develop an effective method to construct polycube maps in an automatic fashion. The proposed algorithm can both construct a similar polycube of high geometric fidelity and compute a high-quality polycube map. In addition, it is theoretically guaranteed to output a one-to-one map.

Finally, we propose a geometry-aware domain decomposition algorithm for T-spline-based manifold modeling by which objects with arbitrary topology (especially objects with long branches) can be modeled elegantly. The segmentation process simultaneously respects local geometric features and global topological structures.

Through our experiments, we demonstrate that the proposed framework is very flexible and can potentially serve as a geometric standard for product data representation and model conversion in shape design and geometric processing. The great potential of our geometric modeling framework will be highlighted through many valuable applications such as shape modeling, remeshing, texture synthesis, finite element analysis, deformation editing, animation morphing, and physics-based modeling. Furthermore, we envision broader application scopes including computer vision, data-driven information retrieval, digital medicine, virtual environments, etc.

*For my beloved family, advisors, friends and colleagues.*

# Contents

# List of Tables

# List of Figures

# Acknowledgements

I would like to express my sincere gratitude to my thesis advisor, Professor Hong Qin for his guidance on research, and his help at various stages of my Ph.D. studies. I cannot imagine myself completing this dissertation without his inspiration, discussion, and encouragement. He led me to the field of geometric modeling and processing and showed such kindness, patience, and deep understanding about geometry and physics that no graduate students expect more from their advisors.

I would like to thank Professor Ying He and Professor Xianfeng Gu for their invaluable inspirations, discussions and collaborations that I have substantially benefited from during my Ph.D. studies.

I would like to thank Professor Joseph Mitchell, Professor Xiangmin Jiao and Professor Klaus Mueller for their kind advice, as well as for serving on various committees.

I would like to thank Professor Ye Duan for taking the time to serve as the external member of my dissertation committee.

I would also like to thank my colleagues in our center of visual computing and computer science department, Xin Li, Xiaohu Guo, Kexiang Wang, Sen Wang, Tingbo Hou, Bo Li, Hao Huang, Miao Jin, Xiaotian Yin, Yang Wang, Feng Qiu, Zhe Fan, Xiang Zeng, Ziyi Zheng, Ruirui Jiang, Wei Zeng, Min Zhang for delightful collaborations and discussions we had together, and thank my friends in Stony Brook, Yunfan Bao, Maohua Lu, Yang Yang, Bo Han, Xiaoxuan Zhang, Yan Zhang, Xiaoxu Lu, Ying Liu, Lei Huang, Xu Han, Jiening Li, Marvin Hazan, Jane Hazan, and so many others for their help during the past five years. The members of the excellent support staff in the Computer Science Department were also of great assistance during the last several years.

The current research presented in this dissertation is partially supported by

NSF grants: IIS-0949467, IIS-0710819, and IIS-0830183.

Last but not least, I want to thank my parents, my sister and brother, for their endless love and support. Without their support, this thesis would not have been possible.

This dissertation is dedicated to them.

# Publications

### Journal Papers

1. **Hongyu Wang**, Ying He, Xin Li, Xiangfeng Gu, and Hong Qin, "Geometry-Aware Domain Decomposition for TSpline-based Manifold Modeling", *in Computer and Graphics (Special Issue of IEEE International Conference of Shape Modeling and Applications), Vol. 33, No. 3, pp. 359 - 368, 2009*.
2. Ying He, **Hongyu Wang**, Chi-Wing Fu, and Hong Qin, "A Divide-and-Conquer Approach for Automatic Polycube Map Construction", *in Computer and Graphics (Special Issue of IEEE International Conference of Shape Modeling and Applications), Vol. 33, No. 3, pp. 369 - 380, 2009*.
3. Xin Li, Xiaohu Guo, **Hongyu Wang**, Ying He, Xianfeng Gu, Hong Qin, "Meshless Harmonic Volumetric Mapping using Fundamental Solution Methods" , *in IEEE Transactions on Automation Science and Engineering (TASE), Vol. 6, No. 3, pp. 409 - 422, 2009*.
4. **Hongyu Wang**, Ying He, Xin Li, Xianfeng Gu, Hong Qin, "Polycube Splines" , *in Computer Aided Design, Vol. 40, No. 6, pp. 721 - 733, 2008*.

### Conference Papers

5. **Hongyu Wang**, Miao Jin, Ying He, Xiangfeng Gu, Hong Qin, "User-controllable Polycube Map for Manifold Spline Construction" , *in Proc. of ACM Solid and Physical Modeling Symposium, pp. 397 - 404, 2008*.
6. **Hongyu Wang**, Ying He, Xin Li, Xiangfeng Gu, Hong Qin, "Polycube Splines" , *in Proc. of ACM Solid and Physical Modeling Symposium, pp. 241-251, 2007*.
7. Xin Li, Xiaohu Guo, **Hongyu Wang**, Ying He, Xiangfeng Gu, Hong Qin,

"Harmonic Volumetric Mapping for Solid Modeling Applications" , *in Proc. of ACM Solid and Physical Modeling Symposium, pp. 109 - 120, 2007*.

8. Ying He, Kexiang Wang, **Hongyu Wang**, Xiangfeng Gu, Hong Qin, "Manifold T-spline" , *in Geometric Modeling and Processing, pp. 409 - 422, 2006*.

**Technical Reports**

9. **Hongyu Wang**. "A Spline-Based Data Modeling Framework over Regular Domains", *Ph.D. Dissertation Proposal, Technical Report, Stony Brook University (SUNY), January 2009*.
10. **Hongyu Wang**. "A Survey on Splines and its Applications", *Master Thesis, Technical Report, Stony Brook University (SUNY), September 2006*.
11. Kyle Hegeman, **Hongyu Wang**, Michael Ashikhmin, Xiangfeng Gu, and Hong Qin. "GPU-based Conformal Flow on Surfaces", *Technical report, May 2006*.

**Other Publications (not included in this dissertation):**

12. Zhenguo Wang, Zhijia Yuan, **Hongyu Wang**, Yingtian Pan, "Increasing the imaging depth of spectral-domaian OCT by using interpixel shift technique" , *in Optics Express, Vol. 14, Issue 16, pp. 7014 - 7023, 2006*.
13. **Hongyu Wang**, Beng Chin Ooi, and Anthony Tung, "iSearch: Mining Retrieval History for Content-Based Image Retrieval", *in 8th International Conference on Database Systems for Advanced Applications, pp. 275 - 282, 2003*.
14. Huiqi Li, Wynne Hsu, Mong Li Lee, and **Hongyu Wang**, "A Piecewise Gaussian Model for Profiling and Differentiating Retinal Vessels", *in International Conference on Image Processing, pp. 14 - 17, 2003*.
15. Wynne Hsu, Mong Li Lee, Huiqi Li, **Hongyu Wang**, and Tien Yin Wong, "Optimal Number of Retinal Vessels Needed for the Prediction of Cardiovascular Disease", *in First SERI-ARVO Meeting on Research in Vision and Ophthalmology, Singapore 2003*.

# Chapter 1

# Introduction

## 1.1 Problem Statement

With the ever-improved modern 3D scanning technologies comes the urgent demand for more efficient, robust, and powerful data modeling techniques for routinely acquired CAD-based digital prototypes which are in forms of raw points or triangular meshes. These data have to be converted into continuous, compact representations to enable geometric design and downstream product development processes (e.g., accurate shape analysis, finite element simulation, and e-manufacturing) in CAE environments. Subdivision surfaces and spline schemes have been extensively investigated during the recent past to fulfill the aforementioned goal.

Real-world physical prototypes are frequently 2-manifolds of complex geometry and arbitrary topology. Naturally, subdivision surfaces can start with a coarser piecewise linear polygonal mesh, and the smooth surface can be calculated as the limit of a sequence of successive refinements from the coarse mesh. Despite their modeling advantages for arbitrary complicated surfaces (especially in animation and digital entertainment), subdivision surfaces have certain drawbacks. Accurate surface evaluation is usually too computationally intensive for realtime applications since most subdivision schemes do not allow closed-form analytic formulation for their basis functions. In addition, extraordinary points solely depend on the connectivity of the control mesh and need special care. On the other hand, spline surfaces

have demonstrated their significance in shape modeling, finite element analysis, scientific computation, visualization, manufacturing, etc. In order to model an arbitrary surface in 3D, conventional spline schemes will segment the surface to many smaller open patches, and cover each patch by a single coordinate system, so that each patch can be modeled by a spline surface. Finally, any generic approach must glue all the spline patches together by adjusting the control points and the knots along their common boundaries in order to ensure continuity of certain degree. The entire segmenting and patching process is performed manually, and it requires user knowledge and skills, and for non-trivial topology and complicated geometry this task is laborious and error-prone.

Manifold splines proposed by Gu, He, and Qin [60] provides a technical solution for directly defining continuous surfaces over arbitrary manifold domains. In their work, they extend the existing spline schemes defined over planar domains to any manifold domain of arbitrary topology using affine structures. To further promote their work in real-world applications, in [75] they present the manifold T-splines, a natural and necessary integration of T-splines and manifold splines, which naturally extends the concept and the currently available algorithms/techniques of the popular planar tensor-product NURBS and T-splines to arbitrary manifold domain of any topological type. Manifold T-splines can be directly defined over the manifold of arbitrary topology to accurately represent various shapes with complicated geometry/topology. It naturally inherits all the attractive properties from T-splines defined over a planar domain, including the powerful local refinement capabilities and the hierarchical organization for LOD control. Despite this earlier success, certain drawbacks of manifold T-splines still remain: (i) There must be singularities for any closed manifold except tori, and in practice small holes must be punched around the singularities in order to enable the easy construction of manifold splines in the finite dimension space. No efforts for hole-filling in the vicinity of singular points were made; (ii) It is impossible to specify the locations of all the singularities on the domain manifold given the fact that the number of singularities is actually fixed, but their positions are somehow globally related; (iii) The proposed domain construction method is far from sufficient for surfaces with boundaries or surfaces with long branches. For surfaces with long branches, the existing global parameterization methods usually introduce extremely large area distortion

and therefore make it even harder and numerically unstable for spline fitting process later on. The only feasible way is to introduce additional cuts in these areas to make it a surface with boundaries and then use double covering method to achieve a better parameterization result. However, this technique will at least double the time complexity and not practical for a large scale complex dataset.

To overcome the above modeling and design difficulties and address the topological issue, we seek novel modeling techniques based on tensor-product spline schemes that would allow designers to directly define continuous spline models over any manifolds (serving as parametric domains). Such a global approach would have many modeling benefits, including no need of the transition from local patch definition to global surface construction via gluing and abutting, the elimination of non-intuitive segmentation and patching process, and ensuring the high-order continuity requirements. More importantly, we can expect a true one-piece representation for shapes of complicated topology, with a hope to automate the entire reverse engineering process (by converting points and/or polygonal meshes to spline surfaces with high accuracy) without human intervention.

Towards this goal, we present a spline-based data modeling framework based on regular domain construction. Regular domains will for sure facilitate the definition of tensor-product splines (NURBS, T-splines, the current industry standard) naturally, and the GPU based geometric modeling and shape analysis. In this dissertation, we study and present various regular domain construction methods (polycube maps and geometry-based domain decomposition) and demonstrate their applications in modeling 3D objects of arbitrary topology.

Through our experiments, we hope to demonstrate that the proposed data modeling framework is very flexible and can potentially serve as a geometric standard for product data representation and model conversion in shape design and geometric processing.

## 1.2 Contributions

In this dissertation, we systematically study the planar splines, spherical splines, and manifold splines, and present a spline-based data modeling framework based on regular domain construction of 3D objects.

In particular, the contributions of this dissertation are as follows:

- We develop the polycube splines which not only inherit all the features of general manifold splines but also have new and more attractive properties of its own, including hierarchical representation, level-of-detail control, regular domain, partition-of-unity for basis functions, easy chart construction, and easy handling of extraordinary points. The polycube splines are naturally built upon the polycube map which serve as its parametric domain. The use of polycubes for spline surface definition and construction is the first attempt to take advantage of the rectangular structure over the boundary of polycubes, allowing the parametric domain to actually mimic the geometry of the modeled objects with lower area distortion while enforcing their topological consistence. We present algorithms to construct polycube maps as the first step to enable spline construction over polycubes of arbitrary topology. We show that the introduced polycube maps easily induce the affine structures except at the finite number of corner points, where we also articulate our strategy for hole-filling. Through extensive experiments on various models, we demonstrate that polycube splines are a very good candidate for accurately representing complicated geometric models of arbitrarily complicated topology with low fitting errors and fewer control points (in comparison with polygonal models).
- We present a novel framework of user-controllable polycube maps, which can overcome the disadvantages of the conventional methods for polycube map construction, and can be easily generalized to complicated surfaces of arbitrary topology. The newly-proposed method allows users to directly select the corner points of the polycubes on the original 3D surfaces in an interactive manner, then constructs the polycube maps by using discrete Euclidean Ricci flow. The resulting polycube map usually has lower area distortion and retains small angle distortion as well, both of which are strongly desirable for spline construction in reverse engineering. We develop algorithms for computing such polycube maps, and show that the resulting user-controllable polycube map serves as an ideal parametric domain for constructing spline

surfaces. The location of singularities can be interactively controlled. Therefore, the later hole-filling process and better data-fitting results can be easily accomplished by placing the singularities at regions where no rich geometric features exist. Through extensive experiments on various models, we demonstrate that the user-controllable polycube maps are well suited for spline construction of complicated geometric models of arbitrarily complicated topology towards better data-fitting results in reverse engineering and solid modeling applications.

- We propose an automatic method to construct a polycube map for surfaces of arbitrary topology. The underlying theory and the entire algorithmic pipeline are clearly documented. Within our framework, the users only need to control how close the polycube resembles the given shape by using two intuitive parameters. With no user intervention after the initial parameter setup, our new method can automatically construct a high-quality polycube map. Furthermore, our method is theoretically sound and numerically robust and stable to guarantee a one-to-one map between the constructed polycube and the given 3D model. We applied the constructed polycube maps to various graphics applications, such as seamless texture synthesis and tiling, T-spline construction, and quad mesh generation. The experimental results have demonstrated the great promise of our method over existing techniques. Extensive comparisons have been conducted to hightlight all the advantages of our algorithm.
- We present a new and effective method to construct manifold T-splines for surfaces of complicated topology/geometry. The most significant new idea of our approach is the geometry-aware object segmentation that simultaneously respects local geometric features and global topological structures. Our divide-and-conquer strategy can decompose an arbitrarily complicated surface into a group of non-overlapping components that comprise branches, handles, and base patches. This object segmentation greatly simplifies objects of arbitrary topological type into a family of genus-zero regular surfaces with four curved boundaries. Popular spline schemes such as tensor-product B-splines and T-splines can be easily employed to model segmented patches with high accuracy. Furthermore, the entire segmentation process is extremely flexible and intuitive, accommodating either full automation or

interactive user control. This local-to-global surface reconstruction is made possible through a global gluing process followed by a global relaxation algorithm. The proposed construction pipeline is extremely flexible and has great potential in shape modeling and reverse engineering applications of complicated real-world objects.

## 1.3 Dissertation Organization

The remainder of this dissertation is organized as follows. In the next two chapters, we briefly review prior research work related to surface parameterization, skeleton extraction, mesh decomposition and splines. In Chapter 4, we present a new concept of polycube splines and develop novel modeling techniques for using the polycube splines in solid modeling and shape computing. In Chapter 5, we introduce a more powerful polycube map construction framework: user-controllable polycube map, which allows users to directly select the corner points of the polycubes on the original 3D surfaces in an interactive manner, then constructs the polycube maps by using discrete Euclidean Ricci flow. The resulting polycube map usually has lower area distortion and retains small angle distortion as well. In Chapter 6, we discuss our work on polycube-map construction for surfaces with complicated topology and geometry in an automatic fashion. Using our method, users can simply specify how close the target polycube mimics a given shape in a quantitative way. Our algorithm can both construct a similar polycube of high geometric fidelity and compute a high-quality polycube map. In addition, our method is theoretically guaranteed to output a one-to-one map. In Chapter 7, we present our geometry-aware domain decomposition framework for T-spline based manifold modeling. Finally, we conclude this dissertation and outline some future research directions in Chapter 8.

# Chapter 2

# Previous Work on Mesh-Based Geometry Processing Techniques

## 2.1 Previous Work on Surface Parameterization

Parameterization of 3D mesh data is very important in shape modeling and interactive graphics applications, including spline construction, texture mapping, remeshing, and morphing. This section reviews the literatures in planar, spherical and global parameterization.

### 2.1.1 Geometric Structures on Surfaces

According to Felix Klein's Erlanger program, a geometry is the study of properties of a space $X$ invariant under a group $G$ of transformations of $X$. For example, planar Euclidean geometry is the geometry of 2-dimensional Euclidean space $\mathbb{R}^2$ invariant under rigid motions (translations, rotations). The central invariant is the distance between two points. Planar affine geometry studies the invariants of the plane under affine transformations (non singular linear maps). The invariants are parallelism and bary-centric coordinates. Real projective geometry on real projective space $\mathbb{RP}^2$ studies the invariants under projective transformation (linear rational maps), and the cross ratio is the central invariant.

Most algorithms in geometric modeling, computational geometry and computer graphics are constructed on planar spaces, and different algorithms are based

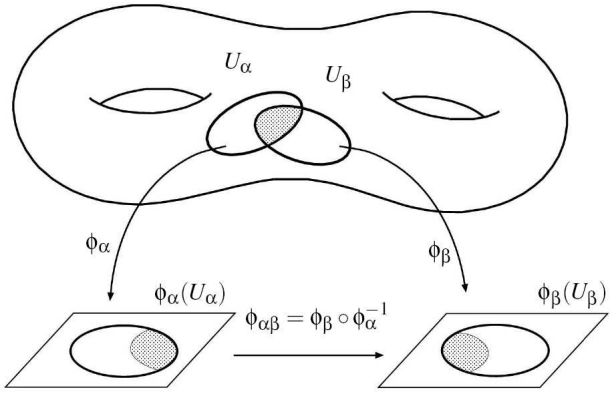



**Figure 1:** Geometric structures. (Image Courtesy of Gu et al. [60])

on different geometries. For example, planar Delaunay triangulation uses Euclidean geometry, and the distances among points play the central role; Splines with planar domains based on polar forms use affine geometry, and the bary-centric coordinates play the central roles. The fundamental task of geometric modeling is to study shapes, therefore it is highly desirable to find a systematic way to generalize the conventional mature planar constructions onto the surfaces. Hence, we need a solid theoretic tool to define different geometries on surfaces.

*Geometric structures* are natural surface structures, which enable different geometries to be defined on the surfaces coherently and allow general planar algorithmic constructions to be generalized onto the surfaces directly.

Surfaces are manifolds, in general, there are no global coordinates. Instead, a surface $M$ is covered by a set of open sets $\{U_\alpha\}$ as shown in Figure 1. Each $U_\alpha$ can be parameterized by a local coordinate system, and a map $\phi_\alpha : U_\alpha \to \mathbb{R}^2$ maps $U_\alpha$ to its parameter domain. $(U_\alpha, \phi_\alpha)$ is a local chart for the surface $M$. A particular point $p$ may be covered by two local coordinates systems $(U_\alpha, \phi_\alpha)$ and $(U_\beta, \phi_\beta)$. The transformation of the local coordinates of $p$ in $(U_\alpha, \phi_\alpha)$ to those in $(U_\beta, \phi_\beta)$ is formulated as the chart transition map $\phi_{\alpha\beta} = \phi_\beta \circ \phi_\alpha^{-1}$. All the charts form the atlas $\{(U_\alpha, \phi_\alpha)\}$.

If all chart transition maps are rigid motions on $\mathbb{R}^2$, then we can discuss the concepts of angle, distance, and parallelism on the surface locally. These geometric measurements can be calculated on one chart, and the results are independent of the choice of the charts. Namely, we can define Euclidean geometry on the surface.

Similarly, if all transition maps are affine, then we can define parallelism on the surface. If all transition maps belong to a particular transformation group of $\mathbb{R}^2$, we can define the corresponding geometry on the surface. Thurston [143] gave the concept of $(X,G)$ *structure*, where $X$ is a topological space, and $G$ is a subgroup of the transformation group of $X$, if $M$ has an atlas $\{(U_\alpha,\phi_\alpha)\}$, such that the parameter domain $\phi_\alpha(U_\alpha) \subset X$ is in space $X$, and the transition maps $\phi_{\alpha\beta} \in G$ are in $G$.

Surfaces have rich $(X,G)$ geometric structures. For topological structure, $X$ is $\mathbb{R}^2$, $G$ is the homeomorphisms of $\mathbb{R}^2$. For conformal structure, $X$ is the complex plane $\mathbb{C}$, $G$ is the group of bi-holomorphic functions (conformal maps). A genus zero surface has a spherical structure, where $X$ is the unit sphere $\mathbb{S}^2$ and $G$ is the rotation group. A genus one surface has an affine structure, which plays vital roles in manifold splines. For affine structure, $X$ is the plane $\mathbb{R}$, and $G$ is the general linear maps $GL(\mathbb{R},2)$.

### 2.1.2 Conformal Structure

For conformal structure, X is the complex plane C, and G is the group of holomorphic functions (conformal maps). Conformal maps are also called angle preserving maps, and locally distances and areas are only changed by a scaling factor. A conformal mapping is intrinsic to the geometry of a mesh, independent of the resolution of the mesh, and preserves the consistency of the orientation.

Because of these nice properties, one big application in computer graphics, computer vision and medical images for conformal structures on surfaces is parameterization. conformal parameterization has been proposed for texture mapping [66,94,103], geometry remeshing [3], and visualization [5,62]. Conformal parameterization continuously depends on the metric of the surface, so it can be used to match two similar surfaces. One such matching method is introduced in [65]. Furthermore, all surfaces can be classified easily by conformal invariants. A method to compute the conformal invariants for meshes is introduced in [65].

Many techniques have been developed to compute conformal parameterizations, but most of them only deal with genus zero surfaces or have to segment the high genus surfaces into patches. These methods decompose meshes into topological disks, then parameterize each patch individually, which introduces discontinuity

along the patch boundaries and conformality can not be preserved everywhere. So global conformal parameterization, which preserves conformality everywhere (except for a few points), is highly desirable.

Global conformal parameterization for closed genus zero surface has been addressed in [52, 62, 65, 66]. Global conformal parameterization for non zero genus closed surfaces with arbitrary boundaries is investigated in [65, 81, 82, 87].

### 2.1.3 Computing Conformal Parameterization

A parameterization of a surface can be viewed as a one-to-one mapping from a suitable domain to the surface. Generally, the parameter domain can be any surface, so constructing a parameterization actually means mapping one surface into another. It is best to parameterize the mesh over a domain which is topological equivalent to it. This significantly reduces the *distortion* introduced by the parameterization without resorting to methods which introduce other artifacts such as cutting seams. So if a mesh is genus zero open surface, it is best to use a planar, or a disk like, domain; if the mesh is genus zero closed surface, it is best to use a spherical parameter domain; for the higher genus surface, a corresponding same genus canonical parameter domain can be used according to the request of the application.

Parameterization is important for many graphics applications, for example, texture mapping, remeshing, morphing, and registration. The main challenge is to produce a planar triangulation that best matches the geometry of the 3D mesh, minimizing some measure of *distortion*, for example, in angles or areas. Many different ways of achieving this have been proposed in the literature.

Since the surfaces are usually represented by triangular meshes, the mappings we are finding are piecewise linear.

#### 2.1.3.1 Parameterization of Triangle Meshes

In computer graphics, people usually use triangle meshes, which is piecewise linear, to approximate underlying smooth surfaces discussed above. So actually, we are constructing a parameterization of a triangulation. A triangulation is defined as follows.

**Definition 1** *Let $\mathcal{T} = T_1, \ldots, T_n$ be a set of triangles in $\mathbb{R}^3$. We call $\mathcal{T}$ a triangulation if*
*(i) $T_i \cap T_j$ is either empty, a common vertex, or a common edge ($i \neq j$), and*
*(ii) the union of the triangles $\Omega_{\mathcal{T}} = \cap_{i=1}^{n} T_i$ is an orientable 2-manifold.*

In general a parameterization $\phi$ of a triangulation $\mathcal{T}$ over a parameter domain $\Omega \subset \mathbb{R}^k$ is a homeomorphism between this domain and the surface of $\mathcal{T}$.

From differential geometry, we know that such a homeomorphism and the inverse parameterization $\psi = \phi^{-1}$ exists if and only if $\Omega$ and $\Omega_{\mathcal{T}}$ are topologically equivalent. As discussed in Hormann's thesis [79], $\psi$ and $\phi$ are uniquely determined by the images $\psi(\mathbf{v})$ which we call the parameter points or parameter values of the vertices of the triangulation. Since we want $\psi$ to be *injective* we have to assure the parameter points to be arranged such that the parameter triangles do not overlap and the parameter triangulation is valid in the sense of Definition 1.

In the following sections, we denote set of all vertex of triangulation as $V$, set of all faces as $F$, and set of all edges as $E$, and we use $v_i$ to denote the i-th vertex, $[v_i, v_j]$ to denote an edge connecting vertex $v_i, v_j$, $[v_i, v_j, v_k]$ to denote a face formed by $v_i, v_j, v_k$, and $N_{v_i}$ to denote the set of vertex in one-ring neighborhood of vertex $v_i$.

#### 2.1.3.2 Conformal Parameterization of Topological Disk Surfaces

**Harmonic Maps** One of the earliest methods for mapping disk-like surfaces into the plane was to approximate a harmonic map using the finite element method. This method was introduced by Eck et al. [34], called *discrete harmonic map*.

***Dirichlet's boundary value problem:*** *Given a 2-manifold $M$ with boundary, a simply-connected region $D \subset \mathbb{R}^2$, and a homeomorphism $g : \partial M \to \partial D$ between the boundaries of $M$ and $D$, find a function $f : M \to R$ that agrees with $g$ on $\partial M$ and is harmonic: $\Delta f = 0$.*

Harmonic maps can be visualized as this. Imagine the topological disk surface $M$ to be composed of elastic, triangular rubber sheets sewn together along their edges. Stretch the boundary of $M$ over the boundary of the target planar polygon boundary $P$ according to the map $g$. The harmonic map minimizes the total energy of this configuration of rubber sheets. The energy is called *Dirichlet energy*.

Variational calculus states that the solution $f$ to this *Dirichlet's boundary value problem* is the minimizer of *Dirichlet energy*

$$E_D(f) = \frac{1}{2}\int_M ||\nabla f||^2,$$

subject to the same boundary condition. Here the manifold $M$ becomes a triangulation $\mathcal{T}$ and the boundary condition is given by the pre-fixed parameter points of the boundary vertices, the Dirichlet energy of the piecewise linear mappings can be represented by

$$E_D(f) = \frac{1}{2}\sum_{[v_i,v_j]\in E} \kappa_{i,j}||h(v_i) - h(v_j)||^2,$$

The following lemma and corollary introduced and proved by Pinkall et al. [118] give us the representation of weight $\kappa_{i,j}$.

**Lemma 2** *Let $f$ be the linear map between two triangles $\triangle_1$ and $\triangle_2$. Then the Dirichlet energy of $f$ is*

$$E_D(f) = \frac{1}{4}(\cot\alpha a^2 + \cot\beta b^2 + \cot\gamma c^2),$$

*where $\alpha,\beta,\gamma$ are the angles in $\triangle_1$ and $a,b,c$ are the corresponding side lengths in $\triangle_2$. (Figure 2).*

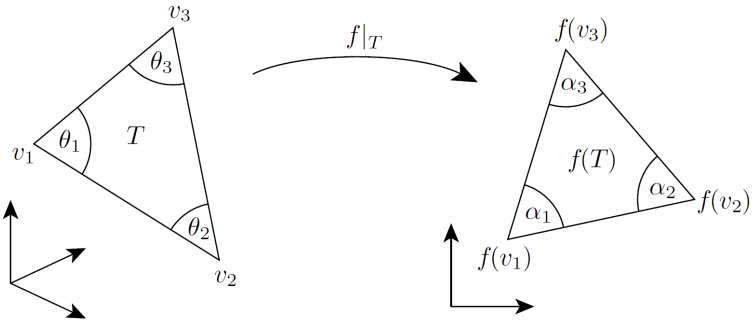



**Figure 2:** Linear map between two triangles. (Image Courtesy of Floater et al. [42])

**Corollary 3** *The Dirichlet energy of a piecewise linear mapping $\psi$ is*

$$E_D(\psi) = \frac{1}{2}\sum_{[v_i,v_j]\in E} \frac{1}{2}w_{ij}||\psi(v_i) - \psi(v_j)||^2$$

*with the harmonic weights*

$$w_{ij} = \frac{1}{2}(\cot\alpha_{ij} + \cot\beta_{ij})$$

*where* $\alpha_{ij}$ *and* $\beta_{ij}$ *are the angles opposite to* $[v_i, v_j]$ *in the adjacent triangles (see Figure 3).*

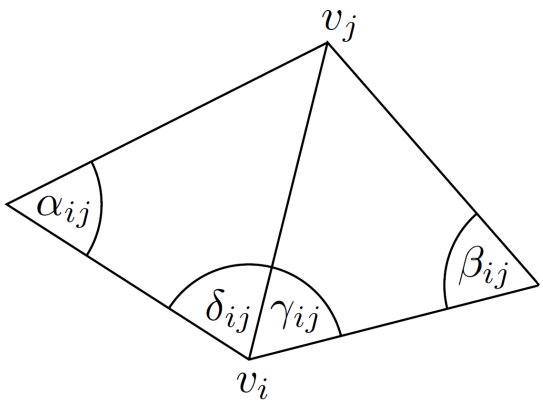



**Figure 3:** Angles for the discrete Dirichlet energy and the mean value coordinates. (Image coutesy of Floater et al. [42])

Now if we fix the parameter points $\psi(v_i)$ for the boundary vertices, and minimize the Dirichlet energy with respect to the interior parameter points $\psi(v_i)$, we get a harmonic map $\psi$ and the corresponding harmonic parameterization $\phi = \psi^{-1}$.

In [34], Eck et al. show that these maps also minimize *metric dispersion*, which measures how much a map stretches regions of small diameter in $\mathcal{T}$.

Minimizing the Dirichlet energy can be reduced to

$$\Delta\psi(v_i) = \sum_{[v_i,v_j]\in E} w_{ij}(\psi(v_i) - \psi(v_j)) = 0$$

and leads to a linear system $A\mathbf{x} = \mathbf{b}$. The positive definiteness is guaranteed for the coefficient matrix $A$(proved in [79]), which shows the existence and uniqueness of a solution to this minimization problem.

**Convex Combination Maps** Another linear parameterization method was introduced by Floater in [39]. First fix the resultant parameter points $\psi(v_i)$ for all

boundary vertices. Then for each interior vertex $v_i$ a set of strictly positive convex weights $\lambda_{ij}$ for $v_i$ and $v_j \in N_{v_i}$ with

$$\sum_{v_j \in N_{v_i}} \lambda_{ij} = 1$$

is chosen; and so the $\psi(v_i)$ for interior vertex are determined by solving the linear system of equations

$$\psi(v_i) = \sum_{v_j \in N_{v_i}} \lambda_{ij} \psi(v_j)$$

Therefore, every interior parameter point is a convex combination of its neighbors. This piecewise linear function $\psi$ that are defined by the parameter points $\psi(v_i)$ above are called convex combinations maps and the following theorem [39, 79], guarantees them leading to proper parameterizations under certain conditions.

**Theorem 4** *If the boundary polygon that is formed by the fixed parameter points $\psi(v)$ for all boundary vertex, then the convex combination map $\psi$ is a bijection, i.e. the planar triangulation $\psi(\mathcal{T})$ is without self-intersections.*

For harmonic maps, the weight $w_{ij}$ can be negative, the weight is not convex, so harmonic map does not fulfill the requirements of this theorem, and we say it is not convex but only an affine combination map.

Floater gives a slightly different version of this theorem with weaker assumptions in [41].

**Corollary 5** *If the boundary polygon is weakly convex and there is no triangle $[v_i, v_j, v_k] \in \mathcal{T}$ for boundary points $v_i, v_j, v_k$ such that $\psi(v_i), \psi(v_j), \psi(v_k)$ are collinear, then $\psi$ is a bijection.*

Now we know the convex combination maps is unique and bijective. The question is whether the weights $\lambda_{ij}$ can be chosen such that the reproduction property holds in addition. The positive answer was given by Floater in [39, 40].

**Shape Preserving Maps and Mean Value Coordinates Maps** Discrete harmonic maps have this reproduction property but are not guaranteed to be injective. The shape-preserving method of [39] is a convex combination mapping designed to get the reproduction property(since it is convex, injectivity is assured). In many

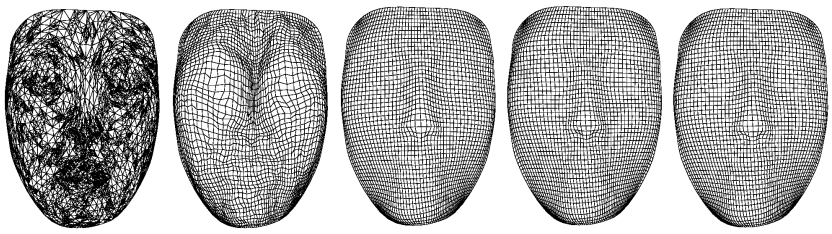


**Figure 4:** Remeshing a triangle mesh with a regular quadrilateral mesh using different parameterization methods. (Image coutesy of Floater et al. [42])

numerical examples, the discrete harmonic map and shape-preserving maps look visually very similar, the two methods begin to differ more, with the shape-preserving map being more robust in the presence of long and thin triangles.

A more recent Floater's paper [40] gives an alternative construction of a convex combination mapping with the reproduction property, which both simplifies the shape-preserving method of [39] and at the same time directly discretizes a harmonic map. It is based on mean value coordinates and motivated as explain below. The numerical results are quite similar to the shape-preserving parameterization. Figure 4 shows the result of first mapping a triangle mesh $(a)$ to a square and then mapping a regular rectangular grid back onto the mesh. The four mappings used are barycentric $(b)$, discrete harmonic $(c)$, shape-preserving $(d)$, and mean value $(e)$.

In [40], Floater gave an observation that harmonic functions satisfy the mean value theorem. At every point in its planar domain, the value of a harmonic function is equal to the average of its values around any circle centered at that point. This suggests finding a piecewise linear map $f : S_{\mathcal{T}} \to S^*$, for a planar triangular mesh $S_{\mathcal{T}}$, which satisfies the mean value theorem at every interior vertex $v_i$ of the mesh. We let $\Gamma_i$ be a circle centered at $v_i$ with radius $r_i > 0$ small enough that $\Gamma_i$ only intersects triangles in $\mathcal{T}$ which are incident on $v_i$. We then demand that

$$f(v_i) = \frac{1}{2\pi r_i} \int_{\Gamma_i} f(v) ds.$$

Some algebra then shows that for the small enough $r_i$, independent of $r_i > 0$, the above equation is similar to the previous frame work for computing harmonic and

convex combination maps, with the weights $w_{ij}$ replaced by

$$w_{ij} = \frac{\tan(\delta_{ij}/2) + \tan(\gamma_{ij}/2)}{||v_j - v_i||},$$

with the angles shown in Figure 3. When $S_{\mathcal{T}}$ is a surface mesh, they use the same weights with the angles taken from the mesh triangles.

Mean value maps guarantee the bijective and reproduction property, but in contrast to discrete harmonic maps(which minimize the Dirichlet energy) and shape-preserving maps(which minimize an energy that is based on second differences), they are not the solution of a minimization problem.

**The boundary mapping** The first step in constructing both the discrete harmonic and the convex combination maps is to choose the boundary mapping $f|_{\partial S_{\mathcal{T}}}$. There are two issues here: $(i)$ choosing the shape of the boundary, and $(ii)$ choosing the distribution of the points around the boundary.

- **Choosing the shape**
  In most applications, we only need the domain to be a rectangle or a circle approximated by a polygon. In these cases, the boundary is convex and the methods of the previous section work well.

  The convexity restriction may generate big distortions near the boundary when the boundary of the surface $S_{\mathcal{T}}$ does not resemble a convex shape. One solution to avoid such distortion is to build a "virtual" boundary, i.e., to augment the given mesh with extra triangles around the boundary so as to construct an extended mesh with a "nice" boundary. This approach has been successfully used by Y. Lee in [93].
- **Choosing the distribution**
  The usual procedure in the literature is to choose some simple boundary mapping such as chord length parameterization, either around the whole boundary, or along each side of the boundary.

  Consider first the case of a smooth surface $S$ with a smooth boundary $\partial S$. According to Riemann Mapping Theorem, $S$ can be mapped into any given simply-connected region $S^* \subset \mathbb{R}^2$ by a conformal map $f : S \to S^*$. Since any

such conformal map defines a boundary mapping $f|_{\partial S} : \partial S \to \partial S^*$, this implies that there must exist some boundary mapping such that the harmonic map it defines is also conformal.

**Linear Methods for Discrete Conformal Maps** Lévy et al. [94] and Desbrun et al. [30] both independently developed a third method to compute discrete conformal mappings which has the advantage of being linear. For a bivariate linear function $g : \mathbb{R}^2 \to \mathbb{R}^2$, Lévy et al. propose measuring the violation of Cauchy-Riemann equations in a least squares sense, i.e., with the conformal energy

$$E_C(g) = \frac{1}{2}((u_x - v_y)^2 + (u_y + v_x)^2).$$

Based on this they find the optimal piecewise linear mapping $f : S_{\mathcal{T}} \to S^*$ by minimizing

$$E_C(f) = \sum_{T \in \mathcal{T}} E_C(f|_T) A(T).$$

$E_C(g)$ can be expressed in terms of the singular values of the Jacobian of $g$ and there is a close relation to the Dirichlet energy.

We can get

$$E_C(g) = \frac{1}{2}(\sigma_1 - \sigma_2)^2$$

and

$$E_C(g)A(S) = E_D(g) - A(g(S))$$

for any planar region $S$. Therefore we have

$$E_C(f) = E_D(f) - A(f)$$

which also shows that $E_C(f)$ is quadratic in the unknowns $f(v)$ and that the normal equations for the minimization problem can therefore be expressed as a linear system of equations.

Desbrun et al. [30] take a slightly different path to arrive at the same system. They start with the finite element method that yields the equations

$$D_p E_D(f) = 0$$

for all parameter points $p = f(v)$ of the interior vertices $v$; then they imposed natural boundary constraints

$$D_p E_D(f) = D_p A(f),$$

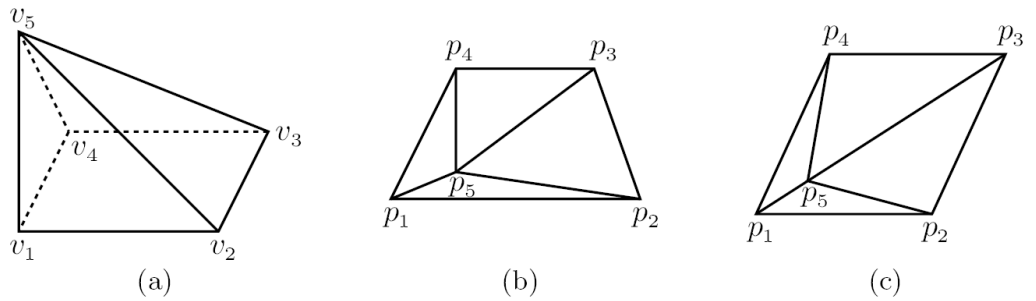



**Figure 5:** Example of two different discrete conformal mappings for the same triangulation. (Image coutesy of Floater et al. [42])

for all $p = f(v)$ on boundary points $v$. But as they also show that $D_pA(f) = 0$ at the interior vertices, this amounts to solving

$$gradE_D = gradA$$

and is thus equivalent to minimizing $E_C(f)$.

However, if there is no additional constraints, $f$ will be degenerate that maps $S_\mathcal{T}$ to a single point because in that case $E_C(f)$ is trivially minimized. Therefore, both papers propose to fix the parameter values $f(v), f(w)$ of two vertices $v, w$ so that a unique non trivial solution can be found. Unfortunately, the solution depends on this choice seriously. For example, if we parameterize the pyramid in Figure 5 (a) whose vertices lie on the corners of a cube, fixing $p_1 = f(v_1)$ and $p_2 = f(v_2)$ gives the solution in (b), while fixing $p_1 = f(v_1)$ and $p_3 = f(v_3)$ results in the parameterization shown in (c).

Finally, this approach sometimes generates folded triangles and according to Floater [42], currently there is no research showing any sufficient conditions that guarantee the resulting parameterization to be a one-to-one mapping.

**Angle Flattening Methods** Sheffer et al. [136] introduce an angle flattening methods to parameterize a genus zero open surface, unlike all the previous methods which focus on finding mapping of vertex positions, their method works on angles. The boundary is also not pre-fixed but computed as a part of the optimization procedure.

In discrete case, the Gaussian curvature at one interior point can be represented by

$$2\pi - \sum_{i=1}^{n} \alpha_i. \tag{1}$$

On planar surface, the Gaussian curvature for interior node is zero everywhere, represented by that equation (1) is zero.

The work of Sheffer et al. starts from this observation. Since the change of the Gaussian curvature inevitably brings the angular deformation, they believe the best case is the deformation is evenly distributed around the vertex.

Denote the set of all one-ring faces around a vertex $v$ by $N_v$, the set of indices of the angles around $v$ by $I(v)$, and denote the sum of these angles by $\theta(v) = \sum_{i \in I(v_i)} \theta_i$.

Using a scalar factor defined on every node $v$

$$s(v) = \begin{cases} 2\pi/\theta(v), & v \text{ is an interior node;} \\ 1, & v \text{ is a boundary node.} \end{cases}$$

The optimal angle $\phi_i$ on planar surface for node $v_i$ is given by:

$$\phi_i = \beta_i s(v_i)$$

where the $\beta_i$ is the original corner angle of the vertex $v_i$.

The objective function is given by

$$F(\alpha) = \sum_i (\alpha_i - \phi_i)^2 w_i \tag{2}$$

where $\alpha_i$ is the planar angle we try to solve, and the $w_i > 0$ are weights, standard initial choice for the weights is $w_i = (\phi_i)^{-2}$. Intuitively, this initialization assures the small angles are not neglected.

The constraints for the optimization problem are

1 All the computed angles should be larger than a preset value $\varepsilon$;

2 The sum of three corners of any triangle equals to $\pi$;

3 The sum of angles around an interior node equals to $2\pi$;

4 Using the sine rule, given an edge and fix any of its interior node, going around all edges sharing same fixed node in counterclockwise order, the

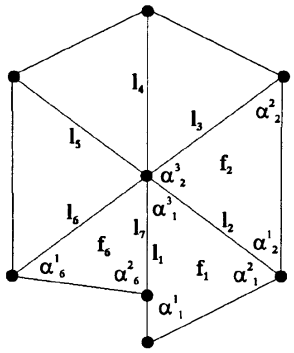



**Figure 6:** Angle Flattening by Sheffer. (Image Courtesy of Sheffer et al. [136])

length of the first one and the last one(coinciding with the first one) agree with each other. As shown in Figure 6, length of $l_7$ should agree with length of $l_1$.

This reduces to a constrained minimization problem, which is non-linear, but proved in [136] to have a global uniquely converged solution. Also the mapping is proved to be injective. One other advantage of this method is it is not affected by the input mesh qualify due to the use of angles only. A main problem for this method is the non-linear system is too slow to solve for practical use.

#### 2.1.3.3 Conformal Parameterization of Genus Zero Closed Surfaces

For meshes having the topology of a sphere, a spherical domain is a best choice for the parameterization. Comparing to domains having other topological structure, this can significantly reduce the distortion introduced by cutting and seaming.

The existence is also guaranteed. Parameterizing a 3D triangle mesh over the sphere is equivalent to embedding its connectivity graph on the sphere, such that the resulting spherical triangles partition the sphere [52]. There is a classical result due to Steinitz guaranteeing the existence. It says that a graph may be embedded on the sphere if and only if it is planar and 3-connected. Thus any triangulation of a closed genus zero manifold satisfies it and can always be mapped to a *spherical triangulation*.

A simplest way to map a closed triangle mesh to the sphere is given by Haker et al. [66]. The primary idea is to convert the mapping to be planar. The method

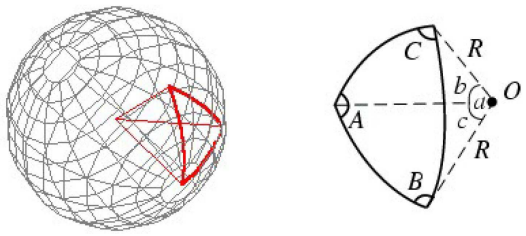



**Figure 7:** Spherical geometry. (Image Courtesy of Sheffer et al. [137])

is first, Cut out one triangle and the left mesh become disk-like and then can be parameterized using planar methods discussed in previous sections, then finally, an inverse stereo projection is applied to map the plane back to a sphere. The main problem of this method is severe distortion. According to Gotsman [52], although the inverse stereo projection is conformal in the continuous case, it does not preserve angles in the discrete case. Also, the projection does not guarantee that the result is a spherical triangulation.

A similar method to take the use of planar mapping can be slicing the closed mesh into two parts, each part is mapped onto a hemisphere, and finally glue the two hemispheres; also the closed meshes can be just sliced to be an open surface patch instead of two, and after the planar mapping, the boundary is seamed together again, as Sheffer did in [135]. These methods can be less distorted than the one of [66], but they severely depend on the slicing, and introduce discontinuities along the cutting boundary.

A better method is to parameterize on sphere directly instead of going back and forth to the plane. Using spherical domain greatly reduce the distortion.

Sheffer et al. [137] use another way to build an injective parameterization. The method is inspired from their angle flattening work on planar parameterization [136] discussed above, and work on the spherical angles instead of the planar location.

A spherical triangle is the region enclosed by three great circles on the unit sphere (a great circle is a circle on the sphere whose center is the origin). As shown in Figure 7, denote the length of the arcs who are the sides of the spherical triangle by $a, b$ and $c$. The spherical *defect* of the triangle is $D = 2\pi - (a+b+c)$. A spherical angle is the dihedral angle between the two planes defined by the two great circles, and we denote these by $A, B$ and $C$. The sum of the spherical angles of a spherical

triangle is always more than $\pi$ and less than $3\pi$. The spherical *excess* of that triangle is $E = A + B + C - \pi$. The solid angle defined by a spherical triangle is the area of the region on the sphere defined by that triangle, and is equal to the excess of the triangle. Hence the sum of all solid angles and the sum of all excesses in a spherical triangulation is $4\pi$. The sum of all spherical angles around any vertex in a spherical triangulation is exactly $2\pi$.

The object function is defined as

$$F(\alpha) = \sum_i (\alpha_i - \phi_i)^2 + \sum_i (e_i - e_i')^2.$$

where the $\alpha_i$ is the angle we try to solve, $\phi_i$ is the optimal angle, $e_i$ is the excess of the planar angles, and $e_i'$ is the optimal excess.

A similar set of constraints is given based on spherical triangles settings accordingly.

The procedure is also non-linear and are reduced to constrained minimization problem which minimizes the least-squares distance of the solution values from their target values set by users. However, this method so far lacks an efficient numerical computation procedure(according to [52]); also, the final embedding procedure may accumulate a numerical error, and the error bound is not given.

Gu et al. [62] compute the conformal mappings of genus zero closed surfaces with a harmonic map, they proved harmonic maps are conformal for genus zero closed surface. So given an initial map, for example, a spherical barycentric map, $h$, it can be diffused to harmonic by minimizing the harmonic energy.

In spatial case, a function $\mathbf{h} : M \to \mathbb{R}^3$ considered, and the Dirichlet energy is defined as

$$E(\mathbf{h}) = \sum_{[v_i, v_j] \in E} w_{ij} ||\mathbf{h}(v_i) - \mathbf{h}(v_j)||^2, \mathbf{h} = (h_0, h_1, h_2)$$

where the weight $w_{ij} = \frac{1}{2}(\cot\alpha_{ij} + \cot\beta_{ij})$ is defined similar to the previous discussion, and the Laplacian on one component is

$$\Delta(h_k) = \sum_{[v_i, v_j] \in E} w_{ij}(h_k(v_i) - h_k(v_j))$$

for $k = 0, 1, 2$. The Laplacian for $\mathbf{h}$ is simply

$$\Delta\mathbf{h} = (\Delta h_0, \Delta h_1, \Delta h_2)$$

**h** is harmonic if the all tangential component of the Laplacian is zero.

The whole optimization problem can be solved by a heat flow procedure:

$$\frac{d\mathbf{h}}{dt} = -D\mathbf{h}(t),$$

the derivative is defined as $D\mathbf{h}(v) = \Delta\mathbf{h}(v) - \Delta\mathbf{h}(v)^{\perp}$ where $\Delta\mathbf{h}(v)^{\perp}$ is the normal component of the Laplacian

$$(\Delta\mathbf{h}(v))^{\perp} = < \Delta\mathbf{h}(v), \mathbf{n}(\mathbf{h}(v)) > \mathbf{n}(\mathbf{h}(v)).$$

#### 2.1.3.4 Conformal Parameterization of Higher Genus Surfaces

For higher genus surfaces, the problem becomes different and much more complicated. A possible method is to cut the given surface into several patches and parameterize them separately. A post-relaxation procedure to blur the discontinuities along cut boundaries may need to be applied. As done in [88, 89, 127, 142]. In addition, a powerful automatical cutting method comes from Morse theory [108]. For more application in topology analysis of shape based on Morse theory, [8, 9, 68, 111] can be referred.

However, like the parameterization in Genus zero closed surfaces, doing parameterization on the surface directly is preferred because it greatly reduces the distortion along the cutting trajectory. Thus, a better approach is to take the topology into account and use the global formulation to obtain the parameterization. However, unlike the genus zero case, some zero points are inevitable due to the topology structure of the surface. For a genus $g(g > 1)$ surface, there are at least $(2g-2)$ zero points .

**One-form Based Global Conformal Parameterization** The problem of computing global conformal parameterization for general closed meshes has two main methods. One is using one-form by Gu et al in [65] and later Gu et al extend the method to general meshes, either closed or open, in [64], [56]. The basic idea is to compute the gradient field of the conformal maps, called holomorphic one-form, which can be integrated to get the final parameterization. The process is briefly explained as follows:

(1) Compute the homology bases $B$ which are $2g$ closed curves $e_1, e_2, \ldots, e_{2g}$;

(2) Compute the cohomology bases $\Omega$ which are $2g$ vector fields $w_1, w_2, \ldots, w_{2g}$;
(3) Compute corresponding harmonic 1-forms $\zeta = \zeta_1, \ldots, \zeta_{2g}$ such that $\zeta_i$ is homologous to $w_i$;
(4) Apply Hodge star operation on $\zeta_i$, and compute holomorphic 1-forms: $\zeta_i + \sqrt{-1}{}^*\zeta_i$.

Then the conformal parameterization is obtained by integrating the gradient field. This method minimize the distortion of the parameterization so that except $2g-2$ zero points, the mapping is conformal.

**Circle Packing Based Global Conformal Parameterization with Curvature Control** Thurston first gave an algorithm to compute conformal structure based on circle packing metric [143]. A practical software system for circle packing can be found in [139].

[87] applied the theory of circle patterns from [12] to globally conformal parameterizations. They obtain the uniform conformality by preserving intersection angles among the circum-circles, each of which is defined from a triangle on the given mesh. In their approach, the set of angles is non-linear optimized first, then the solution is refined with cooperating geometric constraints. They provide several parameterization results, such as 2D parameterization with predefined boundary curvatures, spherical parameterization, and globally smooth parameterization of a high genus model with introduced singularity points. [81] adopt the discrete Ricci flow [16,67] as the tool to do the conformal surface parameterization for Euclidean case, and further compute the uniform hyperbolic metric and real projective structure induced from hyperbolic structure for general surfaces [82].

In theory, the Ricci flow [16] and the variations with circle patterns [12] have the same mathematical power. But the way to compute the uniform metric using the Ricci flow in [81,82] is using Newton' method, which is far more practical than using gradient flow method as circle pattern.

**Applications for Conformal Parameterization** Conformal Parameterization of 3D mesh data is very important in shape modeling and interactive graphics applications, including spline construction, texture mapping, remeshing, and morphing.

In [94], the model to be textured is decomposed into parts with natural shapes,which are homeomorphic to discs, referred to as charts, and each chart

is provided with a quasi-conformal parameterization based on a least-squares approximation of the Cauchy-Riemann equations introduced in this paper, then a new packing algorithm is used to gather them in texture space.

[149] uses the global conformal parameterization to convert the 3D surface texture synthesis problem to a 2D image synthesis problem, which is more intuitive, easier, and conceptually simpler. While the conformality of the parameterization naturally preserves the angles of the texture, they provide a multi-scale technique to maintain a more uniform area scaling factor.This multi-scale method synthesizes nonuniform textures on a 2D geometry image by considering the area stretching factor (the inverse of the conformal factor) in order to obtain the uniform 3D textures.

[3] introduces an interactive remeshing for surface with irregular geometry. First, the original (arbitrary genus) mesh is substituted by a series of 2D maps in parameter space, including conformal parametric domain, area stretching map, mean curvature map and Gaussian curvature map. The user can easily combine these maps to create a control map which controls the sampling density over the surface patch. This map is then sampled at interactive rates allowing the user to easily design a tailored resampling. Once this sampling is complete, a Delaunay triangulation and fast optimization are performed to perfect the final mesh.

[62] proposes a method which can find a unique mapping between any two genus zero manifolds by minimizing the harmonic energy of the map with some constraints added to ensure that the conformal map is unique.

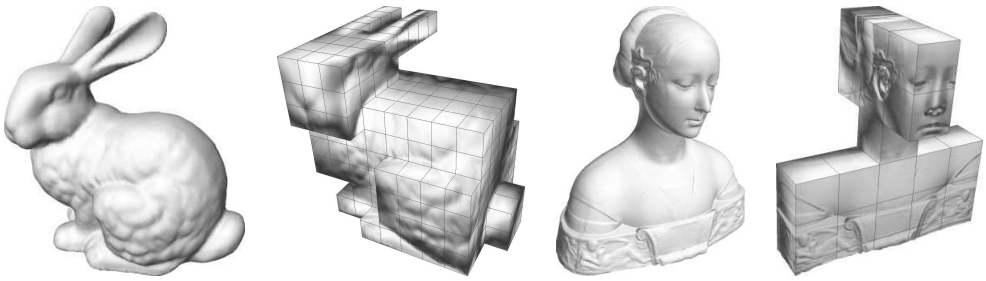



**Figure 8:** Examples of models with poly-cubic parameterizations: the original model ((a)(c)), and shaded parameterization of the mesh over the polycube surface ((b)(d)). (Image Courtesy of Tarini et al. [142])

### 2.1.4 Polycube Maps

Tarini *et al.* [142] pioneered the concept of polycube maps, a new surface parameterization technique in which the parametric domain roughly resembles the given mesh and faithfully represents its topology a natural way to reduce the parametric distortion (see Figure 8 for examples of polycube-maps). They demonstrated that polycube maps naturally lead to a seamless texture mapping method that is simple enough to be implemented in currently available graphics hardware. Their algorithm for the construction of the poly-cubic parameterization of a given mesh $M$ can be summarized as follows (refer to Figure 9 for the 2D analogue of the algorithm):

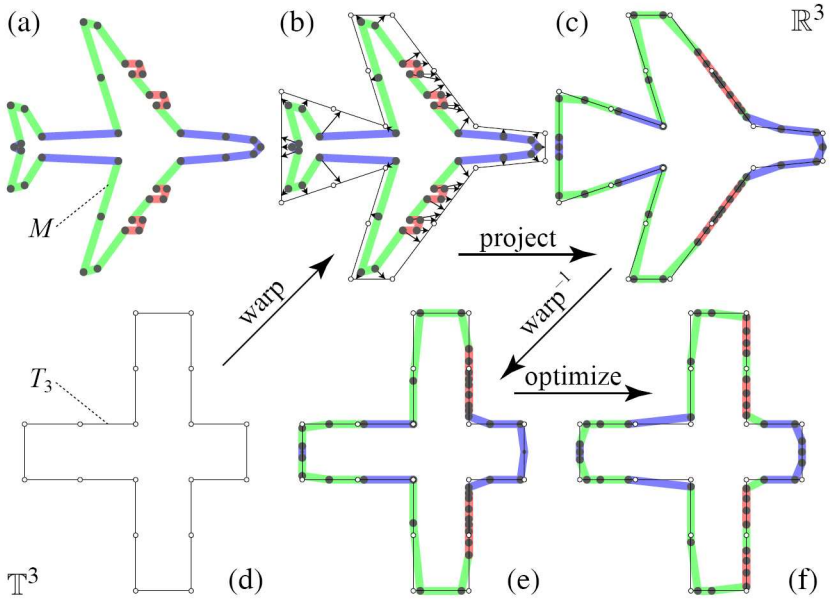



**Figure 9:** The 2D analogue of Tarini's algorithm for the construction of the poly-cubic parameterization. (Image Courtesy of Tarini et al. [142])

1. Define a polycube that has roughly the same shape as the given mesh $M$ and captures all the large scale features.
2. Warp the surface $T_3$ of the polycube from its axis aligned position in the 3D texture space $\mathbb{T}^3$ to the object space $\mathbb{R}^3$.
3. Establish a correspondence between both surfaces by moving every vertex $v$ of $M$ along the surface normal direction onto the deformed polycube.

4. Apply the inverse warp function to the projected vertices and map them to $T_3$.
5. Optimize the texture positions by minimizing the overall distortion of the parameterization.

Tarini *et al.*'s technique is trying to find the one-to-one mapping of the 3D shape and polycube extrinsically, which typically requires the projection of points from one shape to the other. As a result, their method is usually quite difficult to handle cases where the two shapes differ too much and the point projection does not establish the one-to-one correspondence. Recently Lin *et al.* used Reeb graph to segment the surface and then developed an automatic method to construct polycube map [101]. However, their segmentation method may not work for shapes with complicated topology and geometry and does not guarantee a bijection between the polycube and the 3D model.

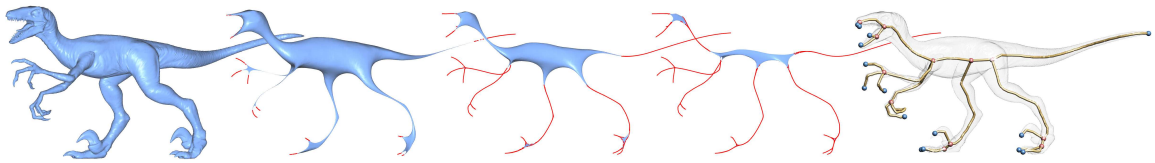


**Figure 10:** Skeleton extraction by mesh contraction. (Image Courtesy of Au et al. [6])

## 2.2 Previous Work on Skeleton Extraction

Curve-skeletons are 1D structures that represent a simplified version of the geometry and topology of a 3D object. Figure 10 shows an example of the skeleton extraction procedure. The extraction of curve-skeletons from 3D models is a fundamental problem in computer graphics and visualization, which has received a lot of attention in recent decades. Compared to the well known skeletal representation, Blum's medial axis [11], which is designed to capture reflectional symmetries in a shape and generally a non-manifold containing 2D sheets that are hard to store and manipulate (refer to Figure 11 for an example), a 1D curve skeleton is more useful in practice due to its topological simplicity, leading to computational efficiency and ease of manipulation. Examples of applications that use a curve

skeleton include: virtual navigation, registration, animation, morphing, scientific analysis, shape recognition, and shape retrieval.

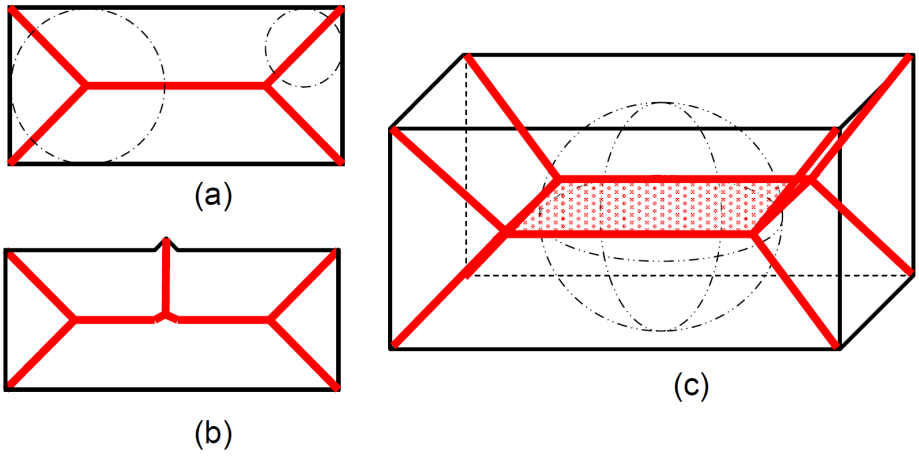



**Figure 11:** A medial axis in 2D (a and b) and a medial surface in 3D (c) and a few examples of inscribed discs (2D) and ball (3D). (Image Courtesy of Cornea et al. [19])

One of the difficulties is that a curve-skeleton is an ill-defined object. This has led to a large number of algorithms and heuristics in the literature and many more constantly being proposed. Many of the algorithms in the literature use different definitions, parameters and thresholds and demonstrate their performance on a limited number of diverse 3D objects. Additionally, some are fine-tuned for a specific application.

Many algorithms for curve skeleton extraction have been developed [19] in recent decades. These methods can be roughly classified into two main categories, volumetric and geometric, depending on whether an interior representation or only the surface representation is used [6].

Most existing curve-skeleton extraction methods make use of a volumetric discrete representation, either a regularly partitioned voxelized representation [102, 113, 150] or a discretized field function defined in the 3D space [69, 154]. **Voxel-thinning methods** [102, 113, 150] extract curve-skeletons from voxelized representations by iteratively removing boundary voxels while maintaining the topology of the input. In [150] they proposed an algorithm for skeleton extraction which first uses iterative least squares optimization to shrink models and preserving their geometries and topologies, then extracts curve skeletons through the

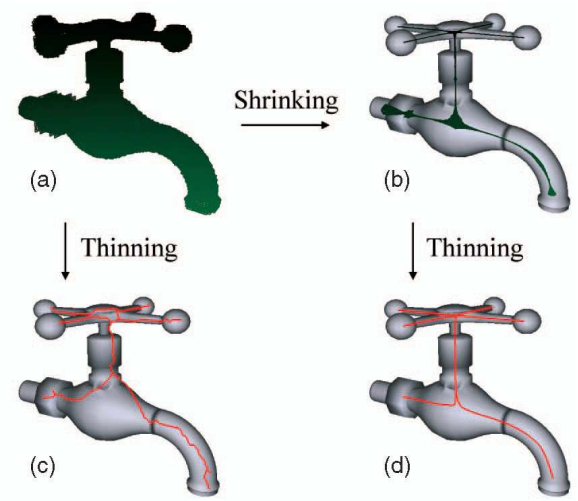



**Figure 12:** System overview of curve-skeleton extraction using iterative least squares optimization. (a) The original volumetric model. (b) The shrunk model transformed by our algorithm. (c) By applying the thinning method to the original model, the extracted skeleton is jagged and deviates from the models center. (d) By applying the thinning method to our shrunk model, a smooth and centered skeleton is obtained. (Image Courtesy of Wang et al. [150])

thinning algorithm, finally prunes unnecessary branches based on shrinking ratios (refer to Figure 12). The proposed method is less sensitive to noise on the surface of models and can generate smoother skeletons. These Voxel-thinning methods differ by the priority for removal and the way of choosing boundary voxels. **Field-based approach** for curve skeleton extraction relies on an Euclidean distance field [69, 104, 154] or an implicit potential field [1, 17, 18] corresponding to the input shape, resulting in a voxelized representation of the internal volume. The skeleton is then computed via volumetric thinning, ridge extraction, or force following along the ridges of a potential field. These methods generally require clear knowledge about the interior of the input shape and the process of connecting candidate voxels is not robust. In general, these volumetric methods share the common drawbacks of potential loss of details, and numerical instability caused by inappropriate discretization resolution. These methods are also usually computationally intensive.

Geometric methods work directly on polygon meshes or point sets without pre-sampling the mesh model into a volumetric representation. Voronoi diagram is

**Figure 13:** An example of Reeb graph: the pseudo-colored surfaces show the function used for computing the Reeb graph. The transparent models show the structure of the Reeb graph and its embedding. (Image Courtesy of Pascucci et al. [114])

a popular geometric approach. Such methods obtain an approximate medial surface by extracting the internal edges and faces of the Voronoi diagram [4, 33, 112] and prune the medial surface to obtain a curve-skeleton. Reeb-graph-based methods are also geometric approaches which have gained much attention in recent years. The Reeb graph [123] is a fundamental data structure that captures the topology of a compact manifold by following the evolution of the level sets of a real-valued function defined on the respective manifold. It is obtained by contracting to a point the connected components of the level-sets of a function defined on a mesh. A lot of algorithms have been proposed to compute Reeb graph of an object using various real-value functions. Aujay et al. [7] proposed a harmonic Reeb graph that uses the harmonic function, found by solving the Laplace equation. He et al. [76] proposed an algorithm for curve skeleton computation by taking advantage of the intrinsic property of harmonic 1-form, i.e., it is determined by the metric and independent of the resolution and embedding. they first construct the skeleton-like Reeb graph of a harmonic function defined on the given poses. Then identify the initial locations of joints by examining the changes of mean curvatures. Finally they refine the joint locations by solving a constrained optimization problem. A robust on-line algorithm for computing Reeb graphs was presented in [114] (Figure 13).

Au et al. [6] (Figure 10) skeletonizes a shape by shrinking it using constrained Laplacian smoothing. Excellent results are obtained, but they can only come from

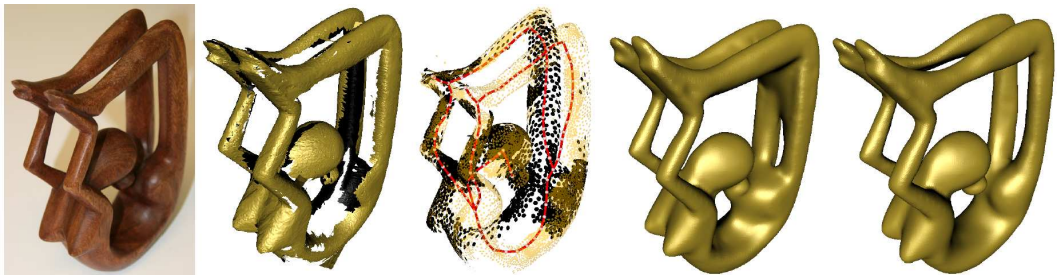


**Figure 14:** Curve skeleton extraction from incomplete point cloud. (Image Courtesy of Tagliasacchi et al. [141])

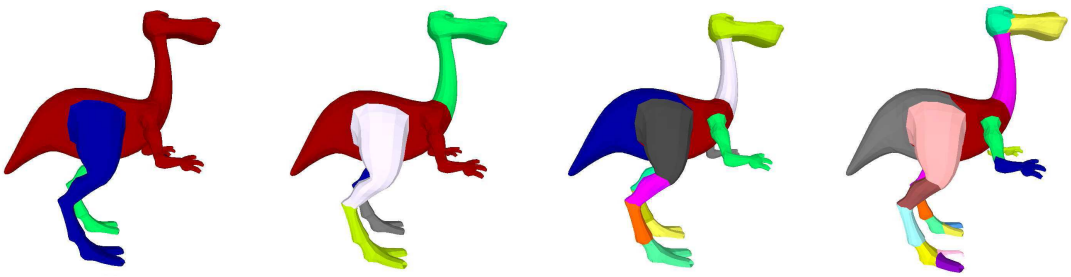
watertight meshes, since Laplacian smoothing requires mesh connectivity and a full model is needed to balance the shrinking process so as to obtain a centered skeleton. More recently, Tagliasacchi et al. [141] presented an algorithm for curve skeleton extraction from imperfect point clouds where large portions of the data may be missing (refer to Figure 14). Their method is primarily based on a novel notion of generalized rotational symmetry axis (ROSA) of an oriented point set. Specifically, given a subset $S$ of oriented points, they introduce a variational definition for an oriented point that is most rotationally symmetric with respect to $S$. Their formulation utilizes normal information to compensate for the missing data and leads to robust curve skeleton computation over regions of a shape that are generally cylindrical. They present an iterative algorithm via planar cuts to compute the ROSA of a point cloud by special handling of non-cylindrical joint regions to obtain a centered, topologically clean, and complete 1D skeleton.

## 2.3 Previous Work on Mesh Decomposition

A hard problem might become easier if only the objects at hand could be cut up into smaller and easier to handle sub-objects. Mesh decomposition is fundamental for many computer graphics and animation techniques. The last few years have witnessed a growing interest in mesh decomposition for computer graphics applications [13, 53, 99, 105, 138].

Mesh decomposition benefits many applications. In metamorphosis [53, 138,

155], mesh decomposition is used for establishing a correspondence. Compression [85] and simplification [46, 156] use decomposition for improving their compression rate. In 3D shape retrieval, a decomposition graph serves as a non-rigid invariant signature [156]. In collision detection, decomposition facilitates the computation of bounding-volume hierarchies [99]. In texture mapping, parameterization is applied to each component [94]. Other potential applications include modification and modeling by parts.



(a) first level (b) second level (c) third level (d) fourth level

**Figure 15:** Hierarchical k-way decomposition of a dino-pet. (Image Courtesy of Katz et al. [86])

Several approaches have been discussed in the past for decomposing meshes. In [13, 14] convex decomposition schemes are proposed, where a patch is called convex if it lies entirely on the boundary of its convex hull. Convex decompositions are important for applications such as collision detection. However, small concavities in the objects result in over-segmentation, which might pose a problem for other applications. In [105] a watershed decomposition is described. In this case, a post-processing step resolves over-segmentation. One problem with the algorithm is the dependency on the exact triangulation of the model. Furthermore, the meaningful components, even planar ones, might get undesirably partitioned. In [46], face clustering is proposed so that the clusters may be well approximated with planar elements. This algorithm is useful for simplification and radiosity, and less for applications seeking the meaningful components. In [99], skeletonization and space sweep are used. Nice-looking results are achieved with this algorithm. However, smoothing effects might cause the disappearance of features for which it is impossible to get a decomposition. In [138] a K-means based clustering algorithm is proposed. The meaningful components of the objects are found. However,

the boundaries between the patches are often jagged and not always correct.

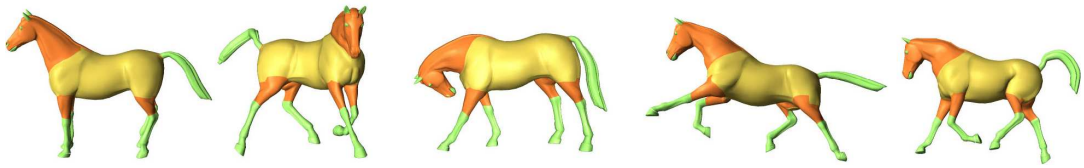


**Figure 16:** Consistent partitioning based on the SDF function. (Image Courtesy of Shapira et al. [133])

Several works explore the strong connection between part-partitioning and skeletonizing [100, 109, 144, 145, 152]. In [144] feature points are extracted from a mesh, and are used to calculate an invariant mapping function, revealing important parts in the mesh. Geometrical and toplogical analysis using Reeb graphs enable the authors to extract a visually meaningful skeleton. This skeleton was employed in a follow-up work [145] to calculate a hierarchical segmentation. In [86] they present an algorithm for hierarchically decomposing meshes. The key idea of the algorithm is to first find the meaningful components of the mesh and only then focus on generating the exactly boundaries between the components. The object part decomposition facilitates the definition of a skeleton, which in turn is used for deformations and animation (Figure 15). In [133] they target the problems of partitioning and skeleton extraction of a family of 3D meshes with consistency. They present an algorithm on a volume-based shape-function called the shape-diameterfunction (SDF), which remains largely oblivious to pose changes of the same object and maintains similar values in analogue parts of different objects. The SDF is a scalar function defined on the mesh surface expressing a measure of the diameter of the objects's volume in the neighborhood of each point on the surface. Using the SDF it is possible to process and manipulate families of objects which contain similarities (Figure 16).

Au K.-C. et. al. in [6] provided a simple mesh segmentation algorithm in their curve-skeleton extraction framework by exploiting the induced skeleton-mesh mapping and the local thickness of each skeleton node. They first order the branches of the extracted curve-skeleton according to their approximate volume. Starting from the thickest branch, they iteratively assign a cut to each branch to segment the mesh, with each cut resulting in exactly one additional segment. The cutting stops


(a) (b)

**Figure 17:** A visualization of the local thickness at each skeleton node in (a), and segmentation results in (b). (Image Courtesy of Au et al. [6])

when every branch has been assigned one cut or a desired number of segments specified by the user has been reached (Figure 17).

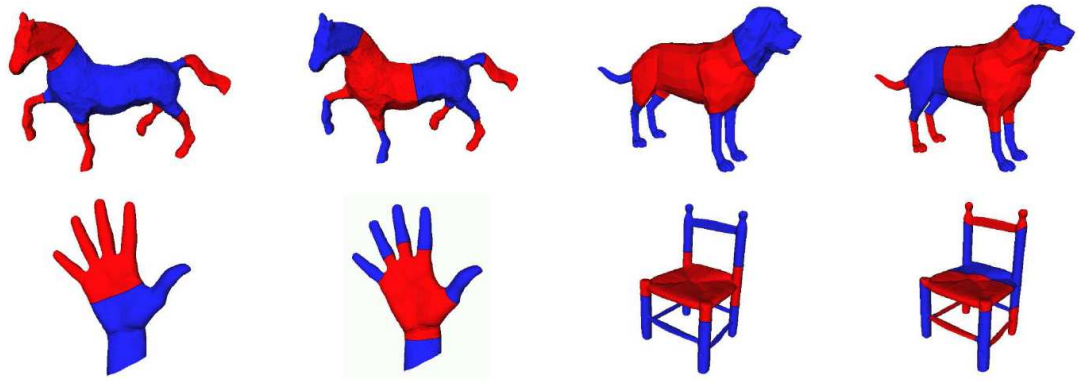


**Figure 18:** Segmentation induced by the analysis of discrete Laplace-Beltrami operators. (Image Courtesy of Reuter et al. [124])

More recently, in [124] Reuter et al. analyzed the correctness of the Laplacian eigenfunctions of different discretizations of the Laplace- Beltrami operator. Then, they selected the FEM operators for eigenfunctions computation, and derived a set of segmentations from the nodal domains of the eigenfunctions in the first part of the Laplacian spectrum. Golovinskiy et al. extended the idea of single-mesh segmentations to a segmentation of multiple meshes [47]: they simultaneously segment models and create correspondences between segments. Specifically, they first build a graph whose nodes represent faces of all the models in the set, and whose edges represent links between adjacent faces within a mesh, and between corresponding

**Figure 19:** Consistent segmentation of 3D models. (Image Courtesy of Golovinskiy et al. [47])

faces of different meshes. They then cluster the graph, creating a segmentation in which adjacent faces of the same model and corresponding faces between different models are encouraged to belong to the same segment.

# Chapter 3

# Previous Work on Splines

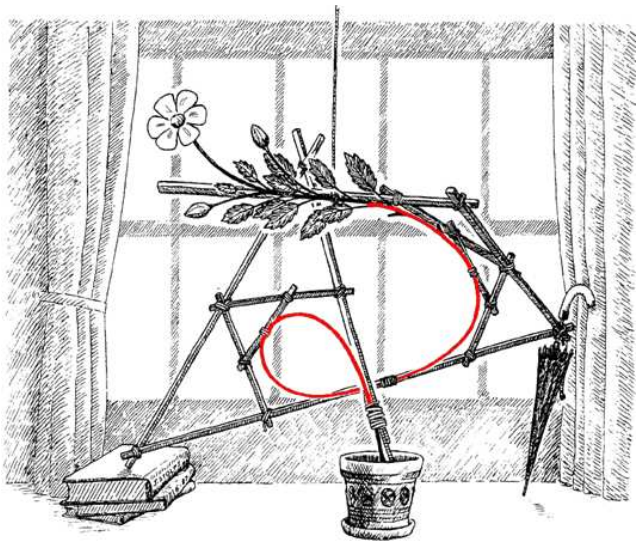


**Figure 20:** An example of spline devices used to draw smooth shapes. (Image Courtesy of Poston et al.)

In mathematics, the term *splines* refers to smooth, piecewise polynomials. They are ideal tools for applications where continuous representations are critical. The classical spline or engineering spline is usually a wooden beam or metal strip, which can be used to draw smooth curves (see Figure 20). The first study on splines goes back to 1946 by Schoenberg. Since then, splines become a very active research area due to the rapid development of computational science. There exists

huge number of literatures on theoretical foundation of splines and their applications. The most popular spline schemes, such as tensor product Bézier surfaces, tensor product *B*-spline surfaces, triangular Bézier surfaces and *B*-patches, can be unified as the different variations of polar forms [120, 122, 132]. We shall briefly explain the concept of polar forms, and then, we concentrate on the introduction of different spline schemes, including the planar splines and manifold splines.

## 3.1 Polynomials and Polar Forms

The most basic class of curves and surfaces is the class of parametric polynomials. In the context of graphics and CAGD these curves and surfaces are best studied with the help of a classical mathematical tool, the polar form [122, 131].

**Definition 6 (Affine Map)** *A map $f : \mathbb{R}^k \to \mathbb{R}^t (k \geq 1)$ is affine, if and only if it preserves affine combinations, i.e., if and only if $f$ satisfies $f(\sum_{i=0}^{m} \alpha_i \mathbf{u}_i) = \sum_{i=0}^{m} \alpha_i f(\mathbf{u}_i)$ for all scalars $\alpha_0, \ldots, \alpha_m \in \mathbb{R}$ with $\sum_{i=0}^{m} \alpha_i = 1$.*

**Definition 7 (Symmetric, Multi-Affine)** *Let $F$ be an n-variable map. $F$ is symmetric if and only*

$$F(\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n) = F(\mathbf{u}_{\pi(1)}, \mathbf{u}_{\pi(2)}, \cdots, \mathbf{u}_{\pi(n)})$$

*for all permutations $\pi \in \sum_n$. The map $F$ is multi-affine if and only if $F$ is affine in each argument if the others are held fixed.*

The well-known blossoming principle indicates that any polynomial is equivalent to its polar form [132]:

**Theorem 8 (Blossoming Principle)** *Polynomials $F : \mathbb{R}^k \to \mathbb{R}^t (k \geq 1)$ of degree n, and a symmetric multi-affine map $f : (\mathbb{R}^k)^n \to \mathbb{R}^t$ are equivalent. Given a map of either type, unique map of the other type exists that satisfies the identity $F(\mathbf{u}) = f(\underbrace{\mathbf{u}, \cdots, \mathbf{u}}_{n})$. The map $f$ is called the multi-affine polar form or blossom of $F$.*

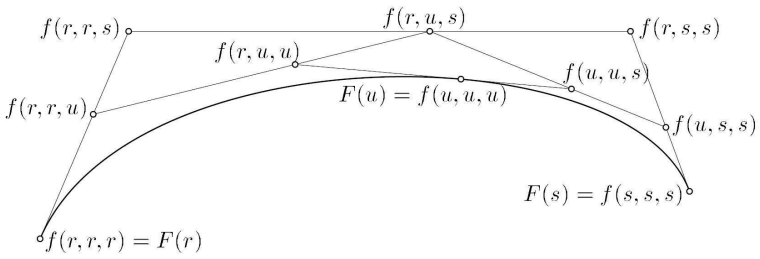



**Figure 21:** The de Casteljau Algorithm in the case n=3. (Image Courtesy of Seidel et al. [131])

## 3.2 Bézier Curve

A Bézier curve in its most common form is a simple cubic equation that can be used in any number of useful ways. It is attributed and named after a French engineer, Pierre Bézier, who used them for the body design of the Renault car in the 1970's. They have since obtained dominance in the typesetting industry and in particular with the Adobe Postscript and font products. A Bézier curve can be defined as following:

**Definition 9 (Bézier Curve)** *Given a set of $n+1$ control points $P_0, P_1, \ldots, P_n$, the corresponding **Bézier curve** (or Bernstein-Bézier curve) is given by*

$$C(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$$

*where $B_{i,n}(t)$ is a Bernstein polynomial $B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$ and $t \in [0,1]$.*

The Bézier representation of a polynomial F is closely related to its polar form [27, 28]:

**Theorem 10** ***(Bézier Points)*** *Let $\Delta = [r,s]$ be an arbitrary interval. Every polynomial $F : \mathbb{R} \to \mathbb{R}^t$ can be represented as a Bézier polynomial w.r.t. $\Delta$. The Bézier points are given as*

$$\mathbf{b_j} = f(\underbrace{r,\ldots,r}_{n-j},\underbrace{s,\ldots,s}_{j}), \tag{3.1}$$

*where $f$ is the polar form of $F$.*

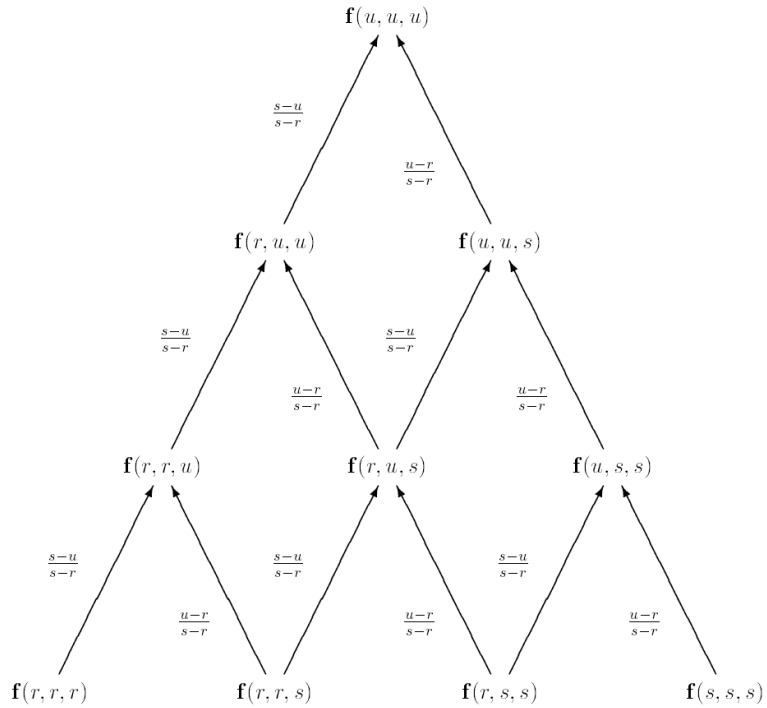



**Figure 22:** The de Casteljau algorithm for a cubic Bézier curve. (Image Courtesy of Seidel et al. [131])

Equation (3.1) immediately leads to an evaluation algorithm that recursively computes the values

$$\begin{aligned}
\mathbf{b}_j^l(u) &= f(\underbrace{r,\ldots,r}_{n-l-j},\underbrace{u,\ldots,u}_{l},\underbrace{s,\ldots,s}_{j}) \\
&= \tfrac{s-u}{s-r} f(\underbrace{r,\ldots,r}_{n-l-j+1},\underbrace{u,\ldots,u}_{l-1},\underbrace{s,\ldots,s}_{j}) + \tfrac{u-r}{s-r} f(\underbrace{r,\ldots,r}_{n-l-j},\underbrace{u,\ldots,u}_{l-1},\underbrace{s,\ldots,s}_{j+1}) \\
&= \tfrac{s-u}{s-r}\mathbf{b}_j^{l-1}(u) + \tfrac{u-r}{s-r}\mathbf{b}_{j+1}^{l-1}(u)
\end{aligned}$$

from the given control points. For $l = n$ we finally compute $\mathbf{b}_0^n = f(u,\ldots,u) = F(u)$ which is the desired point on the curve. The resulting computational scheme is illustrated by Figure 21 and Figure 22. This algorithm is called *de Casteljau Algorithm*, which was first studied by Paul de Faget de Casteljau [27, 28].

Formula (3.1) also shows that the de Casteljau Algorithm offers a way to subdivide a Bézier curve: suppose that we wish to subdivide a Bézier curve F over a

given interval $\Delta = [s,t]$ at an arbitrary parameter $u \in \Delta$. The new Bézier points of the left and right segments $F_l$ and $F_r$ with respect to the subintervals $\Delta_l = [r,u]$ and $\Delta_r = [u,s]$ are given as

$$\mathbf{b}_0^l = f(r,\ldots,r), \mathbf{b}_1^l = f(r,\ldots,r,u), \ldots, \mathbf{b}_n^l = f(u,\ldots,u)$$

and

$$\mathbf{b}_0^r = f(u,\ldots,u), \mathbf{b}_1^r = f(u,\ldots,u,s), \ldots, \mathbf{b}_n^r = f(s,\ldots,s).$$

Bézier curves satisfy the following shape properties:

1. Convex Hull Property: A Bézier curve is contained in the closed convex hull of its Bézier polygon.
2. End Point Interpolation: A Bézier curve interpolates the end points of its control polygon and is tangent to the control polygon there.
3. Variation Diminishing Property: The number of intersection points of a Bézier curve with an affine hyperplane H is bounded by the number of intersection points of H with the control polygon. Intuitively this means that a Bézier curve wiggles no more than its control polygon.
4. Affine Invariance: The relationship between a Bézier curve and its control polygon is invariant under affine transformations.

Undesirable properties of Bézier curves are their numerical instability for large numbers of control points, and the fact that moving a single control point changes the global shape of the curve. The former is sometimes avoided by smoothly patching together low-order Bézier curves. A generalization of the Bézier curve is the B-spline.

## 3.3 B-Splines

*B*-splines (short for Basis Splines) go back to Schoenberg who introduced them in 1946 [125, 126] for the case of uniform knots. *B*-splines over nonuniform knots go back to a review article by Curry in 1947 [20]. de Boor derives the recursive evaluation of *B*-spline curves [25]. It was this recursion that made *B*-splines a truly viable tool in CAGD. Before its discovery, *B*-splines were defined using a tedious divided difference approach which was numerically unstable. Later on, Gordon and

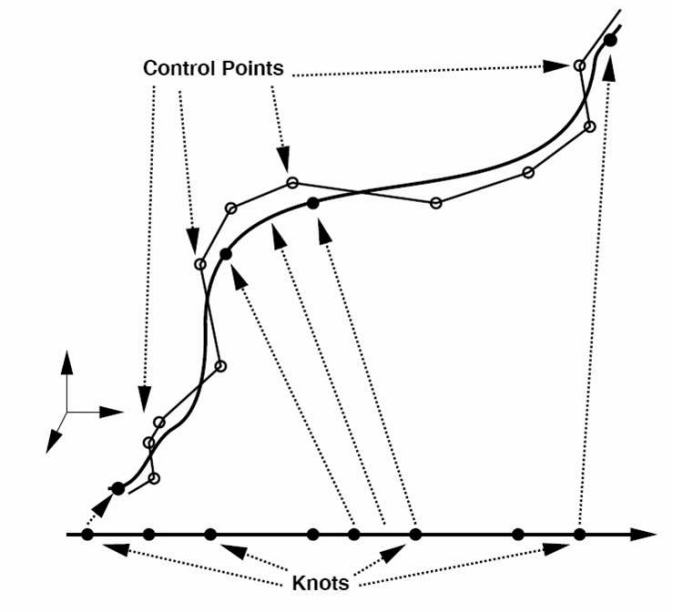



**Figure 23:** An example of B-spline curves.

Riesenfeld [49] realize that de Boor's recursive *B*-spline evaluation is the natural generalization of the de Casteljau algorithm and Bézier curves are just subset of *B*-spline curves. Versprille [146] generalizes *B*-spline curves to NURBS (non-uniform rational *B*-spline) which has become the standard curve and surface form in the CAD/CAM industry [117]. In this section, we first introduce basic concepts of B-spline and its polar form representation, then describe the rational extention of B-splines - the NURBS.

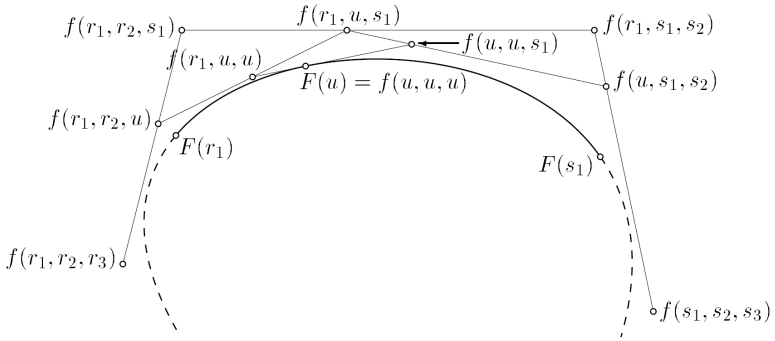



**Figure 24:** The de Boor Algorithm in the case n=3. (Image Courtesy of Seidel et al. [131])

### 3.3.1 Univariate B-Splines

**Definition 11 (B-Spline)** *Let a vector known as the knot vector be defined*

$$\mathbf{T} = \{t_0, t_1, \ldots, t_m\}$$

*where* $\mathbf{T}$ *is a nondecreasing sequence with* $t_i \in [0,1]$ *, and define control points* $P_0, \ldots, P_n$ *. Define the degree as*

$$p \equiv m - n - 1$$

*The "knots"* $t_{p+1}, ..., t_{m-p-1}$ *are called internal knots.*
*Define the basis functions as*

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1} \text{ and } t_i < t_{i+1}; \\ 0 & \text{otherwise}. \end{cases}$$

$$N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}$$

*Then the curve defined by*

$$C(t) = \sum_{i=0}^{n} P_i N_{i,p}(t)$$

*is a* ***B-Spline****.*

Figure 23 shows an example of B-spline curves. Similar to section 3.2, we have the following [121, 122]:

**Theorem 12** ***(de Boor Points)*** *Every polynomial* $F : \mathbb{R} \to \mathbb{R}^t$ *can be represented as a B-spline segment over a non-decreasing knot sequence* $r_n \leq \ldots \leq r_1 < s_1 \leq \ldots \leq s_n$*. The de Boor points are given as*

$$\mathbf{d_j} = f(r_1, \ldots, r_{n-j}, s_1, \ldots, s_j), \tag{3.2}$$

*where* $f$ *is the polar form of* $F$*.*

Theorem 12 and the de Boor Algorithm are illustrated by Figure 24 and Figure 25.

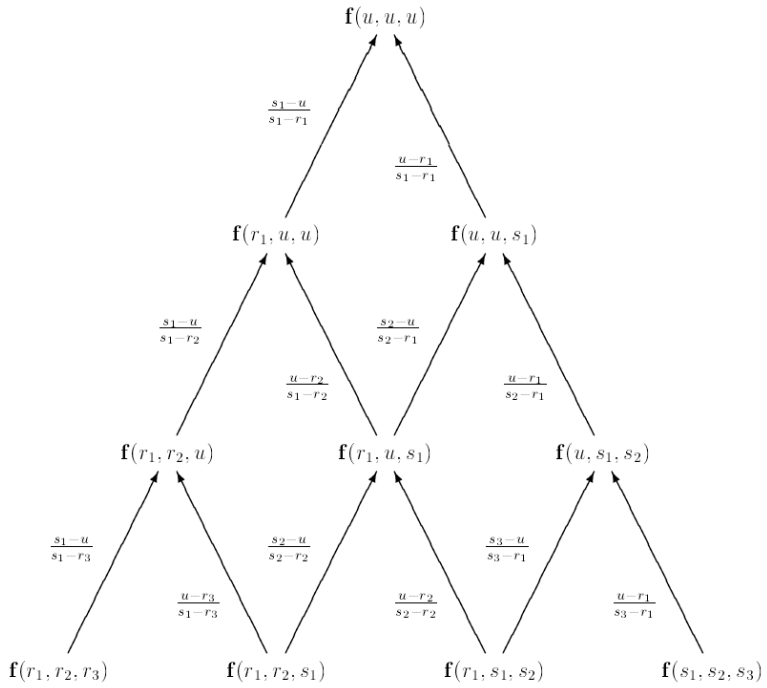



**Figure 25:** The deBoor Algorithm for a cubic B-spline segment. (Image Courtesy of Seidel et al. [131])

The B-spline basis functions are positive and form a partition of unity. In addition, they have local support ($N_i^n(u) = 0$ for $u \notin [t_i, t_{i+n+1}]$), and it can be shown that they are $C^{n-\mu}$-continuous at a knot of multiplicity $\mu$. The knot values determine the extent of the control of the control points.

B-spline curves have similar shape properties as Bézier curves:

1. Convex Hull Property: A B-spline is contained in the convex hull of its de Boor points. Moreover, if $u \in [t_j, t_{j+1})$, then $F(u) \subset [\mathbf{d}_{j-n}, \ldots, \mathbf{d}_j]$(local convex hull property).
2. Multiple Control Points:If $n$ control points $\mathbf{d}_{j-n+1} = \ldots = \mathbf{d}_j = \mathbf{d}$ coincide, then $F(t_{j+n}) = \mathbf{d}$, i.e., the curve interpolates to this point and is tangent to the control polygon there.
3. Collinear Control Points:If $n+1$ control points $\mathbf{d}_{j-n}, \ldots, \mathbf{d}_j$ lie on a line $L$, then $F([t_j, t_{j+1}]) \in L$, i.e., the curve contains a line segment.

4. Multiple Knots:If $n$ knots $t_{j+1} = \ldots = t_{j+n} =: t$ coincide, then $F(t) = \mathbf{d}_j$, i.e., the curve interpolates to this point and is tangent to the control polygon there.
5. Variation Diminishing Property:The number of intersection points of B-spline curve with an affine hyperplane $H$ is bounded by the number of intersection points of $H$ with the control polygon. Intuitively this means that a B-spline curve wiggles no more than its control polygon.
6. Affine Invariance Then relationship between a B-spline curve and its control polygon is invariant under affine transformations.

Specific B-spline types include the *nonperiodic B-spline* (first $n+1$ knots equal 0 and last $n+1$ knots equal to 1) and *uniform B-spline* (internal knots are equally spaced). Bézier curves are a special case of B-splines (for special knot vector $T = (\underbrace{s,\ldots,s}_{n} < \underbrace{t,\ldots,t}_{n})$)

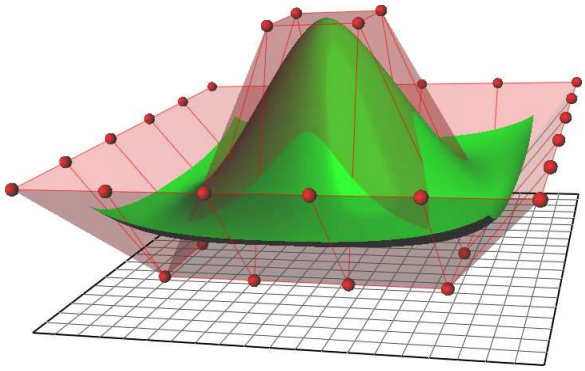


**Figure 26:** An example of tensor product surfaces. (Image Courtesy of Wikepedia)

### 3.3.2 Tensor Product B-Spline

By far the most popular surfaces in CAGD and computer graphics are tensor product surfaces (see Figure 26): Given a curve scheme $F(u) = \sum_{i=0}^{n} B_i(u)\mathbf{b}_i, \mathbf{b}_i \in \mathbb{R}^t$, the corresponding tensor product scheme is defined as

$$F(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_i(u)B_j(v)\mathbf{b}_{ij}, \mathbf{b}_{ij} \in \mathbb{R}^t,$$

which can also be written as
$F(u,v) = \sum_{i=0}^{n} B_i(u)\mathbf{b}_{i_v}$ with $\mathbf{b}_{i_v} = \mathbf{b}_i(v) = \sum_{j=0}^{m} B_j(v)\mathbf{b}_{ij}$.

The last equation demonstrates that tensor product surfaces may be considered as curves of curves. In generation of the curve case, the Bézier points $\mathbf{b}_{ij}$ of $F$ in the representation $F(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u)B_j^m(v)\mathbf{b}_{ij}$ as a tensor product Bézier surface over $[p,q]\times[r,s]$ are given as

$$\mathbf{b}_{ij} = f_{TP}(\underbrace{p,\ldots,p}_{n-i},\underbrace{q,\ldots,q}_{i};\underbrace{r,\ldots,r}_{m-j},\underbrace{s,\ldots,s}_{j})$$

while the de Boor points $\mathbf{d}_{ij}$ of $F$ in the representation

$$F(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_i^n(u)N_j^m(v)\mathbf{d}_{ij}$$

as segment of a tensor product B-spline surface over the knot vectors $S = \{s_i\}$ and $T = \{t_j\}$ are given as

$$\mathbf{d}_{ij} = f_{TP}(s_{i+1},\ldots,s_{i+n};t_{j+1},\ldots,t_{j+m}).$$

Many algorithms that have been discussed for Bézier and B-spline curves can then be generalized to B'ezier and B-spline tensor product surfaces.

### 3.3.3 NURBS

Many CAD systems use conic sections(ellipses, parabolas and hyperbolas) as basic components for the construction of more complex objects. Parabolas can be easily represented as B-splines, e.g. as quadratic Bézier curves. However, it is impossible to represent an ellipse or a hyperbola exactly by a B-spline. One can only approximate ellipses or hyperbolas with arbitrary prescribed tolerance. Rational curves (curves with the form $R(u) = \frac{C(u)}{w(u)}$ where $C$ is a polynomial curve and $w$ is a scalar polynomial) allow an exact representation of a conic section. The most important class of rational spline curves is the set of **N**on-**U**niform **R**ational **B**-**S**plines, briefly NURBS, which are the rational extension of B-splines, have all the nice features of B-splines, and include conic sections. In this section we will give a description of this class.

**Definition 13 (NURBS)** *Let a vector known as the knot vector be defined* $\mathbf{T} = \{t_0 \leq t_1 \leq \ldots \leq t_{k+n} \leq t_{k+n+1}\}$ *with the restriction that the interior knots have at most*

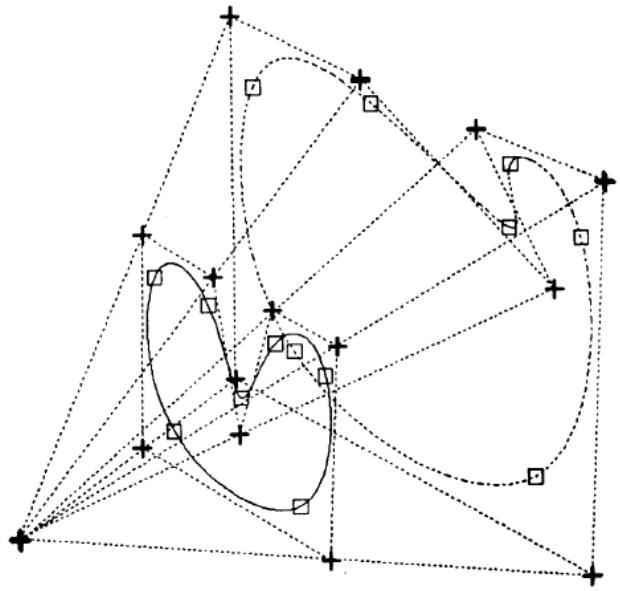


**Figure 27:** Rational curve is the projection of an integral curve. (Image Courtesy of Barsky et al. [10])

*multiplicity n, that is* $t_i < t_{i+n}$ *for* $i = 1,2,\ldots,k$*, define control points* $P_0,\ldots,P_k \in \mathbb{E}^d$*, and define positive weights* $w_0,w_1,\ldots,w_k$*, associated to the control points* $P_i$*. The analytic representation of the corresponding NURBS curve R of degree n in* $\mathbb{E}^d$ *is given by*

$$R(u) = \frac{\sum_{i=0}^{k} w_i P_i N_i^n(u)}{\sum_{i=0}^{k} w_i N_i^n(u)}, u \in [t_0, t_{k+n+1}] \qquad (3.2)$$

*where* $N_i^n, i = 0,1,\ldots,k$ *are the normalized B-spline basis functions of degree n corresponding to the knot vector T.*

Since in definition 13 the weights are assumed to be positive, the denominator in (3.2) does not vanish. If all the weights coincide and the knots at the end points have multiplicity n+1, the curve is a B-spline curve.

A further advantage of a rational formulation is that it is invariant under projective transformation (only affine invariance holds for its integral counterpart). Additionally, there are weights which can be used to control shapes in a manner similar to shape parameters. Geometrically, a rational curve can be viewed as the projection of an integral curve from a vector space of one higher dimension (see Figure 27).

The NURBS curve (3.2) can be obtained by projecting the B-spline curve $\hat{R}$ in $\mathbb{E}^{d+1}$ having the same knot vector and control points $\hat{P}_i = (w_i P_i, w_i)$. As a consequence, the NURBS inherit all the nice properties from B-splines, and can represent conic sections.

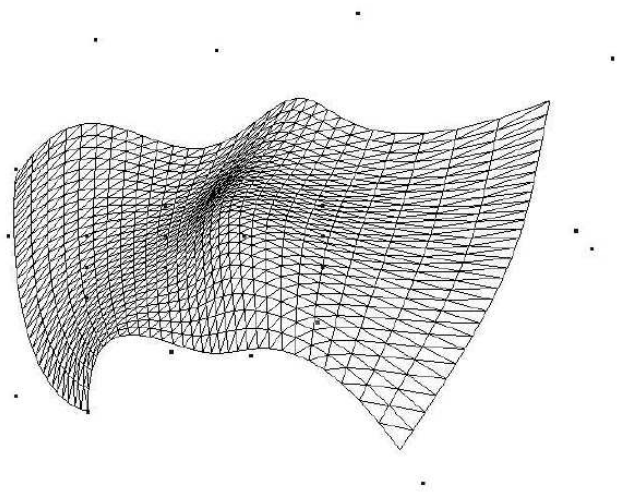


**Figure 28:** NURBS surface, 6x4 control points. (Image Courtesy of Barsky et al. [10])

If we extend equation (3.2) in two parametric directions we obtain a *surface* with the same properties as the NURBS curve:

$$F(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} w_i P_i B_i(u) B_j(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m} w_i B_i(u) B_j(v)} \qquad (3.3)$$

The surface does not have to be of equal degree in both directions. Observe the surface in its rendered form in Figure 28 where we clearly see the local control property.

## 3.4 Hierarchical B-Splines and T-splines

Forsey and Bartels present the hierarchical *B*-spline [44], in which a single control point can be inserted without propagating an entire row or column of control points. In their work two concepts are introduced: local refinement using an efficient representation, and multi-resolution editing. These notions extend

to any refineable surface such as subdivision surfaces. Gonzalez-Ochoa and Peters [48] present the localized-hierarchy surface splines which extended the hierarchical spline paradigm to surfaces of arbitrary topology.

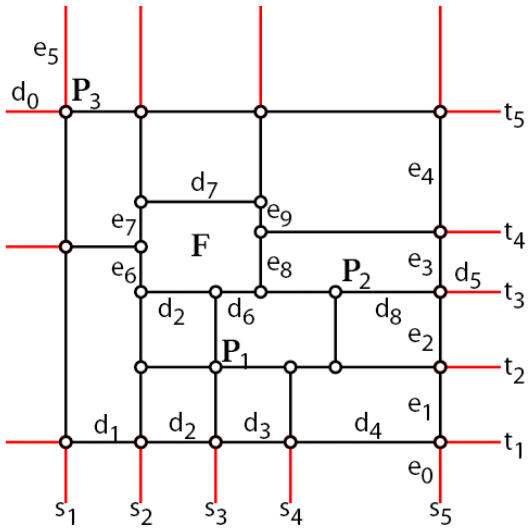
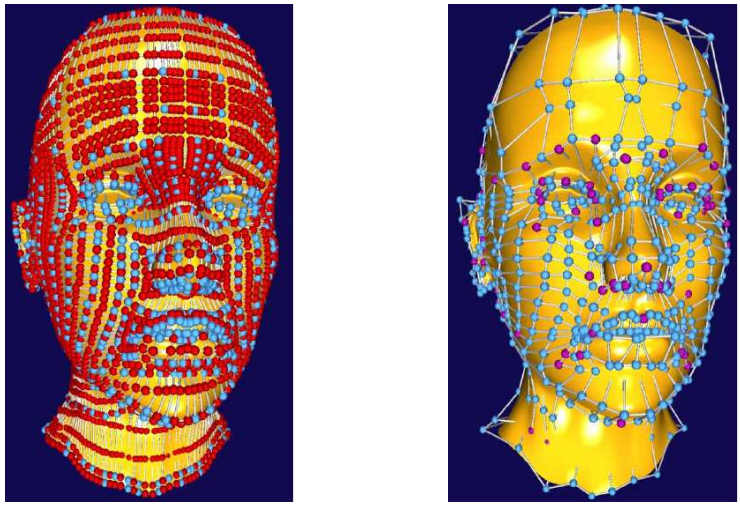



**Figure 29:** Pre-image of a T-mesh. (Image Courtesy of Sederberg et al. [129])

Hierarchical B-splines were also studied by Kraft [90]. He constructed a multilevel spline space which is a linear span of tensor product B-splines on different, hierarchically ordered grid levels. his basic idea is to provide a selection mechanism for B-splines which guarantees linear independence to form a basis. CHARMS [55] focuses on the space of basis functions in a similar way, but in a more general setting and hence with more applications. Weller and Hagen [151] studied spaces of piecewise polynomials with an irregular, locally refinable knot structure. They considered the domain partition with knot segments and knot rays in the tensor-product B-spline domain. Their approach is restricted to so-called ”semi-regular bases”.

In [129], Sederberg *et al.* present the T-spline, a generalization of the non-uniform B-spline surfaces. T-spline control grids need not to be totally regular. In particular, they allow *T-junctions* (the final control point in a partial row), and lines of control points need not to traverse the entire control grid (see Figure 29). Therefore, T-splines enable true local refinement without introducing additional, unnecessary control point in nearby regions. A serious weakness with NURBS models is that NURBS control points must lie topologically in a rectangular grid. This means



(a) NURBS (b) T-spline

**Figure 30:** Head modeled (a) as a NURBS with 4712 control points and (b) as a T-spline with 1109 control points. The red NURBS control points are superfluous (Image Courtesy of Sederberg et al. [128])

that typically, a large number of NURBS control points serve no purpose other than to satisfy topological constraints. They carry no significant geometric information. In Figure 30 (a), all the red NURBS control points are, in this sense, superfluous. Figure 30 (b) shows a T-spline control grid which was obtained by eliminating the superfluous control points from the NURBS model. Sederberg *et al.* also develop an algorithm to convert NURBS surfaces into T-spline surfaces, in which a large percentage of superfluous control points are eliminated [128].

## 3.5 Triangular Splines

While tensor products have proven themselves an excellent tool for the modeling of fairly regular surfaces, e.g., outer car bodies, they have well-known drawbacks if the modeling of largely irregular objects is required. In this section, we will give a brief introduction to the surfaces defined on triangles instead of rectangles.

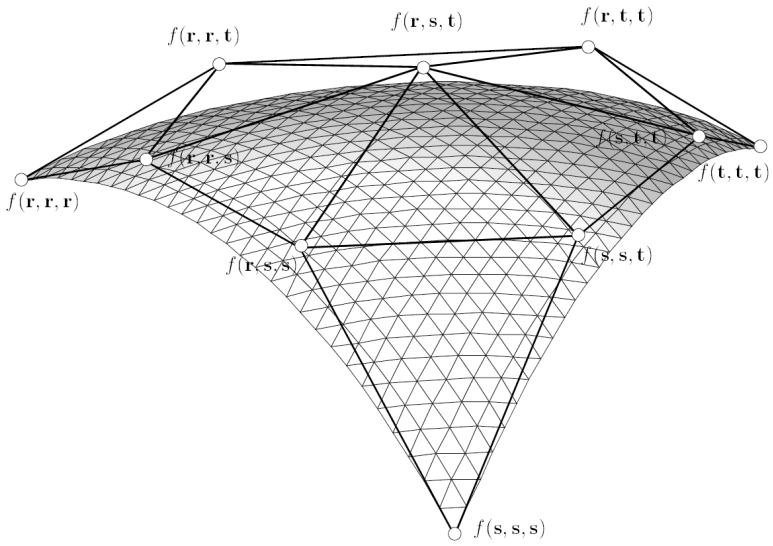



**Figure 31:** A cubic Bézier patch. (Image Courtesy of Seidel et al. [132])

### 3.5.1 Triangular Bézier Patches

Triangular Bézier patches (see Figure 31 for an example of cubic Bézier patch) have originally been introduced by de Casteljau using polar forms [132]: Consider a polynomial surface $F : \mathbb{R}^2 \to \mathbb{R}^t$. Suppose we wish to represent $F$ as a triangular Bézier patch over some given domain triangle $\Delta = \Delta(r,s,t)$. Representing $\mathbf{u} \in \mathbb{R}^2$ in barycentric coordinates w.r.t. $\Delta$,

$$\mathbf{u} = r(\mathbf{u})\mathbf{r} + s(\mathbf{u})\mathbf{s} + t(\mathbf{u})\mathbf{t}, r+s+t = 1,$$

we obtain

$$\begin{aligned} F(\mathbf{u}) &= f(\mathbf{u},\ldots,\mathbf{u}) \\ &= r(\mathbf{u})f(\mathbf{u},\ldots,\mathbf{u},\mathbf{r}) + s(\mathbf{u})f(\mathbf{u},\ldots,\mathbf{u},\mathbf{s}) + t(\mathbf{u})f(\mathbf{u},\ldots,\mathbf{u},\mathbf{t}) \\ &= \textstyle\sum_{i+j+k=n} B^{\Delta,n}_{ijk}(\mathbf{u}) f(\underbrace{\mathbf{r},\ldots,\mathbf{r}}_{i},\underbrace{\mathbf{s},\ldots,\mathbf{s}}_{j},\underbrace{\mathbf{t},\ldots,\mathbf{t}}_{k}) \end{aligned}$$

where

$$B^{\Delta,n}_{ijk}(\mathbf{u}) = \begin{pmatrix} n \\ ijk \end{pmatrix} r(\mathbf{u})^i s(\mathbf{u})^j t(\mathbf{u})^k$$

are the Bernstein polynomials w.r.t. $\Delta = \Delta(\mathbf{r},\mathbf{s},\mathbf{t})$. we have shown

**Theorem 14 (Bézier Points)** *Let $\Delta = \Delta(\mathbf{r},\mathbf{s},\mathbf{t})$ be an arbitrary triangle. Every polynomial $F : \mathbb{R}^2 \to \mathbb{R}^t$ can be represented as a Bézier triangle w.r.t. $\Delta$. The Bézier points are given as*

$$\mathbf{b}_{ijk} = f(\underbrace{\mathbf{r},\ldots,\mathbf{r}}_{i},\underbrace{\mathbf{s},\ldots,\mathbf{s}}_{j},\underbrace{\mathbf{t},\ldots,\mathbf{t}}_{k}),$$

*where $f$ is the polar form of $F$.*

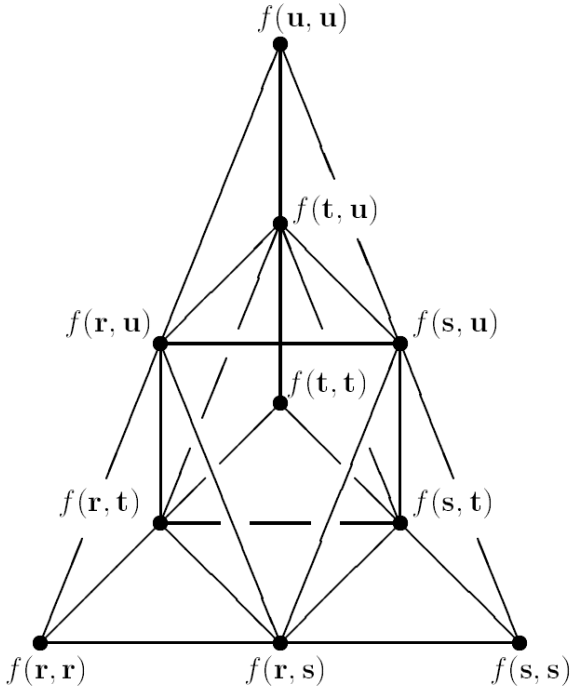



**Figure 32:** The de Casteljau Algorithm for a quadratic Bézier patch. (Image Courtesy of Seidel et al. [132])

Similar to the curve case, theorem 3.9 leads to the *de Casteljau Algorithm* for evaluation, subdivision, and computation of the polar form (see Figure 32 ). Finally, affine invariance also follows in exactly the same way as in the curve case.

Triangular Bézier patches are frequently used for interpolation. The Clough-Tocher interpolant [140] is a $C^1$-continuous piecewise cubic that interpolates to $C^1$-data at the vertices of the macro-triangle and to a given cross-boundary derivative along each edge. The split is obtained by splitting the macro-triangle into three

pieces along the centroid. The boundary Bézier points are determined by the interpolation conditions while the inner Bézier points are set to satisfy the desired $C^1$-conditions.

The Powell-Sabin interpolant [119] is a $C^1$-continous piecewise quadratic that interpolates to $C^1$-data at the vertices of the macro-triangle. In the generic case each macro triangle is split into six subtriangles by connecting every vertex and midpoint of an edge to the centroid. The Bézier points are set to interpolate the given data and to make the interpolant $C^1$-continuous throughout.

### 3.5.2 *B*-patches and Triangular *B*-Splines

The theoretical foundation of triangular *B*-splines lies in the multivariate *B*-spline, or simplex spline, introduced by de Boor [26]. Meanwhile, many researchers have tried to produce useful linear combinations of simplex splines sharing some of the properties of the univariate *B*-splines, in particular, the polynomial or piecewise polynomial reproduction property (see [21] for a survey in simplex splines). Dahmen and Micchelli [23, 24] and Höllig [78], using combinatorial arguments, propose convenient basis of simplex splines that reproduce polynomials of degree $n$. But the reproduction of $C^{n-1}$ piecewise polynomial functions on a given triangulation could not be settled.

Based on the blossom or polar form [122] and *B*-patch [130], Dahmen, Micchelli and Seidel [22] propose a general spline scheme in $s$-dimensional space, which constructs a collection of multivariate *B*-splines whose linear span comprises all polynomials of degree at most $n$. The bivariate case is called triangular *B*-spline or DMS spline. Due to its elegant construction and many attractive properties for geometric modeling, triangular *B*-spline has received much attention since its inception. Fong and Seidel [43] present the first prototype implementation of triangular *B*-splines and show several useful properties, such as affine invariance, convex hull, locality, and smoothness. Greiner and Seidel [54] show the practical feasibility of multivariate *B*-spline algorithms in graphics and shape design. Pfeifle and Seidel [115] demonstrate the fitting of a triangular *B*-spline surface to scattered functional data through the use of least squares and optimization techniques. Gormaz and Laurent study the piecewise polynomial reproduction of triangular *B*-spline and

give a direct and intuitive proof [51]. Pfeifle and Seidel [116] present scalar spherical triangular splines and demonstrate the use of these splines for approximating spherical scattered data. Franssen et al. [45] propose an efficient evaluation algorithm, which works for triangular *B*-spline surfaces of arbitrary degree. Recently, Neamtu [110] describes a new paradigm of bivariate simplex splines based on the higher degree Delaunay configurations. He and Qin [74] present a method to surface reconstruction using triangular B-splines with free knots. He et al. [71] present rational spherical spline for genus zero shape modeling. He et al. [70] present an efficient method to fair triangular *B*-spline surfaces of arbitrary topology.

Triangular *B*-spline surfaces can be defined on planar domains with arbitrary triangulations. In particular regions, triangular *B*-splines are *B*-patches. For the convenience, we introduce notations according to [59].which are similar to those employed in [22, 50]. Essentially, we formulate *B*-patches through the use of a polar form. Let $\Delta^I := [\mathbf{t}_0^I, \mathbf{t}_1^I, \mathbf{t}_2^I]$ be the triangle "*I*" of our triangulation $\mathcal{T}$ of $\mathbb{R}^2$. For each vertex $\mathbf{t}_i^I$ we assign a list of $k+1$ distinct additional knots

$$\mathbf{t}_i^I := \{\mathbf{t}_{i,0}^I, \mathbf{t}_{i,1}^I, \cdots, \mathbf{t}_{i,k}^I\}. \tag{3}$$

The rule proposed in [22] consists of producing a subset $V_\beta^I$, where $\beta = (\beta_0, \beta_1, \beta_2)$ are three non negative integers, as follows:

$$V_\beta^I := \{\mathbf{t}_{0,0}^I, \mathbf{t}_{0,1}^I, \cdots, \mathbf{t}_{0,\beta_0}^I, \mathbf{t}_{1,0}^I, \mathbf{t}_{1,1}^I, \cdots, \mathbf{t}_{1,\beta_1}^I, \mathbf{t}_{2,0}^I, \mathbf{t}_{2,1}^I, \cdots, \mathbf{t}_{2,\beta_2}^I\}.$$

If we want to define a degree $k$ simplex splines, we must impose that

$$|\beta| := \beta_0 + \beta_1 + \beta_2 = k.$$

$V_\beta^I$ is the set of all knots associated with one vertex in $\mathcal{T}$.

We further define $\Delta_\beta^I := [\mathbf{t}_{0,\beta_0}^I, \mathbf{t}_{1,\beta_1}^I, \mathbf{t}_{2,\beta_2}^I]$ and

$$X_\beta^I := (\mathbf{t}_{0,0}^I, \cdots, \mathbf{t}_{0,\beta_0-1}^I, \mathbf{t}_{1,0}^I, \cdots, \mathbf{t}_{1,\beta_1-1}^I, \mathbf{t}_{2,0}^I, \cdots, \mathbf{t}_{2,\beta_2-1}^I) \in (\mathbb{R}^2)^{|\beta|}. \tag{4}$$

$X_\beta^I$ is the set of knots associated with one control point $f(X_\beta^I)$.

If $\Delta_\beta^I$ is non-degenerate, it is possible to define the barycentric coordinates of $\mathbf{u} \in \mathbb{R}^2$ with respect to this triangle:

$$\mathbf{u} = \sum_{i=0}^{2} \lambda_{\beta,i}^I(\mathbf{u}) \mathbf{t}_{i,\beta_i}^I, \text{ and } \sum_{i=0}^{2} \lambda_{\beta,i}^I(\mathbf{u}) = 1. \tag{5}$$

The generalized algorithm computes $F(\mathbf{u})$ starting from the values $f(X^I_\beta)$, $|\beta| = k$. Those values are called the *poles* of $F$. Let us define

$$X^I_\beta \mathbf{u}^v := X^I_\beta \times (\underbrace{\mathbf{u}, \mathbf{u}, \cdots, \mathbf{u}}_{v}) \in (\mathbb{R}^2)^{|\beta|+v}$$

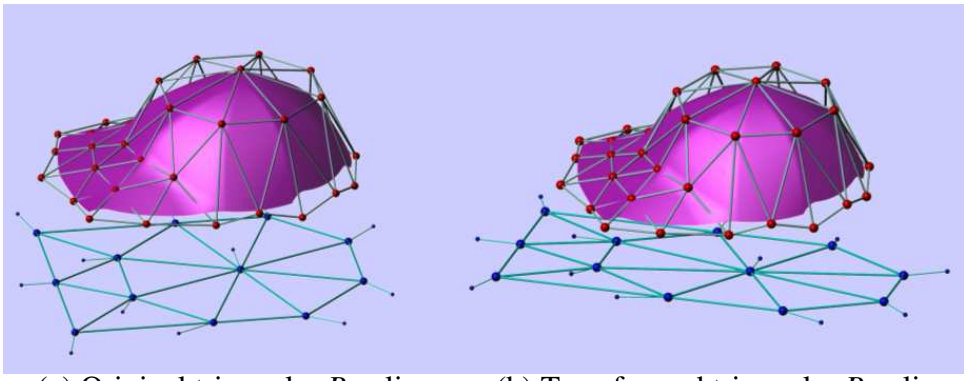
and assign $C^v_\beta(\mathbf{u}) := f(X^I_\beta \mathbf{u}^v)$ with $|\beta| = k - v$, the algorithm uses the $k$-affinity of $f$ stating the recurrence relation:

$$C^0_\beta(\mathbf{u}) := f(X^I_\beta), |\beta| = k$$

$$C^{v+1}_\beta(\mathbf{u}) := \sum_{i=0}^{2} \lambda^I_{\beta,i}(\mathbf{u}) C^v_{\beta+e^i}(\mathbf{u}), \tag{6}$$

where $e^i$ denotes the canonical basis vector. Then $F(\mathbf{u}) = C^k_0(\mathbf{u})$. If the basis function for the pole $f(X^I_\beta)$ is denoted as $B^I_\beta(\cdot)$, then we obtain

$$F(\mathbf{u}) = \sum_{|\beta|=k} f(X^I_\beta) B^I_\beta(\mathbf{u}).$$



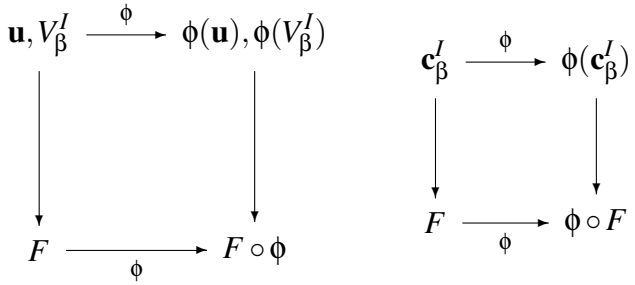
(a) Original triangular $B$-spline. (b) Transformed triangular $B$-spline.

**Figure 33:** Parametric affine invariance: (a) and (b) are two triangular $B$-splines sharing the same control net, the two parametric domains differ only by an affine transformation. The same control nets result in the same polynomial surfaces shown in (a) and (b). (Spline model courtesy of M. Franssen) [59].

Triangular $B$-splines have the following valuable properties which are critical for geometric and solid modeling:

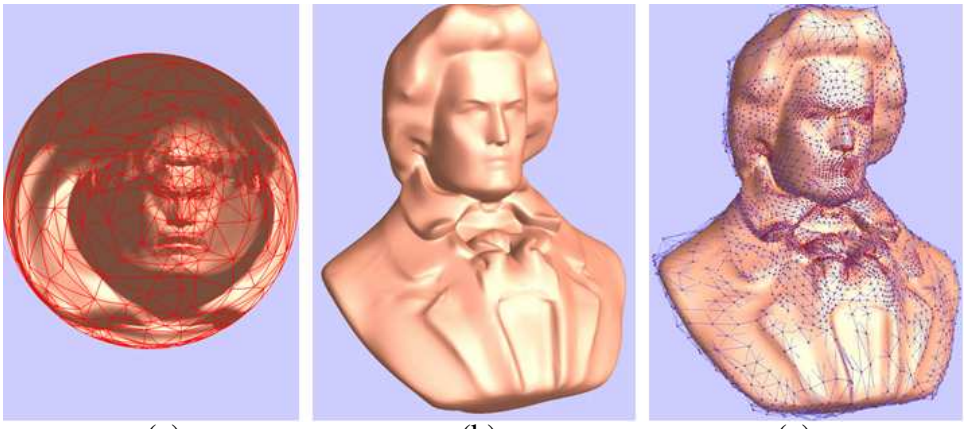
1. *Local support.* The spline surface has local support. In order to evaluate the image $F(\mathbf{u})$ of a point $\mathbf{u} \in \Delta^I$, we only need control points $\mathbf{c}_\beta^J$ (associated with knot set $V_\beta^J$ on triangle $J$), where triangle $J$ belongs to the 1-ring neighborhood of triangle $I$.
2. *Convex hull.* The polynomial surface is completely inside the convex hull of the control points.
3. *Completeness.* The $B$-spline basis is complete, namely, a set of degree $n$ $B$-spline basis can represent any polynomial with degree no greater than $n$ via a linear combination.
4. *Smoothness.* A degree $n$ $B$-spline surface is a piecewise polynomial of degree $n$ over the sub-triangulation induced by its knot net that is $C^{n-1}$-continuous everwhere if its knots are in general position.
5. *Parametric affine invariance.* The choice of parameter is not unique: if one transforms the parameter affinely and the corresponding knots of control points are transformed accordingly, then the polynomial surface remains unchanged (see Figure 33).
6. *Affine invariance.* If the control net is transformed affinely, the polynomial surface will be consistently transformed affinely.

Note that parametric affine invariance is different from affine invariance. The diagrams below illustrate the radical difference.




(a) Parametric affine invariance (b) Affine invariance

The left one above represents parametric affine invariance, which refers to the property that, under a transformation between parameter domains, the shape of the polynomial surface remains the same; the right one above indicates affine invariance, which refers to the property that under a transformation of the control points, the polynomial surface will change accordingly.



(a) (b) (c)

**Figure 34:** Modeling a genus zero surface using a single rational spherical spline. (a) Spherical parameterization and domain triangulation (1,022 spherical triangles); (b) A $C^2$ spherical spline; (c) Control net (4,601 control points) overlaid on the spline surface. (Image Courtesy of He et al. [71])

### 3.5.3 Spherical Triangular Spline

Conventional approaches for modeling a closed manifold surface with either regular tensor-product or triangular splines (defined over an open planar domain) require decomposing the acquired geometric data into a group of charts, mapping each chart to a planar parametric domain, fitting an open surface patch of certain degree to each chart, and finally, trimming the patches (if necessary) and stitching all of them together to form a closed manifold. In shape modeling, many objects are closed, genus zero surfaces, which are topologically equivalent to a sphere. In order to model a closed manifold surface without introducing degeneracy, one must define a network of tensor-product $B$-spline patches and maintain certain continuity (usually $G^1$ or $C^1$) between adjacent patches [35, 91]. This process requires many user interventions and is very labor intensive and un-intuitive. Therefore, it is necessary to devise a natural way to effectively and accurately model genus zero surfaces using splines without any cutting and patching work.

Based on the breakthrough work on spherical barycentric coordinates and spherical Bernstein-Bézier (SBB) polynomials by Alfeld, Neamtu and Schumaker [2], Pfeifle and Seidel [116] present scalar spherical triangular $B$-splines. These splines inherit many attractive properties from their planar counterpart (i.e., planar

triangular $B$-splines), such as the capability of representing any piecewise smooth surfaces of $C^{n-1}$ continuity and modeling the SBB polynomials as a special case. Furthermore, these spherical spline surfaces exhibit no degeneracies that frequently arise when attempting to employ planar parametric spline surfaces for modeling sphere-like, geometrically closed point clouds [116]. Because of the topological equivalence between spheres and other genus zero objects, spherical splines promise to be ideal for modeling closed genus zero surfaces both in theory and in practice.

However, the drawback of the spherical spline proposed by Pfeifle and Seidel is that it does not satisfy the convex hull property, because the partition of unity of the basis functions does not hold. Therefore, it may be difficult and less intuitive for ordinary users to interactively edit a spline surface by modifying its control net. To make spherical splines more accessible to a broader community and more useful in shape modeling applications, He et al. [71] present the *rational* spherical spline which inherits all the attractive properties of Pfeifle and Seidel's spline. More importantly, they offer the convex hull property because of the partition of unity of the rational basis functions. Figure 34 shows an example of modeling a genus zero surface with rational spherical splines.

## 3.6 Manifold Spline

In shape modeling, many objects are surfaces with complicated geometry and arbitrary topology. In order to model such shapes using planar splines, one must define a network of tensor-product $B$-spline patches and maintain certain continuity (usually $G^1$ or $C^1$) between adjacent patches [35, 91]. This process requires many user interventions and is labor intensive and un-intuitive. Therefore, it is necessary to devise a natural way to effectively and accurately model surfaces using splines without any cutting and patching work. In [59], a general theoretical and computational framework was presented to extend spline surfaces defined over planar domains to manifold domains with arbitrary topology with or without boundaries. They studied the affine structure of domain manifolds and proved that the existence of manifold splines is equivalent to the existence of a manifold's affine atlas. They also developed a set of practical algorithms to generalize triangular B-spline

surfaces [59, 72], the popular planar tensor-product NURBS and T-splines [75] to arbitrary manifold domain of any topological type.

### 3.6.1 Manifold Spline Theory and Algorithm

In this section, we will use the notion in [59] to introduce the new spline scheme - manifold splines.

#### 3.6.1.1 Manifold Spline Concepts

**Definition 15 (Affine Atlas)** *For affine structure $(G,X)$, $X$ is $R^2$, and $G$ is the group of affine transformations. A 2 dimensional manifold $M$ with an atlas $(U_\alpha, \phi_\alpha)$, if all chart transition functions*

$$\phi_{\alpha\beta} := \phi_\beta \circ \phi_\alpha^{-1} : \phi_\alpha(U_\alpha \cap U_\beta) \to \phi_\beta(U_\alpha \cap U_\beta))$$

*are affine, then the atlas is called an affine atlas, and $M$ is called an affine manifold (see Figure 1).*

Two affine atlases are equivalent if their union is still an affine atlas. All the equivalent affine atlases form an *affine structure* of the manifold. For closed surfaces, only genus-one surfaces have affine structures, but all surfaces with boundaries have affine structures.

**Definition 16 (Manifold Spline)** *A manifold spline of degree k is a triple $(M,C,F)$ as shown in Figure 35, where $M$ is the domain manifold with an atlas. $F$ is a map representing the entire spline surface. The knots are defined on $M$ directly. $C$ is the control point set, each control point is associated with a set of knots, such that*

1. *On each chart of the atlas, the restriction of $F$ and $C$ is a spline surface patch.*
2. *The evaluation of $F$ is independent of the choice of the charts.*

The central issue of constructing manifold splines is that the atlas must satisfy some special properties in order to meet all the requirements for the evaluation independence of chart selection. Because the existing planar spline schemes are parametric affine invariant, this requires that all the chart transition functions are affine.

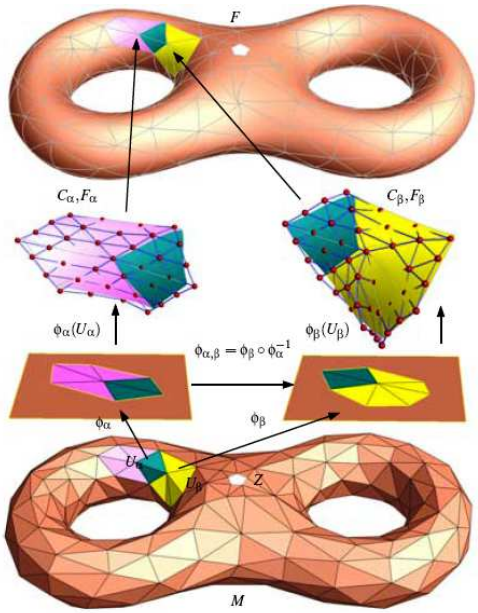



**Figure 35:** Key elements of manifold splines: The parametric domain $M$ is a triangular mesh with arbitrary topology as shown at the bottom. The polynomial spline surface $F$ is shown at the top. Two overlapping spline patches $(\phi_\alpha(U_\alpha), C(\alpha), F(\alpha))$ and $(\phi_\beta(U_\beta), C(\beta), F(\beta))$are magnified and highlighted in the middle. On each parameter chart $(U_\alpha, \phi_\alpha), (U_\beta, \phi_\beta)$ the surface is a triangular B-spline surface. For the overlapping part, its two planar domains differ only by an affine transformation $\phi_{\alpha\beta}$. The zero point neighbor is $Z$. (Image Courtesy of Gu et al. [59])

**Theorem 17** *The sufficient and necessary condition for a manifold M to admit manifold spline is that M is an affine manifold.*

In order to define a manifold spline, an affine atlas of the domain manifold must be found first. According to characteristic class theory [Milnor and Stasheff 1974], general closed 2-manifolds do not have an affine atlas. On the other hand, all open surfaces admit an affine atlas. In order to define manifold splines, the domain manifold has to be modified to admit an atlas by removing a finite number of points. This offers a theoretical evidence to the existence of singular points due to the topological obstruction. A classical result from characteristic class theory claims that the only closed surface admitting affine atlas is of genus one.

**Theorem 18** *(Existence) A domain manifold M admits a manifold spline scheme, if and only if M admits an affine structure, which holds for ALL the planar spline*

*schemes if they are parametric affine invariant.*

**Theorem 19** *(Closed affine manifold [107]) The closed affine 2-manifold must be of genus one.*

**Theorem 20** *Theorem 3. (Open affine manifold) Any oriented open surface admits an affine structure:*

1. *We can construct manifold splines if the domain is open,*
2. *If the domain manifold is closed and not a torus, then we must remove at least one point from the domain manifold to make it open.*

#### 3.6.1.2 Affine Structure Computation and Spline Construction

The key of constructing manifold splines is to compute the affine structure of the domain manifold. The paper [59] builds a connection between a manifold affine structure and its conformal structure with the following theorem:

**Theorem 21** *Suppose $M$ is a 2-manifold with a conformal atlas, $\omega$ is a holomorphic 1-form, then $\omega$ induces an affine structure of $M\backslash Z$, where $Z$ is the zero set of $\omega$ and $|Z|$ equals to the Euler number of $M$, i.e., $|2g-2|$ for manifold $M$ of genus $g$.*

Given a holomorphic 1-form $w$ on a surface $M$, assume its zero point set is $Z$; then, an affine atlas $A$ for $M\backslash Z$ can be constructed straightforwardly as illustrated by a genus two model in Figure 36.

1. Compute conformal structure of the domain manifold $M$.
2. Select one holomorphic 1-form, remove zero point neighbor.
3. Construct affine atlas by integrating the holomorphic 1-form.
4. Define knots on one chart, consistently extend to all charts.
5. Given one point on $M$, evaluate the spline surface on the point by choosing arbitrary chart which covers it by polar form.

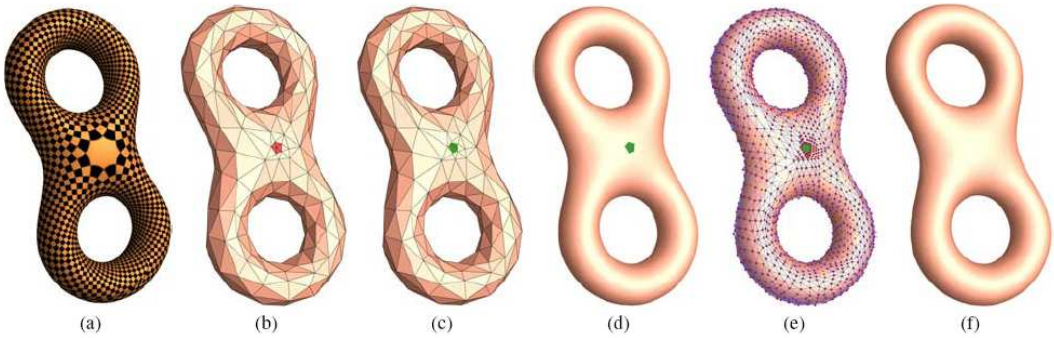



**Figure 36:** Construction of manifold spline: (a) Holomorphic 1-form $w$, the octagonal region indicates a singular point; (b) Domain manifold $M$; (c) Singular point removal $M \backslash Z$; (d) Manifold spline $F$; (e) Spline surface $F$ covered by control net $C$; (f) The regions of singular points are filled. (Image Courtesy of Gu et al. [59])

### 3.6.2 Manifold Interpolatory Splines

Converting point samples and/or triangular meshes to a more compact spline representation for arbitrarily topology is both desirable and necessary for computer vision and computer graphics. [73] presents a C1 manifold interpolatory spline that can exactly pass through all the vertices and interpolate their normals for data input of complicated topological type. Starting from the Powell- Sabin spline as a building block, they integrate the concepts of global parametrization, affine atlas, and splines defined over local, open domains to arrive at an elegant, easy-to-use spline solution for complicated datasets. The proposed global spline scheme enables the rapid surface reconstruction and facilitates the shape editing and analysis functionality.

### 3.6.3 Manifold T-spline

[75] presents the manifold T-splines, a natural and necessary integration of T-splines and manifold splines, with a goal to retain all the desirable properties while overcoming the aforementioned modeling drawbacks at the same time. The manifold T-splines naturally extend the concept and the currently available algorithms/techniques of the popular planar tensor-product NURBS and T-splines to

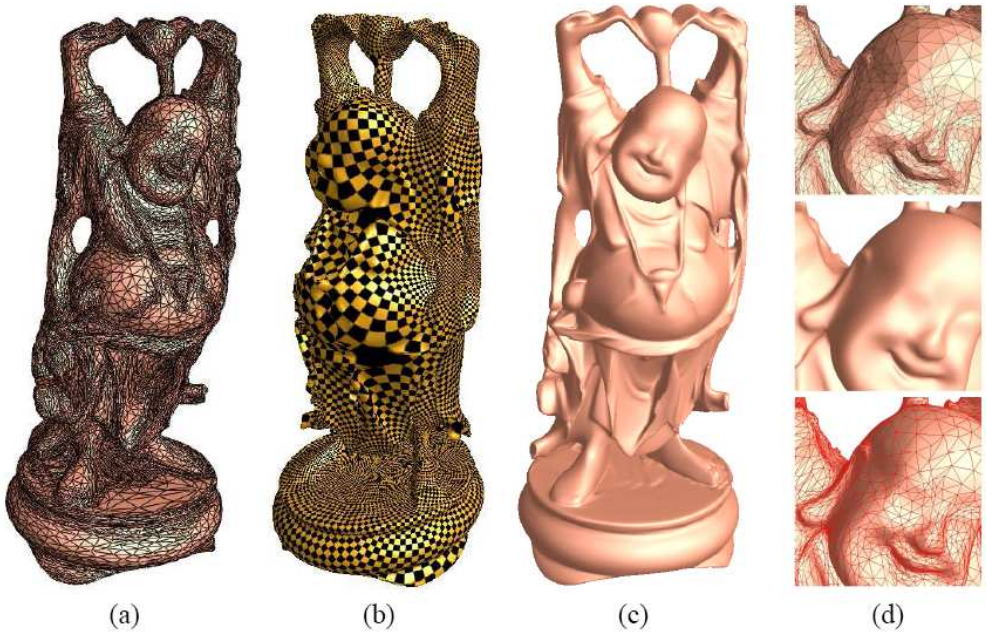



**Figure 37:** Globally interpolatory spline: (a) A genus-6 Buddha model with 25K vertices; (b) Global conformal parameterization; (c) A global $C^1$ spline surface which interpolates all the vertices and their normals of (a); (d) Close-up view: top, original mesh; middle, spline surface; bottom, spline surface with the red curves corresponding to the edges in the mesh. (Image Courtesy of He et al. [73])

arbitrary manifold domain of any topological type. The key idea is the global conformal parameterization that intuitively induces a tensor-product structure with a finite number of zero points, and hence offering a natural mechanism for generalizing the tensor-product splines throughout the entire manifold. In their shape modeling framework, the manifold T-splines are globally well-defined except at a finite number of extraordinary points, without the need of any tedious trimming and patching work. They also present an efficient algorithm to convert triangular meshes to manifold T-splines. Because of the natural, built-in hierarchy of T-splines, it is easy to reconstruct a manifold T-spline surface of high-quality with LOD control and hierarchical structure. Figure 38 shows an example of surface modeling using manifold T-spline.

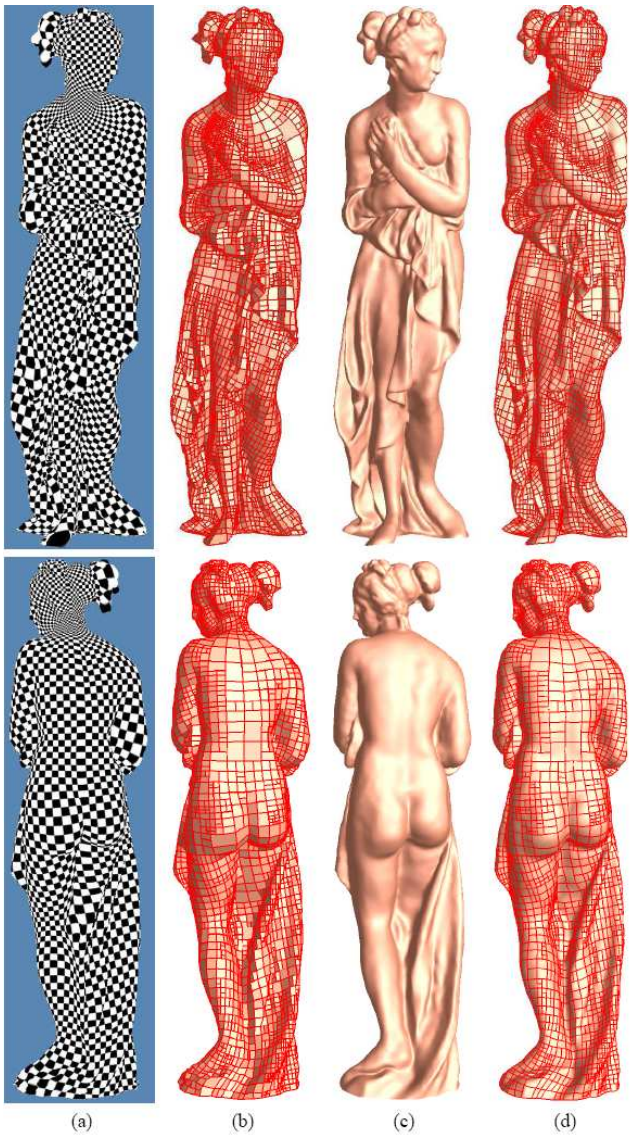



**Figure 38:** Modeling the Iphegenia model using manifold T-spline. (a) Global conformal parameterization; (b) The domain manifold; (c) A $C^2$ manifold T-spline with 9,907 control points; (d) The red curves are the images of the edges of the rectangles in the domain manifold. (Image Courtesy of He et al. [75])

## 3.7 Application

Spline have demonstrated their significance in various areas. In addition to the traditional reverse engineering and geometric design area in which they are usually used for scattered data fitting and surface design, they are also applied to scientific computation, visualization, manufacturing, etc.

Aliasing artifacts are a consequence of the errors introduced by attempting to represent a continuous model on a discrete device. McCool [106] describes a method whereby an image consisting of Gourad-shaded triangles can be represented by simplex splines; these can then be convolved with a box spline filter to form a set of prism splines representing the filtered image. This permits analytic filtering by filters which can be constructed from box spline basis functions, a special case being tensor product B-splines. Any further filtering for reconstruction is performed by digital post-processing.

Ming-Jun Lai *et al.* use bivariate and trivariate spline functions to solve 2D and 3D partial differential equations which can be further used for fluid flow simulation [92]. Some of advantages of spline solutions of PDE's are (1) piecewise polynomials of higher degrees can be used for numerical solution of PDE's very easily; (2) spline solutions of any smooth can be used for numerical solution very easily without constructing macro-elements; (3) Spline solution may be variable smoothness across the domain; (4) Spline solution can be obtained over arbitrary polygonal domains. They are able to solve many different PDE's including 2D and 3D Navier-Stokes equations.

Besides geometric modeling and processing, manifold spline [59, 72] can also serve as a general and powerful tool for scientific computation and engineering analysis, because of its many attractive numerical properties, including global one-patch representation, partition of unity, arbitrary triangulation of the domain manifold, local support, piecewise polynomial reproduction, robustness, stability and efficiency in evaluation, etc. Manifold splines are ideal tools for the global interpolation/approximation, and numerical solution of ordinary and partial differential equations, reverse engineering, rapid prototyping, and many more.

## 3.8 Comparison and Summary

In this section, we have reviewed the existing spline schemes, including the planar spline, spherical spline, and emerging manifold splines. We analyzed both their desired properties and limitations, also their applications in various areas. Refer to Table 1 for the comparison of the properties of popular spline schemes.

**Table 1:** Comparison of different spline schemes.

| | Convex hull | Local support | Polynomial reproduction | Affine invariance | Continuity |
|---|---|---|---|---|---|
| Bézier curve | √ | √ | √ | √ | $C^k$ |
| B-spline | √ | √ | √ | √ | $C^k$ |
| NURBS | √ | √ | √ | √ | $C^k$ |
| T-spline | √ | √ | √ | √ | $C^k$ |
| Triangular Bézier | √ | √ | √ | √ | $C^k$ |
| Powell-Sabin splines | √ | √ | √ | √ | $C^1$ |
| Triangular B-spline | √ | √ | √ | √ | $C^k$ |
| Non-rational spherical triangular B-spline | × | √ | √ | × | $C^k$ |
| Rational spherical triangular B-spline | √ | √ | √ | √ | $C^k$ |

# Chapter 4

# Polycube Splines

## 4.1 Introduction and Motivation

As we discussed in Chapter 1, discrete data inputs must be converted into continuous, compact representations in order to enable geometric design and downstream product development processes (e.g., accurate shape analysis, finite element simulation, and e-manufacturing). In order to model an arbitrary manifold in 3D, traditional planar spline schemes require segmenting and patching process which is primarily performed manually, and requires users' knowledge and skills, therefore laborious and error-prone for non-trivial topology and complicated geometry. The new emerging manifold splines extend the existing spline schemes defined over planar domains to any manifold domain of arbitrary topology using affine structures. Despite this earlier success, certain drawbacks of manifold splines still remain and demand more powerful modeling techniques. First of all, there must be singularities for any closed manifold except tori. The existence of singularities comes from the topological obstruction, which can not be avoided within the current manifold spline framework. [60] proposed a method to compute the affine structure with Euler number $|2-2g|$ singularities for a closed domain manifold of genus $g$. Although in theory singularity points are simply points without occupying any regions or areas, in practice "small" holes must be punched in order to enable the easy construction of manifold splines. Their earlier work makes no efforts to actually fill the "small" holes in the vicinity of extraordinary points. In addition, it is impossible to

specify the locations of all the singularities on the domain manifold given the fact that the number of singularities is actually fixed, but their positions are somehow globally related.

To overcome the above modeling and design difficulties and address the topological issue, we seek novel modeling techniques that would allow designers to directly define continuous spline models (especially tensor-product splines, which are the current industry standard) over any manifolds (serving as parametric domains). Such a global approach would have many modeling benefits, including no need of the transition from local patch definition to global surface construction via gluing, the elimination of non-intuitive segmentation and patching process, and ensuring the high-order continuity requirements. More importantly, we can expect a true "one-piece" representation for shapes of complicated topology, with a hope to automate the entire reverse engineering process (by converting points and/or polygonal meshes to spline surfaces with high accuracy) without human intervention.

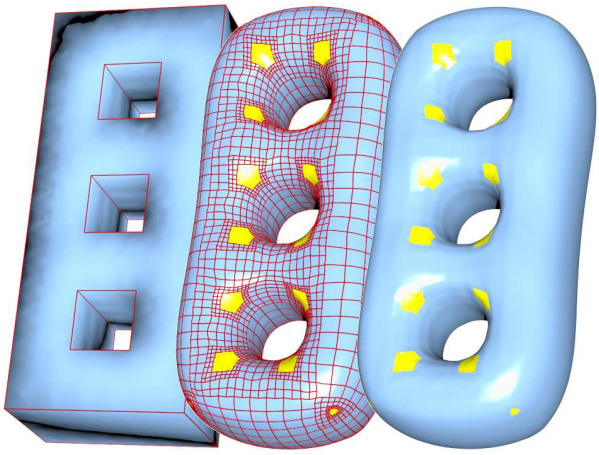


**Figure 39:** T-splines on polycubes. The polycube serves as the parametric domain which mimics the geometry of the 3D model. All the corners are singularities which are colored in yellow.

In this chapter, we forge ahead with our new research efforts by developing the *polycube splines*, with a goal to further improve the current state of knowledge for manifold splines. In a nutshell, our polycube splines can be considered as a novel variant of manifold splines with many new and attractive modeling properties. Unlike the previous manifold splines, the polycube splines are built directly

upon the polycube map, serving as its parametric domain. Because of its regularity, the polycube is now only covered by charts which are uniquely associated with faces and edges belonging to one of the cubes. As a result of the polycube map, all the corner points are now becoming singular. The key motivation for us to pursue the definition and construction of polycube splines is the fact that the polycube map offers a rectangular structure which for sure will facilitate geometric computing and shape analysis. Another main advantage of the polycube spline is that its parametric domain can mimic the geometry of any modeled objects in a topologically correct way, hence, it is much easier to isolate and control the position of the singularities. Furthermore, there are only three kinds of connectivity on the singularities, valence 3, 5 or 6, which can greatly simplify our procedures to handle extraordinary points. The polycube domain can be constructed to approximate the modeled geometry with better accuracy, but at the expense of more cubes and more charts. So, users will have freedom to control the complexity of the underlying parametric domain and place singularity points with great flexibility. Figure 39 demonstrates an example of our polycube splines. Similar to manifold splines, polycube splines also afford a general theoretic and engineering framework in which all the existing planar splines can be generalized to any polycube domain via affine structure. In this chapter, we develop algorithms to construct T-splines over polycubes and demonstrate their applications in shape modeling and reverse engineering in order to take advantage of the properties of partition-of-unity, level-of-detail control, and hierarchical representation. It may be noted that other powerful spline schemes, such as triangular $B$-splines, can be employed in a similar fashion.

The specific contributions of this work are as follows:

1. We present a systematic way to construct polycube maps for surfaces of arbitrary topology. Our method is fundamentally different from Tarini *et al.*'s technique [142] in that we do not need to compute the projection of the points from the 3D shape to the polycube, thus, the polycube can be flexibly constructed at any resolution and complexity.
2. We show that the introduced polycube maps naturally induce the affine structure by removing a finite number of corner points. Thus, polycube splines become a novel variant of manifold splines with many new and attractive

properties (outlined above). Taking advantage of the low area distortion between the domain manifold and the smooth spline surface (because polycubes can be built to approximate the modeled geometry within any user-specified accuracy), the polycube splines can be constructed easily and robustly by using simple and regular charts and isolating all the user-controllable singularity points.

3. Polycube splines offer a general framework in which any existing planar spline scheme can be generalized to a polycube domain via affine structure. Especially, in this chapter, we construct T-splines on polycubes and demonstrate the efficiency of polycube splines to model surfaces with high fidelity, while retaining the attractive properties of partition-of-unity, level-of-detail control, and hierarchical representation.

The remainder of this chapter is organized as follows. We introduce the theoretical foundation of our work in Section 4.2. Then in Section 4.3 we present the detailed algorithms for constructing the polycube map. Next, we show the hierarchical surface reconstruction in Section 4.4. Finally, we demonstrate the experimental results with statistics and performance data in Section 4.4.

## 4.2 Theoretical Foundation

### 4.2.1 Riemannian Uniformization Metric

Suppose a surface $S$ is embedded in $\mathbb{R}^3$, then it has a Riemannian metric, which is represented by its first fundamental form, induced from the Euclidean metric of $\mathbb{R}^3$, denoted by $\mathbf{g}$. Suppose $u : S \to \mathbb{R}$ is a scalar function defined on $S$, then it can be verified that $e^{2u}\mathbf{g}$ is another Riemannian metric on $S$, denoted by $\bar{\mathbf{g}}$. It can be proven that angles measured by $\mathbf{g}$ are equal to those measured by $\bar{\mathbf{g}}$. Therefore, $\bar{\mathbf{g}}$ is conformal to $\mathbf{g}$ and now $e^{2u}$ is called the conformal factor.

In essence, Riemannian metric determines the length, area, curvature and differential operators on $S$. When the Riemannian metric is conformally deformed, these geometric quantities will be changed accordingly. Suppose $\mathbf{g}$ is changed to $\bar{\mathbf{g}} = e^{2u}\mathbf{g}$. Then the Gaussian curvature will become

$$\bar{K} = e^{-2u}(-\Delta u + K), \tag{7}$$

where $\Delta$ is the Laplacian-Beltrami operator under the original metric $\mathbf{g}$. The geodesic curvature will become

$$\bar{k} = e^{-u}(\partial_{\mathbf{n}} u + k), \tag{8}$$

where $\mathbf{n}$ is the tangent vector orthogonal to the boundary. According to Gauss-Bonnet theorem, the total curvature is

$$\int_S K dA + \int_{\partial S} k ds = \int_{\bar{S}} \bar{K} d\bar{A} + \int_{\partial \bar{S}} \bar{k} d\bar{s} = 2\pi\chi(S), \tag{9}$$

where $\chi(S)$ is the Euler characteristic number of $S$ and $\partial S$ is the boundary of $S$.

Riemann uniformization theorem [84] states that for any surface $S$, there exists a unique conformal metric, such that it induces constant Gaussian curvature $\bar{K}$ and zero geodesic curvature $\bar{k}$.

$$\bar{K} = \begin{cases} +1, & \chi(S) > 0 \\ 0, & \chi(S) = 0 \\ -1, & \chi(S) < 0 \end{cases} \tag{10}$$

Such kind of metric is called the uniformization metric of $S$.

## 4.3 Construction of Polycube Maps

In this section, we explain in details our algorithm for constructing affine atlas using polycube maps for surfaces of arbitrary topology. The key difference between the techniques employed in [142] and ours in this work is that Tarini *et al.*'s technique is trying to find the one-to-one mapping of the 3D shape and polycube extrinsically, which typically requires the projection of points from one shape to the other. As a result, their method is usually quite difficult to handle cases where the two shapes differ too much and the point projection does not establish the one-to-one correspondence. In contrast, our method aims to compute such a mapping in an intrinsic way. We first conformally map the 3D shape and the polycube to the same canonical domains (e.g., sphere, Euclidean plane, or hyperbolic disk), then we construct a map between these two domains, which induces a one-to-one map between the 3D shape and the polycube. Since our method avoids the direct projection of

the 3D shape to the polycube, the polycube can be constructed independent of the actual geometry of 3D shape, allowing different complexity and resolution for the polycube.

### 4.3.1 Overview of the Algorithm

Constructing the polycube map is equivalent to seeking a bijective map between the 3D model and the polycube. Our method for establishing such a mapping varies according to different topologies of surfaces:genus zero surfaces, genus one surfaces, and surfaces of high genus. We compute the uniformization metric with heat flow method [63] for genus zero surfaces, holomorphic 1-form method [64, 83] for genus one surfaces, and hyperbolic Ricci flow method [82] for surfaces with genus greater than one.

In the followings, we use notations $M$ and $P$ to denote the 3D model and its polycube approximation (serving as the parametric domain), respectively.

The overall flow of our algorithm for establishing the one-to-one mapping can be summarized as follows:

1. Given a 3D model $M$ from data acquisition, construct a polycube $P$ which roughly resembles the geometry of $M$ and is of the same topology of $M$.
2. Compute the uniformization metric of $M$ and embed $M$ in the canonical domain $D_M$, which is a domain in $\mathbb{S}^2$, $\mathbb{E}^2$ or $\mathbb{H}^2$, i.e., $\phi_M : M \to D_M$.
3. Compute the uniformization metric of $P$ and embed $P$ in the canonical domain $D_P$, i.e., $\phi_P : P \to D_P$.
4. Construct the map $\phi_{D_M \to D_P} : D_M \to D_P$.
5. Finally, the composition $\phi_{M\to P} = \phi_P^{-1} \circ \phi_{D_M \to D_P} \circ \phi_M$ gives the desired polycube map from $M$ to $P$ as shown in equation (11).

$$\begin{array}{ccc} M & \xrightarrow{\phi_{M\to P}} & P \\ \downarrow{\scriptstyle \phi_M} & & \downarrow{\scriptstyle \phi_P} \\ D_M & \xrightarrow[\phi_{D_M \to D_P}]{} & D_P \end{array} \tag{11}$$

Note that, our construction method varies depending on different types of surfaces. Genus zero surfaces are mapped to the unit sphere $\mathbb{S}^2$ with positive curvature $\bar{K} = 1$. Genus one surfaces are mapped to Euclidean plane $\mathbb{E}^2$ with zero curvature $\bar{K} = 0$. Surfaces of high genus are mapped to hyperbolic disk $\mathbb{H}^2$ with negative curvature $\bar{K} = -1$.

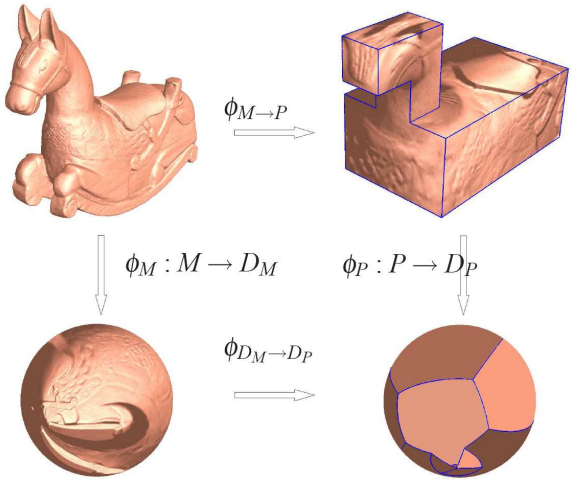



**Figure 40:** Conformal mapping of a genus zero surface to the unit sphere induces the genus zero conformal polycube map. Both the original mesh $M$ and the polycube $P$ are conformally mapped to the canonical domains, i.e., $\mathbb{S}^2$, $\mathbb{E}^2$ or $\mathbb{H}^2$. Denote these maps by $\phi_M : M \to D_M$ and $\phi_P : P \to D_P$. By finding the optimal map between $D_M$ and $D_P$, we get the polycube map $\phi_{M\to P} = \phi_P^{-1} \circ \phi_{D_M\to D_P} \circ \phi_M$.

### 4.3.2 Genus-zero Polycube Map

Genus zero surfaces are topologically equivalent to sphere. Thus, we use sphere as the canonical domain for both $M$ and $P$. We use the heat flow method to construct conformal maps between a closed genus zero surface and the unit sphere $\mathbb{S}^2$ [63]. The idea is that, for genus zero closed surfaces, conformal maps are equivalent to harmonic maps.

Let $\phi : M \to \mathbb{S}^2$ denote the spherical mapping. The harmonic energy is defined as

$$E(\phi) = \int_M < \nabla\phi, \nabla\phi > dA, \tag{12}$$

where $<,>$ is the inner product in $R^3$. The critical point of the harmonic energy is the harmonic map. Define the normal component of the Laplacian as

$$(\Delta\phi)^{\perp} =< \Delta\phi, \mathbf{n}\circ\phi > \mathbf{n}, \tag{13}$$

where $\mathbf{n}$ is the normal of $\phi(M)$. If $\phi$ is the harmonic map, then the tangent component of Laplace-Beltrami operator vanishes, i.e.,

$$\Delta\phi = (\Delta\phi)^{\perp}. \tag{14}$$

Therefore, we can diffuse $\phi$ to harmonic map by the heat flow method:

$$\frac{d\phi}{dt} = -(\Delta\phi - (\Delta\phi)^{\perp}). \tag{15}$$

After computing the maps $\phi_M : M \to \mathbb{S}^2$ and $\phi_P : P \to \mathbb{S}^2$, we need to find a map $\phi_{D_M \to D_P} : \mathbb{S}^2 \to \mathbb{S}^2$ which can align their major features. For example, we want to align the eyes and nose of the Isidore Horse model (see Figure 40) to be at certain positions on the polycube. To do so, we conformally map the sphere to the plane using stereographic projection

$$\tau : (x,y,z) \to (\frac{2x}{1-z}, \frac{2y}{1-z}),\ (x,y,z) \in \mathbb{S}^2. \tag{16}$$

We then use a special conformal map from the plane to itself, a Möbius transformation, to move three arbitrary feature points into any new desired positions. Suppose for the first surface, the three feature points are $z_0$, $z_1$ and $z_2$. We first construct the Möbius transformation which takes them into 0, 1, and $\infty$:

$$\psi_1 = \frac{(z-z_0)(z_1-z_2)}{(z-z_2)(z_1-z_0)}. \tag{17}$$

We then construct $\psi_2$ for three positions on $P$ in a similar way. Then $\psi_1^{-1} \circ \psi_2$ maps the feature points on the second surface into those on the first one. Finally, the conformal map $\phi_{D_M \to D_P} : \mathbb{S}^2 \to \mathbb{S}^2$ is defined as

$$\phi_{D_M \to D_P} = \tau^{-1} \circ \psi_2^{-1} \circ \psi_1 \circ \tau. \tag{18}$$

Note that the polycube map $\phi_{M \to P} = \phi_P^{-1} \circ \phi_{D_M \to D_P} \circ \phi_M$ is conformal since each sub-map is conformal.[1]

[1]Strictly speaking, the map $\phi_P : P \to \mathbb{S}^2$ is conformal everywhere except at the corners of the polycube.

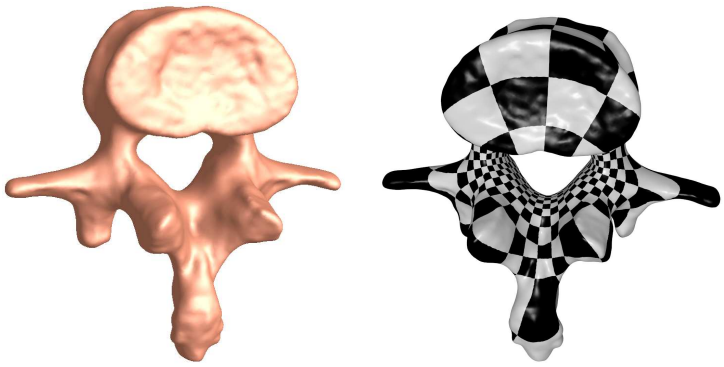


**Figure 41:** Holomorphic 1-form $\omega$ on genus one surface is well defined everywhere.

### 4.3.3 Genus-one Polycube Map

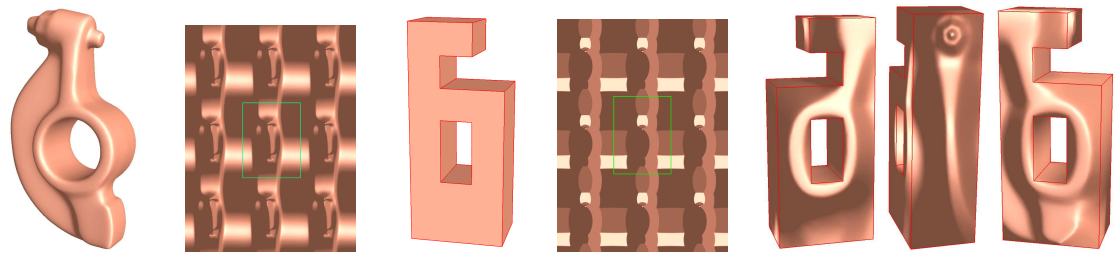
Suppose $M$ is a genus one closed surface, $\omega$ is a holomorphic 1-form. Then, $\omega$ is well-defined everywhere, i.e., there are no zero points as shown in Figure 41.

By integrating $\omega$, $M$ can be periodically mapped to the plane, each period is called a fundamental polygon. Each canonical fundamental polygon of genus one surface is a parallelogram. Given two arbitrary parallelograms, there exists a unique affine map to map one to the other, such that corners are mapped to corners, sides are mapped to sides.

The fundamental polygons of $M$ and $P$, $D_M$ and $D_P$, are parallelograms. Denote the unique affine map between them as $\phi_{D_M \to D_P}$, then the polycube map $\phi_{M \to P} : M \to P$ is formulated as

$$\phi_{M \to P} = \phi_P^{-1} \circ \phi_{D_M \to D_P} \circ \phi_M. \tag{19}$$

Figure 42 demonstrates the above mapping method for constructing a polycube map of the Rockerarm model. The polycube mesh is manually built. Then both the Rockerarm mesh and the polycube model are parameterized using the holomorphic 1-form method [64]. Their fundamental polygons are extracted and mapped by an affine map. The affine map further induces a bijective map between the Rockerarm model and the polycube.



(a) $M$ (b) $\phi_M$ : $M \to D_M \subset \mathbb{E}^2$ (c) $P$ (d) $\phi_P$ : $P \to D_P \subset \mathbb{E}^2$ (e) $\phi : M \to P$

**Figure 42:** Euclidean structure induces the genus-one polycube map. The genus one Rockerarm model $M$ in (a) is conformally mapped to the Euclidean plane in (b). The fundamental domain is a rectangle region enclosed by the green boundary in (b). Then, a polycube $P$ in (c) is also parameterized over the rectangular region in the same way in (d). By matching the two fundamental regions in (b) and (d) via an affine map, the conformal polycube map for the Rockerarm model is established (e).

### 4.3.4 High Genus Polycube Map

Given a high genus surface with simple geometry like the 3-hole torus model shown in Figure 39, the polycube map can be constructed using the techniques in [142]. However, for surfaces with complicated geometries like the model in Figure 44, the direct projection techniques in $\mathbb{R}^3$ hardly generate bijective maps. To avoid these difficulties, we use hyperbolic parameterization method instead.

#### 4.3.4.1 Hyperbolic Ricci flow

Hyperbolic Ricci flow is introduced in [82]. A circle packing on a mesh associates a circle with each vertex, circles intersect each other. A mesh with circle packing is denoted as $(M,\Gamma,\Phi)$, where $M$ represents the triangulation (connectivity) with vertex set V, edge set E and face set F, $\Gamma = \{\gamma_i, v_i \in V\}$ are the vertex radii and $\Phi = \{\phi_{ij}, e_{ij} \in E\}$ are the angles associated with each edge. A circle packing metric is define as $(M,\Phi,\Gamma)$. A *discrete conformal mapping* $\tau : (M,\Gamma,\Phi) \to (M,\bar{\Gamma},\Phi)$ solely changes the vertex radii $\Gamma$, but preserves the intersection angles $\Phi$.

Given the circle packing metric, the length $l_{ij}$ associated with the edge $e_{ij}$ is computed using the hyperbolic cosine law.

$$\cosh l_{ij}^2 = \cosh\gamma_i \cosh\gamma_j + \sinh\gamma_i \sinh\gamma_j \cos\phi_{ij}, \tag{20}$$

where $\phi_{ij}$ is the intersection angle between two circles associated at $v_i$ and $v_j$ with radius $\gamma_i$ and $\gamma_j$ respectively.

The discrete Gaussian curvature $K_i$ at an interior vertex $v_i$ with surrounding face $f_{ijk}$ is defined as

$$K_i = 2\pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, \qquad v_i \notin \partial M, \tag{21}$$

where $\theta_i^{jk}$ is the corner angle of $f_{ijk}$ at $v_i$. While the discrete Gaussian curvature for a boundary vertex $v_i$ is defined as

$$K_i = \pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, \qquad v_i \in \partial M. \tag{22}$$

Then the hyperbolic Ricci flow is defined as

$$\frac{\partial \gamma_i}{\partial t} = -\sinh \gamma_i K_i \tag{23}$$

It can be proven that discrete Ricci Flow is convergent to the uniformization metric and the convergence rate is exponential [16] [82].

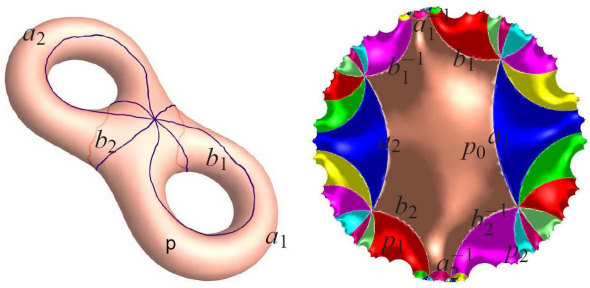



**Figure 43:** A genus two surface with a set of canonical fundamental group generators $\{a_1, b_1, a_2, b_2\}$ is shown on the left. A finite portion of its universal covering space is shown on the right. Different fundamental domains are drawn in different colors. The boundary of each fundamental domain is the preimage of $a_1 b_1 a_1^{-1} b_1^{-1} a_2 b_2 a_2^{-1} b_2^{-1}$. The points $\{p_0, p_1, p_2\}$ are the primages of $p$ on the surface.

#### 4.3.4.2 Hyperbolic embedding

With the uniformization metric, $M$ with $g > 1$ can be periodically mapped onto the hyperbolic space $\mathbb{H}^2$. We use the Poincaré hyperbolic disk model to represent

the hyperbolic space $\mathbb{H}^2$. The Poincaré disk is a two-dimensional space defined in the unit disk $\{z \in \mathbb{C} : |z| < 1\}$ on the complex plane $\mathbb{C}$ with hyperbolic metric. The hyperbolic metric is defined as

$$ds^2 = \frac{dzd\bar{z}}{(1-\bar{z}z)^2}. \tag{24}$$

The geodesic (hyperbolic lines) in the Poincaré disk are Euclidean circular arcs perpendicular to the boundary $|z| = 1$. The rigid motions in the hyperbolic plane are the Möbius transformations $z \to w$, $z \in \mathbb{C}$ with the form

$$w = e^{i\theta}\frac{z - z_0}{1 - \bar{z}_0}, \tag{25}$$

where $z_0$ is an arbitrary point inside the unit disk.

To embed $M$ into Poincaré disk, we need to compute the canonical homology basis, which is a set of $2g$ curves $\{a_1, b_1, a_2, b_2, \ldots, a_g, b_g\}$ satisfying the following criteria:

1. All the curves meet at a single base point, $v$.
2. Each pair of curves $\{a_i, b_i\}$ algebraically intersect each other exactly once.
3. No curve in another pair $\{a_j, b_j\}$ algebraically intersects either of $a_i, b_i$.

We slice the mesh $M$ along $\{a_i, b_i\}_{i=1}^{g}$ to form the fundamental domain $D$ whose boundary $\partial D$ is

$$\partial D = a_1 b_1 a_1^{-1} b_1^{-1} \cdots a_g b_g a_g^{-1} b_g^{-1}.$$

Then the canonical homology basis are mapped to geodesics on the Poincaré disk. Figure 43 illustrates the canonical homology basis and the hyperbolic embedding with the uniformization metric for a genus 2 model.

#### 4.3.4.3 Constructing the polycube map

In order to find the map between $M$ and $P$, we compute their hyperbolic parameterizations by solving the discrete hyperbolic Ricci flow in (23). Then, similar to the genus zero case, a harmonic map $\phi_{D_M \to D_P}$ is constructed such that maps the fundamental polygon of $M$ to the fundamental polygon of $P$. Finally, the polycube map is constructed as

$$\phi_{M \to P} = \phi_P^{-1} \circ \phi_{D_M \to D_P} \circ \phi_M. \tag{26}$$

Figure 44 demonstrates the example of polycube map for a genus-3 surface and highlight our construction pipeline.

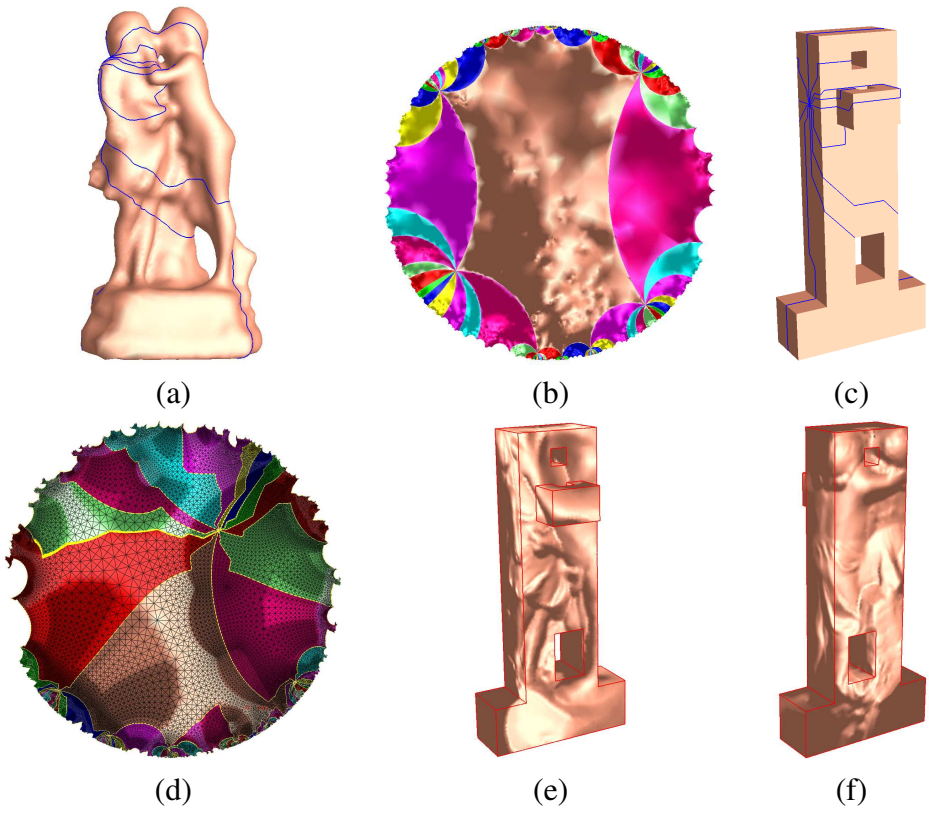
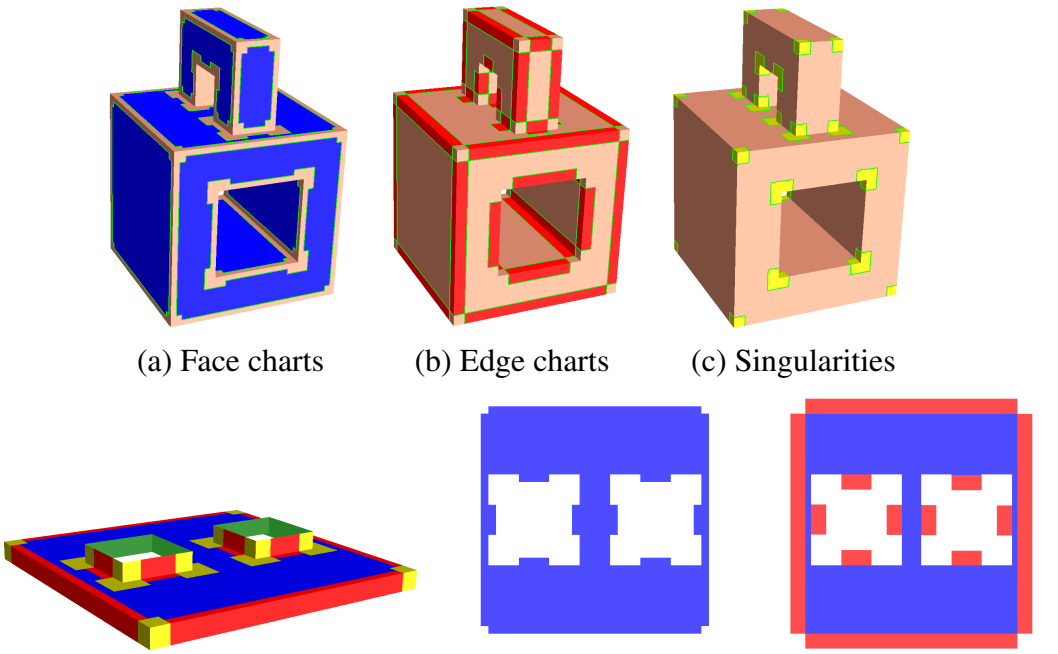



**Figure 44:** Hyperbolic structures induce the high genus polycube map. The canonical homology basis of the genus-3 sculpture model are colored in blue in (a). (b) shows the isometric embedding of its universal covering space on the Poincaré hyperbolic disk. We compute the hyperbolic uniformization metric of the polycube in (c) using a similar approach. The canonical homology basis of the polycube are drawn in blue in (c), (d) shows the isometric embedding of its universal covering space on the Poincaré hyperbolic disk. By establishing the correspondence between the fundamental domains, we construct the polycube map (shown in (e) and (f)) between (a) and (c).

### 4.3.5 The Affine Atlas via Polycube Map

We construct an affine atlas from the polycube map. Each face and edge on the polycube are associated with its own local chart. Each face chart covers only interior points of corresponding face and leaves off all the edges of the face. Each edge



(a) Face charts (b) Edge charts (c) Singularities

(d) Close-up view of one face chart and its associated edge charts and singularities

(e) Face chart

(f) Transition function

**Figure 45:** Polycube map induces affine structure. The polycube is covered by face and edge charts. Each face chart (drawn in blue) covers only interior points of corresponding face and leaves off all the edges of the face. Each edge chart (drawn in red) covers interior points of the edges but leaves off corner vertices. The corners (drawn in yellow) are singularities which are NOT covered by any charts. We highlight one face chart and its associated edge charts and singularities in (d). By flattening the edge charts, we get the planar domain shown in (e). Note that the transition functions between overlapped edge and face charts are simply translations and rotations (f). Therefore, by removing all the corners, the open polycube $P \setminus C$ has the affine structure.

chart covers interior points of the edge but leaves off corner vertices. Furthermore, there are overlaps between face charts and edge charts. The transition functions between overlapped edge and face charts are simply translations and rotations of 90 degrees. Note that there is **NO** vertex chart for the corner vertex, i.e., the corners are singular points. Therefore, by removing all the corners, polycube map naturally induces the affine structure. Figure 45 highlights face and edge charts of a polycube. The extraordinary points are colored in yellow.

In [60], they have pointed out that any planar spline schemes which satisfy the parametric affine invariant property can be generalized to manifold domain via affine structure. By removing all the corner points, a polycube domain is just an affine manifold preserving the affine structure. Therefore, we can define spline surfaces on polycube directly.

## 4.4 Hierarchical Surface Reconstruction Using Polycube T-Splines

After constructing the domain manifold and affine atlas of the original model by computing the polycube maps (section 4.3), we are now ready to generalize T-spline from planar domains to manifold domains via affine structure. This will enable the automatic reverse engineering from polygonal models initially acquired to a more compact spline representation with high accuracy.

### 4.4.1 T-Splines via Polycube Maps

The key advantage for defining T-spline over polycube maps is that each face chart of the polycube is nothing more than a union of rectangles, conventional tensor-product splines are special cases of T-splines, and they are all naturally defined over rectangular regions. More importantly, the hierarchical definition and level-of-detail control are attractive features in practice.

Recall that for every control point in the T-mesh, the covering region of its basis function is a rectangle, whose side lengths (knot vectors) are determined by the connectivity of the T-mesh. In polycube T-splines, we follow the rules defined

in [128,129]. We further require that on each chart, the basis functions vanishes outside the boundary of the chart. Thus, the face charts are totally separate from each other. Each edge chart connects two face charts (one face chart if it is a boundary edge and not shared by two faces). Therefore, given an arbitrary parameter $\mathbf{u} \in P \setminus C$, it may be covered by a single face chart, or a single edge chart, or by one face chart and one edge chart.

On each (edge and face) chart $(U_i, \phi_i)$, the spline patch is defined as a point-based spline whose control points form a T-mesh:

$$\mathbf{F}_i(\mathbf{u}) = \sum_j \mathbf{c}_j B_j(\phi_i(\mathbf{u})), \mathbf{u} \in U_i, \tag{27}$$

where $\mathbf{c}_j \in \mathbb{R}^3$ are the control points.

Given an arbitrary parameter $\mathbf{u} \in P \setminus C$, the spline evaluation can be carried out as follows:

1. Find the set of charts which cover this point $\mathbf{u}$. This set $V$ contains one face chart, or one edge chart, or one face chart and one edge chart.
2. The function value is the partition of unity of the spline patches in the above chart(s), i.e.,
$$\mathbf{F}(\mathbf{u}) = \frac{\sum_{i \in V} \sum_j \mathbf{c}_j B_j(\phi_i(\mathbf{u}))}{\sum_{i \in V} \sum_j B_j(\phi_i(\mathbf{u}))}.$$

### 4.4.2 Least-Square Fitting and Hierarchical Refinement

We now discuss the problem of finding a good approximation of a given polygonal mesh $S$ with vertices $\{\mathbf{p}_i\}_{i=1}^m$ by a manifold T-spline. We assume that the polygonal mesh $S$ has been normalized to be inside the unit cube centered at origin. A commonly-used technology is to minimize a linear combination of interpolation and fairness functionals, i.e.,

$$\min E = E_{dist} + \lambda E_{fair}. \tag{28}$$

The first part is

$$E_{dist} = \sum_{i=1}^m \|\mathbf{F}(\mathbf{u}_i) - \mathbf{p}_i\|^2,$$

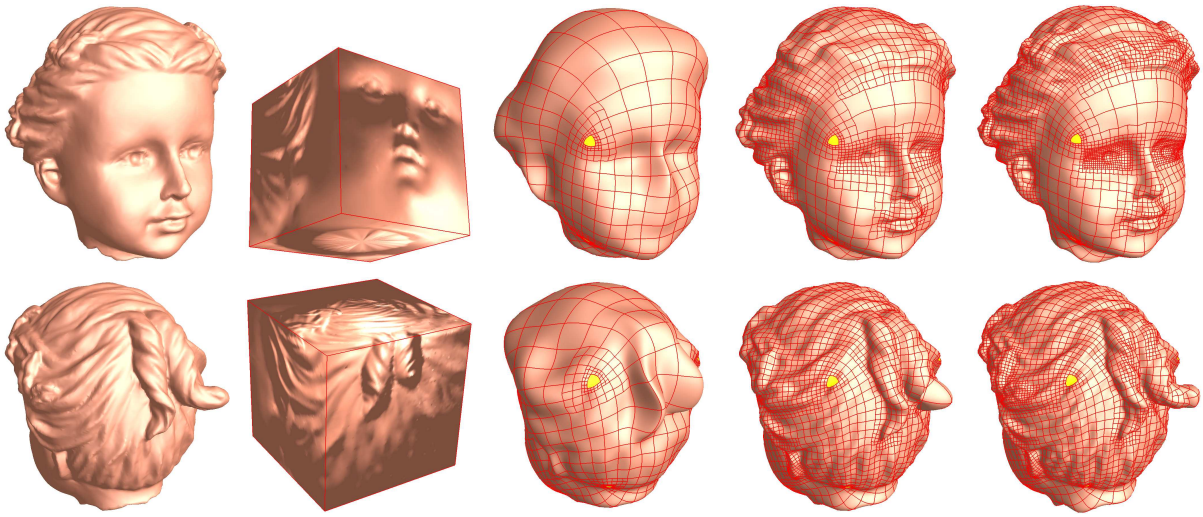



**Figure 46:** Hierarchical surface reconstruction of Polycube T-splines. $N_c^i$ and $L_\infty^i$ are the number of control points and maximal fitting error in iteration $i$. $N_v$ is the number vertices in the input polygonal mesh $S$. The input data is normalized to a unit cube.

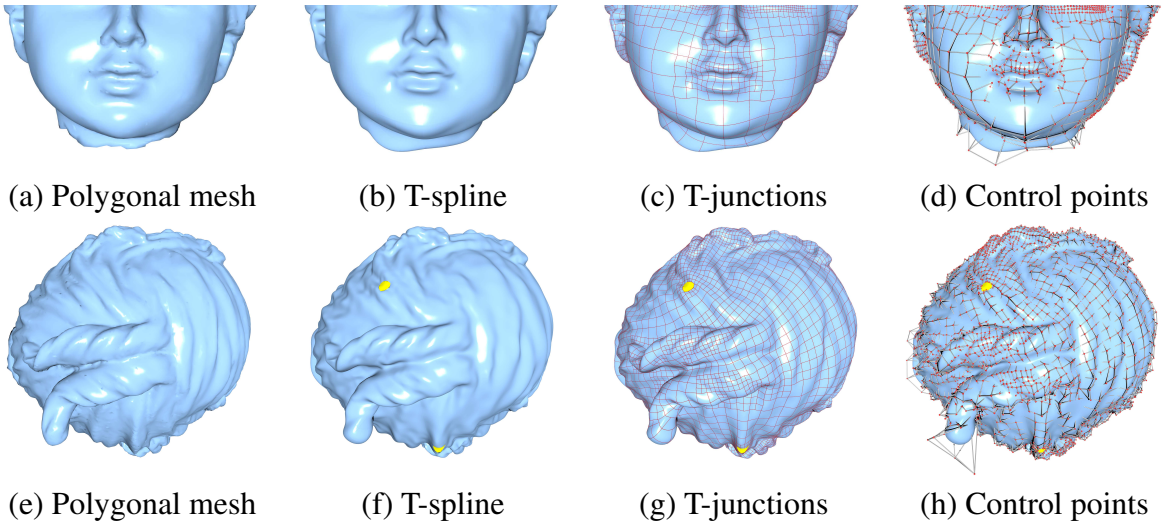



**Figure 47:** Close-up views of the reconstructed details. Our hierarchical surface reconstruction algorithm can faithfully reconstruct the details in the original model. (a) and (e) show the original polygonal model. (b) and (f) show the T-spline surfaces of $C^2$ continuity. (c) and (g) highlight the T-junctions on the spline surfaces. (d) and (h) show the splines overlaid with the control points.

where $\mathbf{u}_i \in M$ is the parameter for $\mathbf{p}_i$, $i = 1, \ldots, m$.
The second part $E_{fair}$ in (28) is a smoothing term. A frequently-used example is the thin-plate energy,

$$E_{fair} = \iint_M (\mathbf{F}_{uu}^2 + 2\mathbf{F}_{uv}^2 + \mathbf{F}_{vv}^2) du dv.$$

Note that both parts are quadratic functions of the unknown control points, leading to a linear system.

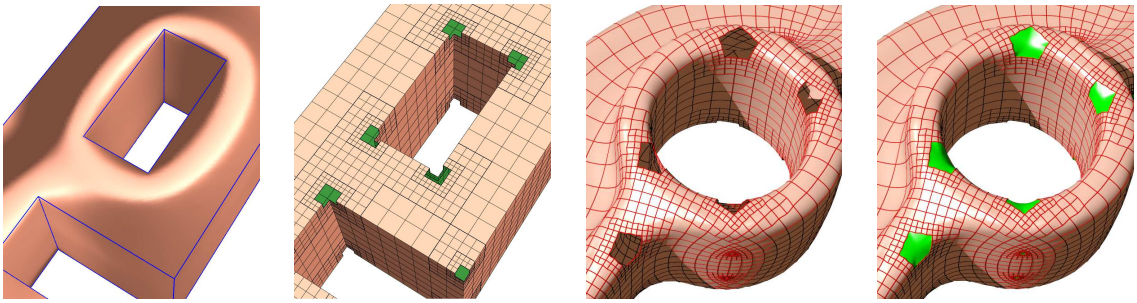
We solve Equation (28) for unknown control points using Conjugate Gradient method. The value and gradient of the interpolation functional and fairness functional can be computed straightforwardly.

In our method, we control the quality of the manifold T-spline spline by specifying the maximal fitting tolerance $L_\infty = \max \|\mathbf{F}(\mathbf{u}_i) - \mathbf{p}_i\|$, $i = 1, \ldots, m$. If the current surface does not satisfy this criterion, we employ adaptive refinement to introduce new degrees of freedom into the surface representation to improve the fitting quality. Because of the natural and elegant hierarchial structure of T-splines, this step can be done easily. Suppose a domain rectangle $I$ violates the criterion and denote $L_\infty^I$ the $L_\infty$ error on rectangle $I$. If the $L_\infty^I > 2\varepsilon$, split the rectangle $I$ using 1-to-4 scheme; Otherwise, we divide $I$ into two rectangles by splitting the longest edge. After adaptive refinement, we then re-calculate the control points until the maximal fitting tolerance is satisfied.

Figure 46 shows the whole procedure of hierarchical fitting of T-splines. For example, the initial spline of the Head model(Figure 46) contains only 1218 control points and the maximal error $L_\infty = 5.8\%$. Through five iterations, we can obtain a much more refined spline surface with 6475 control points by inserting only necessary control points. The maximal fitting error reduces to 0.54%. As shown in the close-up view (Figure 47), our hierarchical data fitting procedure can produce high quality polycube T-splines with high-fidelity and we will be able to recover all the surface details.

### 4.4.3 Handling the Extraordinary Points

In [60], Gu *et al.* proved that manifold splines *MUST* have singularities if the domain manifold is closed and not a torus. The number of extraordinary points



(a) (b) (c) (d)

**Figure 48:** Handing the extraordinary points of the manifold T-spline whose affine atlas is constructed using polycube maps, where all the corners are extraordinary points (shown in (a)). (b) shows the domain manifold after removing all the corners. (c) shows the open manifold T-spline surface with many holes. For each hole, we construct a cubic triangular *B*-spline surface which minimizes the thin-plate energy functional (Equation 30) and satisfies the boundary condition. (d) shows the final result after hole-filling (hole areas are all colored in green).

of the domain manifold via conformal structures and polycube maps are different. Given the surface $M$ with genus $g$ and $b$ boundaries, the number of zero points of the holomorphic 1-form is fixed, i.e., $|2g-2+b|$. Using polycube maps, the number of extraordinary points depends on the geometry of the polycube, i.e., each corner is a singularity.

Although the singularities are just points on the domain manifold, in practice, we have to remove these points and their 1-ring or 2-ring neighbors. As a result, the holes are unavoidably in the spline surface. Thus, we need to find a blending surface patch to fill the holes smoothly. In our implementation, we use a cubic triangular spline to fill each hole such that the surface is $C^2$ inside and $G^1$ along the boundaries of the hole. The reason that we choose triangular *B*-spline [74] is its flexibility in the domain construction and its potential to match with any number of sides of holes.

Thus, our goal is to solve the following optimization problem:

$$E(s) = \iint_{\Omega} (\frac{\partial^2 s}{\partial u^2})^2 + 2(\frac{\partial^2 s}{\partial u \partial v})^2 + (\frac{\partial^2 s}{\partial v^2})^2 du dv. \tag{29}$$

where $s$ is the triangular *B*-spline surface, and $\Omega$ is the parametric domain of $s$. Our

strategy to fill the hole is to find $s$ solving the following minimization problem:

$$min\{E(s) : s|_{\partial\Omega} = f,\ \frac{\partial s}{\partial u} \times \frac{\partial s}{\partial v}|_{\partial\Omega} = n\}. \tag{30}$$

where $f$ and $n$ are the boundary positions and normals.

The boundary conditions are represented by several sampling points on the boundary of the spline surface. The boundary position constraints naturally lead to a system of linear equations on the control points. The normal constraints are expressed as

$$< \frac{\partial s}{\partial u}, n >= 0,\ < \frac{\partial s}{\partial v}, n >= 0.$$

Therefore, Equation (30) is a linear least-square problem with linear constraints, which can be solved easily using Lagrange Multiplier method. Figure 48 demonstrates the procedure pipeline to handle the extraordinary points on the Rocker Arm model.

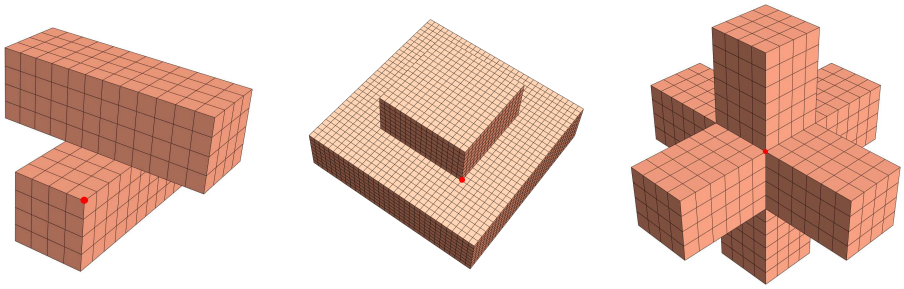


**Figure 49:** Extraordinary points (marked in red) with valence 3, 5, 6.

### 4.4.4 Discussions

This subsection compares the T-splines constructed using conformal structure [75] and polycube map, respectively. From the chart-relation's point of view, these methods differ in three aspects, the number and the locations of singularities, the angle/area distortion, and the type of transition functions. Each method has its own merits and users may choose one or another depending on their specific application needs. Table 2 summaries the salient differences between these methods.

**Table 2:** Comparison of the methods to compute affine structures. $g$, genus of the domain manifold $M$; $b$, number of boundaries of $M$.

| Method | # of singularities | Location of singularities | Area distortion | Angle distortion | Transition function |
|---|---|---|---|---|---|
| Conformal structure | $\|2g-2+b\|$ | difficult to control | large on extruding parts | no | translation |
| Polycube map | many | easy to control | low | low | translation and 90 degree rotation |

Conformal structure induces the affine structure with the fixed number of extraordinary points, i.e., $|2g-2+b|$. For genus-zero surfaces, we usually intentionally cut two boundaries on the model. Note that, although we do not modify the geometry of the original model, the number of extraordinary points drop to zero. Although conformal structure preserves the angles very well, they inevitably introduce large area distortion if the model has some long, extruding parts, e.g., the tail and feet of the bird model. These large area distortions usually make the spline construction very difficult, since we need to introduce more control points in such areas. The transition functions of the affine atlas via conformal structure is simply the translations, which facilitates the implementation of T-splines on manifolds. The valence of extraordinary points of T-splines via conformal structure is eight, i.e., the hole is sixteen-sided.

Polycube maps are ideal to reduce both the area and angle distortion in the affine atlas, as shown in the 3-hole torus models in Figure 39. Thus, it facilitates the spline construction procedures. However, the side-effect to reduce the area distortion is to introduce more extraordinary points simultaneously. Usually, the lesser the area distortion, the more number of extraordinary points. The transition functions of the affine atlas via polycube maps is the composition of translation and 90 degree rotation. The valence of singularities of T-spline via polycube map is three, five, or six, thus, the hole is six, ten or twelve-sided (see Figure 49).

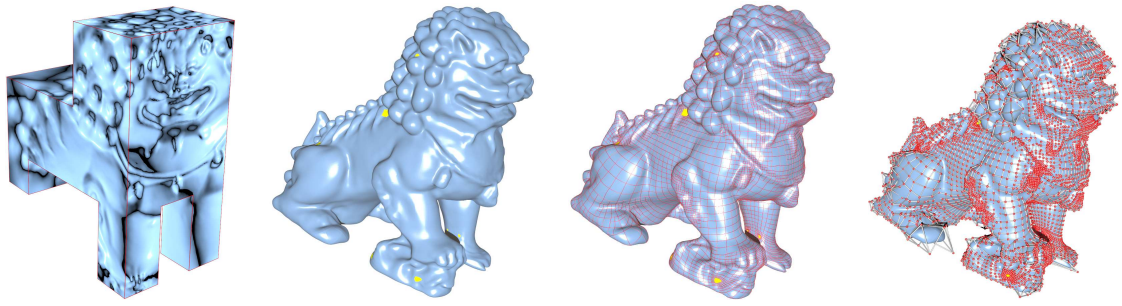
### 4.4.5 Experimental Results

Our prototype system is implemented in C++ on a MS Windows XP PC with dual Intel Xeon 2.6GHz CPUs and 2GB RAM. We built a complete system for computing the conformal structures, the polycube maps and T-splines. We tested our

algorithms on various models from genus zero to genus three. The statistics of the test cases are shown in Table 3. Figure 46 illustrates the hierarchical surface reconstruction. As shown in Figure 46 and 47, we can get high-quality spline surfaces by gradually increasing the number of control points. More complicated models are shown in Figure 50 and Figure 51. The results demonstrate both the theoretic rigor and feasibility in practice for methodologies and computational techniques.

**Table 3:** Statistics of test examples (after 5 iterations). $N_s$, # of singularities; $N_v$, # of vertices in the input polygonal mesh; $N_c$, # of control points; *rms*, root-mean-square error; $L_\infty$, maximal fitting error.

| Object | genus | $N_s$ | $N_v$ | $N_c$ | *rms* | $L_\infty$ |
|---|---|---|---|---|---|---|
| Head (Figure 46, 47) | 0 | 8 | 31$K$ | 6475 | 0.05% | 0.54% |
| Bimba (Figure 51) | 0 | 16 | 98$K$ | 10964 | 0.07% | 0.62% |
| Buddha (Figure 51) | 0 | 16 | 120$K$ | 11067 | 0.04% | 0.55% |
| Rockerarm (Figure 51) | 1 | 24 | 51$K$ | 4132 | 0.03% | 0.32% |
| 3-hole Torus (Figure 39) | 3 | 32 | 48$K$ | 5180 | 0.02% | 0.22% |
| Chinese Dragon (Figure 50) | 0 | 28 | 100$K$ | 11335 | 0.07% | 0.61% |
| Ramesses (Figure 51) | 0 | 24 | 115$K$ | 9874 | 0.04% | 0.59% |



(a) Polycube map (b) T-spline (c) T-junctions (d) Control points

**Figure 50:** Polycube T-splines for the Chinese Dragon model.

## 4.5 Performance Discussion

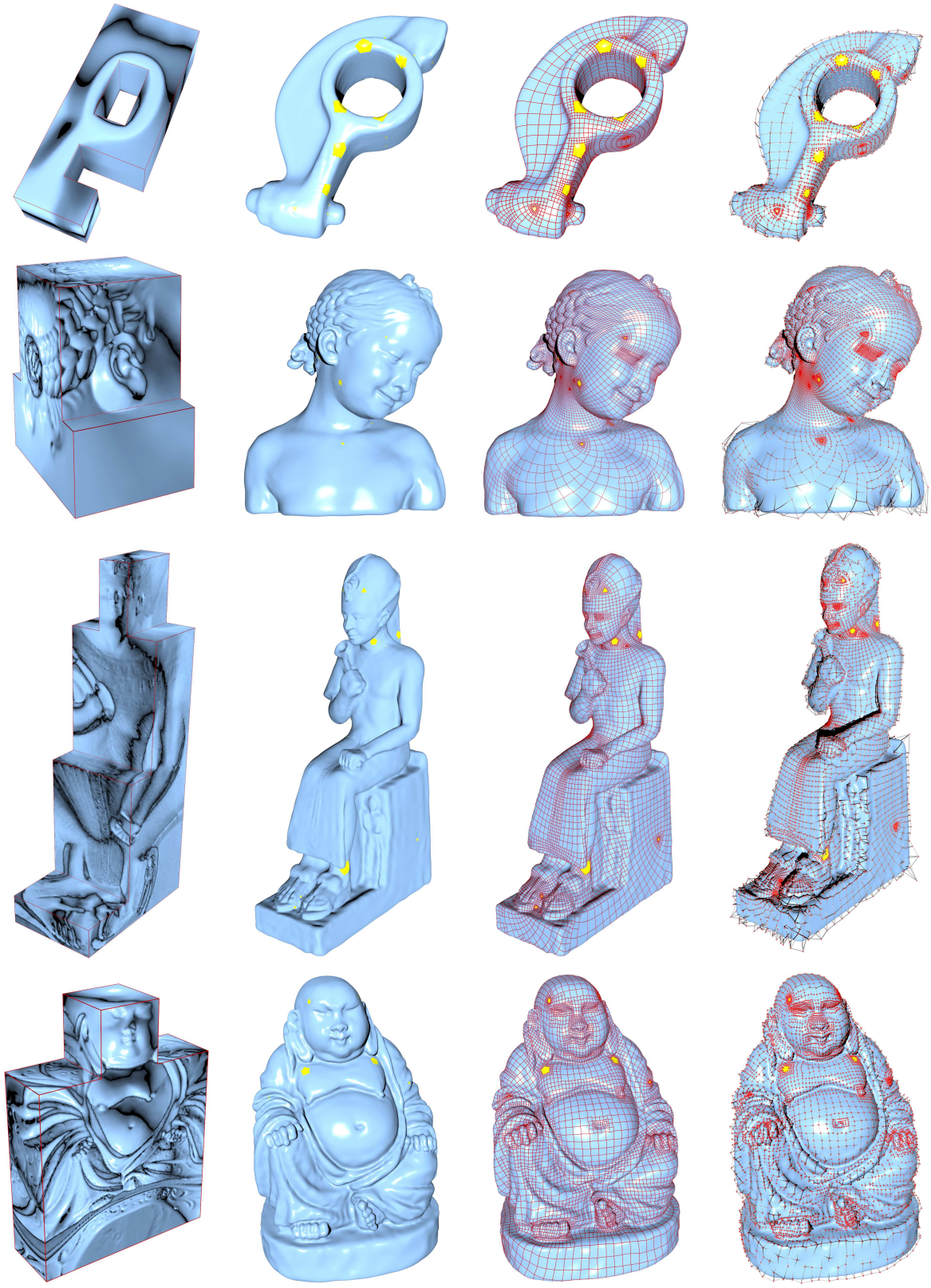
The polycube spline is naturally built upon the polycube map which serves as its parametric domain. The use of polycubes for spline surface definition and construction is the first attempt to take advantage of the rectangular structure over

the boundary of polycubes, allowing the parametric domain to actually mimic the geometry of the modeled objects with lower area distortion while enforcing their topological consistence.

The proposed polycube spline construction pipeline in this work is mainly composed of two steps: polycube map construction and spline construction with the polycube map serving as the parametric domain. The quality of a polycube map (in terms of angle and area distortion) can be quantitatively controlled by designing the polycube which resembles the geometry of the input shape and shares the same topology. In this work the polycube approximation for the input 3D shapes is built manually by using Maya. Designing polycubes for shapes of complicated geometry and topology remains to be very tedious and labor intensive. Domain knowledge and expertise are required to construct a polycube approximation for the input 3D shape which will further induce a polycube map in high quality. Another key factor to the quality of the resulting polycube map is the parameterization methods used to compute the 1-1 mapping between the input 3D shapes and the polycube approximations. This process is usually time consuming especially for shapes with complicated topologies and geometric features, and normally takes up to 70% of the entire execution time for the spline construction pipeline in our experiments (after manually constructing the polycube). Parameterization with small angle distortion and area distortion is highly desirable for spline construction which will usually lead to a spline surface with smaller fitting errors and fewer number of control points. The global conformal parametereization methods used in this work, though theoretically sound to guarantee a bijection, may not be practically useful for a topologically complicated surface since the rounding error will cause serious numerical problems in computing the hyperbolic parameterization and the fundamental domain, which therefore leads to a highly unstable parameterization result. On the other hand, in existing work the 3D surfaces with negative Euler characteristics are usually required to reduce the number of faces significantly before computing the hyperbolic parameterization. Therefore, many geometry details are lost in the resulting polycube maps, which is for sure not desirable for spline-based surface reconstruction. We will address this issue in the subsequent chapters by providing algorithms which are better for complicated shapes such as high genus surfaces and surfaces with open boundaries.

Computational conformal geometry algorithms usually use numerical calculations to approximate smooth cases, and usually involve solving a linear system. As discussed in section 4.4.2, the computation of the spline surface also involves solving a linear system. So the overall time required to compute a spline surface to approximate the given 3D shape is decided by the speed of the linear system solver. Linear system solver has been a research topic and been implemented in commercial softwares (e.g. Matlab®, Intel® MKL, etc) for many years. In some cases the system is sparse, symmetric and positive definite. Then we prefer to use the conjugate gradient method or Cholesky decomposition to save the computational time from $O(n^3)$ to $O(n^2)$ for a linear system of size $n$. NVIDIA® CUDA (Common Unified Device Architecture) technology has been widely applied in solving linear systems recently. IBM® also provides source codes for sparse matrix-vector multiplication (SpMV4GPU). With the rapid development of computer hardwares, the faster linear system solver can be expected. On the other hand, in addition to the traditional reverse engineering and geometric design area in which they are usually used for scattered data fitting and surface design, spline construction is usually used as the preprocessing step for other graphics applications like finite element simulation, fluid simulation, visualization, and e-manufacturing to convert the discrete data inputs into continuous, compact representations. In most of these applications, the fitting quality of the spline surface with respect to the original discrete data input is more important compared to the construction time. Once the spline computation is finished, the result can be reused whenever and wherever the continuous representations are needed. And the spline representations with high quality usually means fewer number of control points and smaller fitting errors, which will for sure lead to faster surface rendering/evaluation and introduce more flexibility in the real world applications.

We will focus on discussing the fitting quality (in terms of root-mean-square error and the number of control points) instead of the execution time when evaluating the proposed data modeling frameworks in the subsequent chapters.



(a) Polycube map (b) T-spline (c) T-junctions (d) Control points

**Figure 51:** Construction of manifold T-splines using polycube maps.

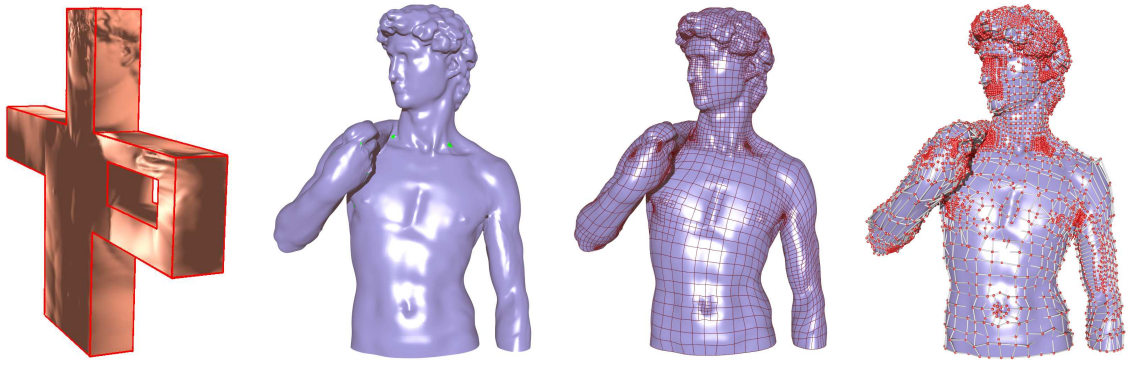
# Chapter 5

# User-Controllable Polycube Map

## 5.1 Introduction and Motivation

There are two research directions immediately following Gu *et al.*'s work on manifold splines. One is to further reduce the number of extraordinary points. In [58], Gu *et al.* presented a method to construct manifold splines with single extraordinary point reaching their theoretic lower bound of singularity for real-world applications. They first computed a special metric of any manifold domain such that the metric becomes flat everywhere except at one point. Then, the metric naturally induces an affine atlas covering the entire manifold except this singular point. Finally, manifold splines are defined over this affine atlas. They showed that the uniformity of the metric varies drastically depending on the location of singularity.

Another direction, in sharp contrast, is to increase the number of extraordinary points to reduce the total area distortion in the affine atlas. In [147](Chapter 4), we proposed polycube T-splines which is a variant of manifold splines such that the metric of the affine manifold (polycube without corners) is explicitly determined by the geodesic distance on the polycube. Compared with [58], the polycube domain offers a rectangular structure which necessarily facilitates subsequent geometric computing and shape analysis. Within our work in [147], the user first constructs the polycube manually. Then both the 3D model and polycube are mapped to one of the canonical domains, i.e., sphere $\mathbb{S}^2$, Euclidean plane $\mathbb{E}^2$, and hyperbolic disc



(a) User-controllable polycube map. (b) Polycube T-spline. (c) T-junctions on polycube spline. (d) Close-up of control points.

**Figure 52:** Polycube spline for the David Body model. (a) The user-controllable polycube map serving as the parametric domain. (b) and (c) Polycube T-splines obtained via affine structure induced by the polycube map. Note that our polycube spline is globally defined as a "one-piece" shape representation without any cutting and gluing work except at the finite number of extraordinary points (corners of the polycube). The extraordinary points are colored in green in (b). The red curves on the spline surface (see (c)) highlight the T-junctions. (d) Close-up of the spline model overlaid by the control points. The polycube T-spline contains 9781 control points. The original model contains $100K$ vertices. The root-mean-square error is 0.3% of the model's main diagonal.

$\mathbb{H}^2$, depending on the topology of the input model. Next, we try to find a map between the fundamental domains which induce the map between the input 3D shape and polycube. This method is completely different from the method introduced in [142] such that the new approach is essentially intrinsic which completely avoids the direct projection of 3D points to the polycube domains.

Although the method presented in [147] can naturally compute the polycube map in an intrinsic way, it has some drawbacks: (1) there is very little user control which can be employed interactively. For example, the user can only specify three points on the 3D model and their images on polycube for genus-zero cases. Therefore, they can not directly control the desired location of extraordinary points (corners of polycube). If the extraordinary points happen to locate on the highly detailed, feature-rich regions, then it is difficult to fill the "holes" in the postprocessing step; (2) It is difficult to handle open surfaces. The only feasible way is to use double covering technique introduced in [64] which converts the open surfaces into closed ones. However this technique will at least double the time complexity

and not practical for complex large-scale datasets; (3) It is difficult to handle high genus models, since computing the fundamental domain of any high genus model is known to be error-prone since the numerical truncated error may cause serious problems when the points are near the boundary of the Poincare disk; and (4) It is difficult to control the total area distortion if the user-designed polycube differs from the given 3D model too much.

In this chapter, we aim to further improve our work in [147] (Chapter 4) by developing a novel framework of user-controllable polycube maps, which overcomes the aforementioned disadvantages and challenges, and is much more efficient and accurate. Within this framework, the current approach allows users to directly specify the extraordinary (corner) points of the polycubes on the input 3D surfaces. The location of singularities can be interactively placed where no important geometric features exist in order to facilitate the subsequent hole-filling process. We then develop algorithms for computing polycube maps in an intrinsic way, and show that the resulting user-controllable polycube map is an ideal parametric domain for spline construction, reverse engineering, and other applications. Figure 52 demonstrates one example of polycube splines construction upon user-controllable polycube maps.

The specific contributions of this work are as follows:

1. We propose a novel framework to construct user-controllable polycube maps by using discrete Ricci flow. Our method is fundamentally different from Tarini *et al.*'s technique [142] and the method proposed in [147]. The user is allowed to choose the extraordinary points directly and freely on the given 3D surfaces, thus, can avoid the high detailed feature-rich regions which then facilitates later hole-filling processes.
2. The proposed method for polycube map construction has lower area distortion compared with traditional methods and enforce small angle distortion as well. By minimizing the size of singularities on the parametric domain, we can ensure that the corresponding holes in the resulting surfaces are also small.
3. The proposed method can construct polycube map easily for *high genus* surfaces and *open* surfaces, which are usually difficult to be handled by the traditional methods as explained above.

The remainder of this chapter is organized as follows. We present the details of our algorithm to construct the user-controllable polycube map of arbitrary topology in Section 5.2. We then discuss the benefits of the user-controllable polycube map in manifold spline construction and demonstrate the experimental results in Section 5.3.

## 5.2 Construction of Polycube Maps

In this section, we explain in details our algorithm for constructing polycube maps for surfaces with arbitrary topologies.

The key differences between the techniques employed in [142, 147] and ours in this work are that Tarini *et al.*'s technique is trying to find a one-to-one mapping from the original surface to the polycube surface extrinsically, which typically requires the projection of points from the suface to the polycube. As a result, their method is usually quite difficult to handle cases where the surface and the polycube differ significantly, because the point projection does not guarantee a one-to-one correspondence; the methods used in [147] compute such a mapping in an intrinsic way. They first conformally map the 3D shape and the polycube to the same canonical domains (e.g., sphere, Euclidean plane, or hyperbolic disk), then construct a map between these two domains, which induces a one-to-one map between the 3D shape and the polycube. The drawback of this intrinsic method in practice is that the user has very limited control on the global mapping. For example, users can not control the positions of extraordinary (corner) points. If the vicinity regions of those points have rich geometric features, later hole-filling processes will unavoidably become very challenging and error prone. In contrast, our new method offers users the full control of the corner point placement, therefore, users can choose the corner points at regions with fewer geometric features to simplify the hole filling procedure. Furthermore, the method in [147] builds the polycube manually first, then construct the mapping between the surface and the polycube. If the polycube changes, the mapping need to be re-calculated completely; whereas, in our current method, we establish the mapping first, then we determine the polycube based on the mapping. If we modify the shape of the polycube, the correspondence between the surface and the polycube does not change. Therefore, we can adjust the shape

of the polycube easily to obtain a better fitting for the original surface defined over the polycube. Our experimental results show that the new polycube method also introduces lower area distortion. Lower area distortion in the vicinity of corner points will ensure better hole filling results.

The polycube map is constructed in the following way:

1. Users set the positions and the curvatures of the corner points on the surface.
2. We deform the Riemannian metric of the surface by Ricci flow, such that all the corners have the prescribed Gaussian curvatures, and other points are flat.
3. We compute the straight lines connecting corners on the surface under the new metric to partition the surface to a collection of planar quadrilaterals.
4. We transform each quadrilaterial to a planar rectangle by setting the corner angles to be $\frac{\pi}{2}$'s and running Ricci flow.
5. Assembly all the planar rectangles to the desired polycube. For vertices on the edges of the polycube, they might be mismatched. We enforce them to meet together on the edge, and use harmonic map to relax the interior of each rectangle.

Next we will give a brief introduction to discrete Ricci flow in 5.2.1, then the detailed algorithm for polycube map construction will be presented in 5.2.2.

### 5.2.1 Discrete Ricci Flow

Suppose $S$ is a surface with a Riemannian metric $\mathbf{g}$. Let $u : S \to \mathbb{R}$ be a function on the surface, then $\bar{\mathbf{g}} = e^{2u}\mathbf{g}$ is also a Riemannian metric of $S$, where $u$ represents the area distortion and called the *conformal factor*. Furthermore, the angles between two tangle vectors at the same point measured by $\mathbf{g}$ equal to those measured by $\bar{\mathbf{g}}$, therefore, we say $\bar{\mathbf{g}}$ is *conformal* to $\mathbf{g}$. Gaussian curvatures are determined by Riemannian metrics. Let $K$ and $\bar{K}$ are the Gaussian curvature functions induced by $\mathbf{g}$ and $\bar{\mathbf{g}}$, respectively. Then $K$, $\bar{K}$, and $u$ are governed by the following Yamabe equation:

$$\bar{K} = e^{2u}K(-\Delta_{\mathbf{g}}u + K), \tag{31}$$

where $\Delta_{\mathbf{g}}$ is the Laplace-Beltrami operator determined by $\mathbf{g}$. This equation shows that given a desired Gaussian curvature $\bar{K}$, we can uniquely determine a Riemannian

metric $e^{2u}\mathbf{g}$. The desired metric can be computed using *Ricci flow* method:

$$\frac{du(t)}{dt} = \bar{K} - K(t), \tag{32}$$

where the initial condition is $u(0) = 0$, and $K(t)$ is the Gaussian curvature induced by the metric $e^{2u(t)}\mathbf{g}$. Riccif flow is proven to be convergent to the unique solution under the constraint that the surface area is preserved during the flow [67].

Discrete Ricci flow method is introduced in [16] and discrete Euclidean Ricci flow is applied for solid modeling in [58]. In practice, the surface is approximated by a triangular mesh. The Riemannian metrics are approximated by the edge lengths. The Gaussian curvatures are approximated as the angle deficit from $2\pi$ at each vertex. The conformal metric is approximated by circle packing metric, where the mesh is covered by a collection of circles centered at each vertex. The circles intersect with each other. We can change the circle radii and preserve the intersection angles, then the radii and the intersection angle together determine the edge lengths, then the discrete curvatures at the vertices. Let the circle radii at vertex $v_i$ be $\gamma_i$, $u_i$ be $\ln\gamma_i$, then *discrete Ricci flow* has exact the same form as the smooth Ricci flow

$$\frac{du_i(t)}{dt} = \bar{K}_i - K_i(t),$$

with a normalization constraint, that during the flow the total area of the mesh is preserved. Discrete Ricci flow is a powerful tool to design edge lengths according to the user defined curvatures.

Furthermore, discrete Ricci flow is the gradient flow of the so called discrete Ricci energy. Let $\mathbf{u}$ be the vector of logarithms of radii $(u_1, u_2, \cdots, u_n)$, $\mathbf{k}$ be the vector of vertex Gaussian curvature
$(K_1, K_2, \cdots, K_n)$. Let $\mathbf{u}_0$ be $(0, 0, \cdots, 0)$, then the discrete Ricci energy is given by

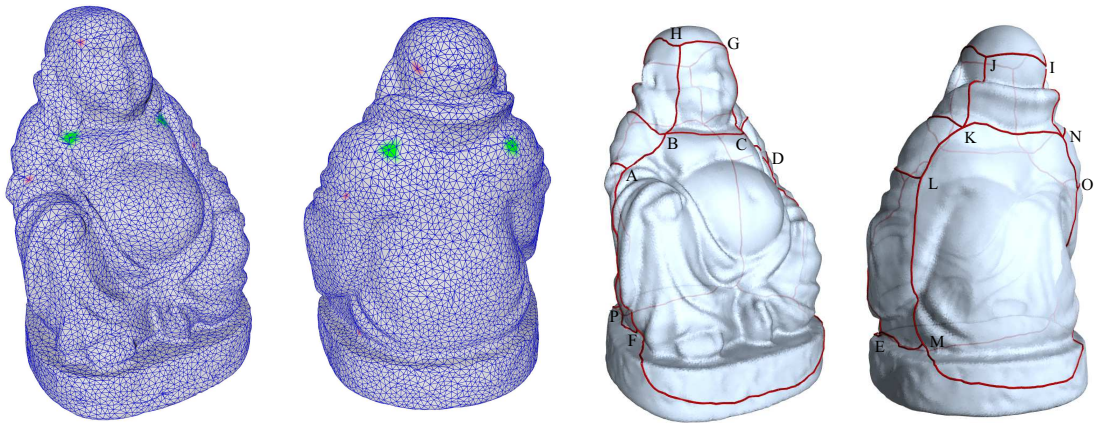
$$E(\mathbf{u} = \int_{\mathbf{u}_0}^{\mathbf{u}} \sum_{i=1}^{n} (\bar{K}_i - K_i) du_i.$$

It is proven that the discrete Ricci energy is convex, therefore has a unique global minimum, which induces the curvature $\bar{\mathbf{k}}$. Therefore, we can use Newton's method to compute the desired metric from the user-defined curvature.

### 5.2.2 Construction of Polycube Maps

The fully user-controllable polycube map can be interactively built using the following procedure:

1. Corner Selection. Given a mesh $M$ with arbitrary topology, user can design the polycube $P$ based on the shape of the surface by directly selecting corners of $P$ on $M$. The choices of the corners reflect the symmetry of $M$. The curvature at each corner $c$ equals to $(2-\frac{k}{2})\pi$, where $k$ is the valence of $c$ on the polycube $p$. Namely, protruding corners are with $\frac{\pi}{2}$, recessed corners are with $-\frac{\pi}{2}$. The total curvatures of all corners equals to $2\pi\chi(M)$, where $\chi(M)$ is the Euler-characteristic number of $M$. Figure 53 (a)(b) shows the selected corner points on Buddha model. The red corners are the protruding corners, the green corners are the recessed corners. For non-corner vertices, we set the curvature to be zero.



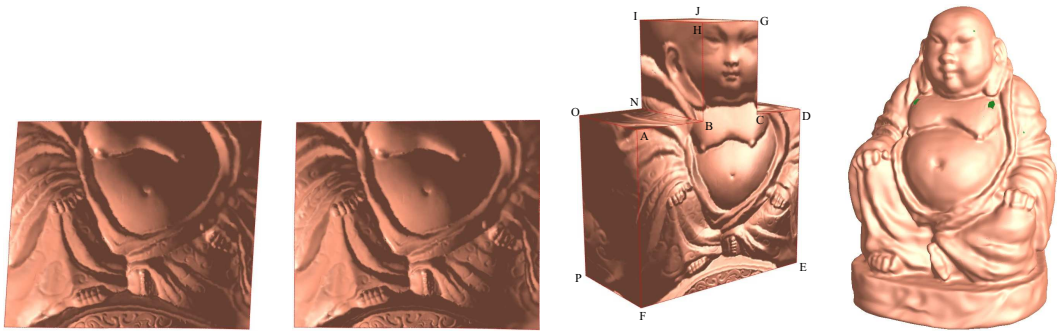

(a) (b) (c) (d)

**Figure 53:** Corner points are marked on Buddha model in (a) and (b), red ones with $\pi/2$ target Gaussian curvature, and green ones with $-\pi/2$ target Gaussian curvatures. Geodesics between corner points are marked with sharp edges in (c) and (d), which are computed using Dijkstra's algorithm with computed conformal metric as edge lengths.

2. Mesh Partition. We use the discrete Euclidean Ricci flow to compute a new circle packing metric according to the target curvature. For any two corners $c_1, c_2$ on the mesh, whose correspondences are connected on the polycube, we compute the shortest path connecting them on the mesh under the new metric

using Dijkstra's method (Figure 53(c)(d)). All such shortest paths segment the mesh to patches. Figure 54 (a) shows one patch from the partition of the buddha mesh using this step, which corresponds to one face of the polycube.

3. Rectification. Each patch is a planar quadrilateral under the new metric, but may not be a rectangle. We can use the Ricci flow method to rectify the planar quadrilateral to the rectangle by setting the target curvatures of 4 corners to be $\frac{\pi}{2}$, and all the other interior and boundary vertex curvatures to be zero. Ricci flow can find a flat metric, the layout of the mesh under the flat metric is a rectangle. The aspect ratio of the rectangle is solely determined by original geometry of the patch. Figure 54 (b) illustrates the rectification result.




(a) (b) (c) (d)

**Figure 54:** One patch from Buddha model after partition is shown in (a) and (b), which corresponds to one face of the polycube. (a) shows the quadrilateral before rectification, (b) shows the rectangle after rectification. User-controllable polycube map for the Buddha model is shown in (c), and its corresponding polycube T-spline surface is shown in (d). The extraordinary points are colored in green.

4. Polycube Assembly. Assemble all the rectangles to a polycube, scale each rectangle along $x$ and $y$ directions whenever it is necessary. First, we build the dual graph of the polycube, each node represents a face of the polycube, each edge corresponds to an edge. Then we use breadth first searching method to traverse the dual graph. We first embed the root face, each time we access a new face, we determine the coordinates of its corners. In this way, we can embed the whole polycube in $\mathbb{R}^3$. Figure 54 (c) shows the polycube map for genus-0 Buddha model.

If two rectangles on the polycube share one edge, enforce the corresponding vertices to align each other. Then we use a discrete harmonic map to relax the positions of the interior vertices of each rectangle with the fixed boundary condition.

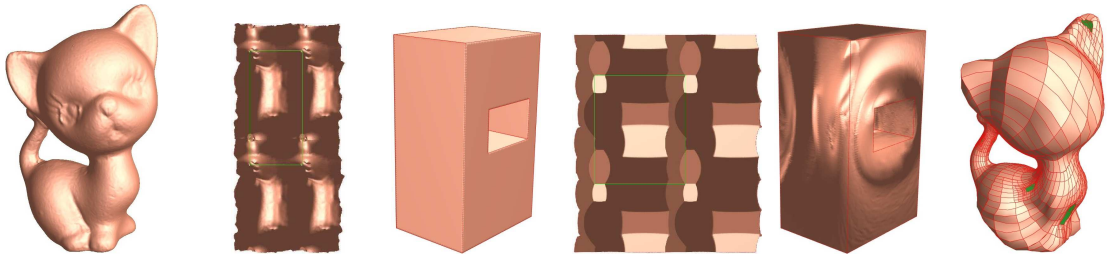
Using the above construction procedure, the mapping between the polycube and the surface is automatically established. The shape of the polycube and the correspondence are fully determined by corner points. Therefore, the choices of the corner points become critical. The followings are the important criteria for choosing the corner positions: (1) the corners should be at regions with fewer geometric features for the purpose of better hole filling; and (2) the configuration of the corners should reflect the symmetry of the original surface.

Our experimental results in 5.3 will show that current method gives users much more freedom to design the polycube; it induces lower area distortion between the surface and the polycube; it is capable of handling surfaces with more complicated topologies, such as high genus surfaces or open surfaces, which are difficult to handle by using conventional methods.

## 5.3 Defining Manifold Splines Over Polycube Maps

As shown in [147], the polycube map of given 3D surfaces naturally induces the affine structure with a finite number of extraordinary points (corners). Any planar spline schemes which satisfy the parametric affine invariant property can be generalized to manifold domain via affine structure [60]. Therefore, spline surfaces can be defined over the polycube map directly. The data fitting quality using polycube T-splines depends heavily on the construction of underlying polycube maps. In this section, we explain with examples that the introduced user-controllable polycube maps are better for manifold spline construction compared with traditonal methods [147].

In [147], they first conformally map the 3D shape and the polycube to the same canonical domains (e.g., sphere, Euclidean plane, or hyperbolic disk), then construct a map between these two domains, which induces a one-to-one map between the 3D shape and the polycube. Figure 55 demonstrates the above mapping method for constructing a polycube map of the genus-one kitten model. Both the

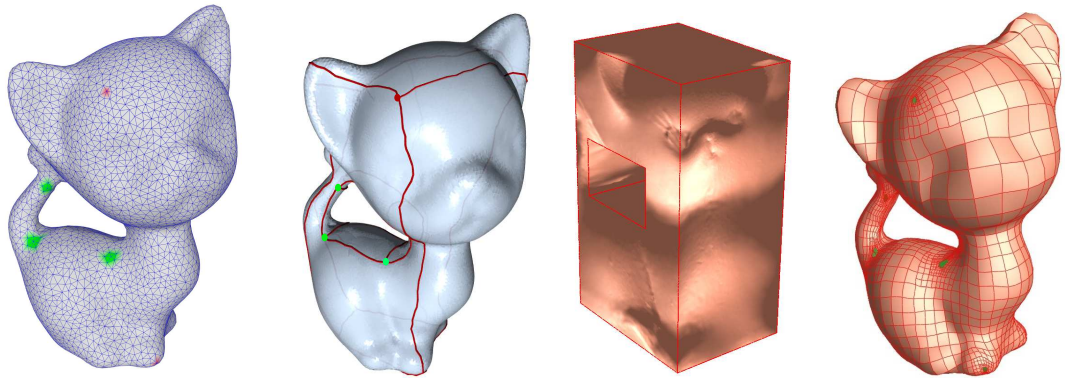


(a) $M$ (b) $\phi_M$ : $M \to D_M \subset \mathbb{E}^2$ (c) $P$ (d) $\phi_P$ : $P \to D_P \subset \mathbb{E}^2$ (e) $\phi : M \to P$ (f) Initial fitting result

**Figure 55:** Genus-one polycube map induced by Euclidean structure. The genus one Kitten model $M$ in (a) is conformally mapped to the Euclidean plane in (b). The fundamental domain is a rectangle region enclosed by the green boundary in (b). Then, a polycube $P$ in (c) is also parameterized over the rectangular region in the same way in (d). By matching the two fundamental regions in (b) and (d) via an affine map, the polycube map for the Kitten model is established in (e).

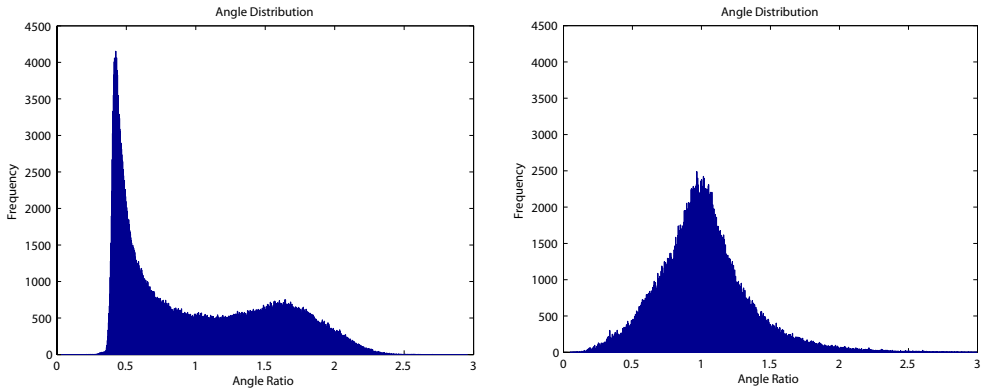
kitten model and its polycube approximation are parameterized using the holomorphic 1-form method. Their fundamental polygons are extracted and mapped by an affine map. The affine map further induces a bijective map between the kitten model and the polycube. The drawback of this intrinsic method is that the user has very limited control on the entire mapping. For example, the user can not control the positions of extraordinary points. If the vicinity regions of those points have important geometric features, the following hole-filling process will be very challenging and error prone. Figure 55(f) shows the initial fitting result from the polycube map in Figure 55(e). One of the corner points is at the tip of the ear, which is difficult to fill compared with other smooth regions.

Since our method allows users to select the locations of corner points on the 3D surfaces directly, we can put corner points at regions where no important geometric features exist. As shown in Figure 56(a), we place corner points at smooth regions. The resulting polycube map is shown in Figure 56(c). Figure 56(d) demonstrates the initial fitting result from the user-controlled polycube map. We can also tell from the shape of T-cells (Figure 55(f), and Figure 56(d)) that the introduced polycube map is more conformal compared with [147]. The affine map used in [147] to align the two fundamental polygons in $E^2$ sacrifices quite a lot the conformality of the mapping. We measure the angles of each triangle in $P$ and compare with the original



(a) (b) (c) (d)

**Figure 56:** Genus-one polycube map constructed by discrete Euclidean Ricci flow. Corner points are marked in (a), red ones with $\pi/2$ target Gaussian curvature, and green ones with $-\pi/2$ target Gaussian curvature. Geodesics between corner points are marked with sharp edges in (b). (c) shows the resulting polycube map and (d) demonstrates the initial spline fitting result based on (c). The red curves on the spline surface highlight the T-junctions.




(a) (b)

**Figure 57:** Comparison of angle ratio distribution: (a) The result for the polycube map in [26] (Figure 55(e)); (b) The result for the user-controllable polycube map (Figure 56(c)).

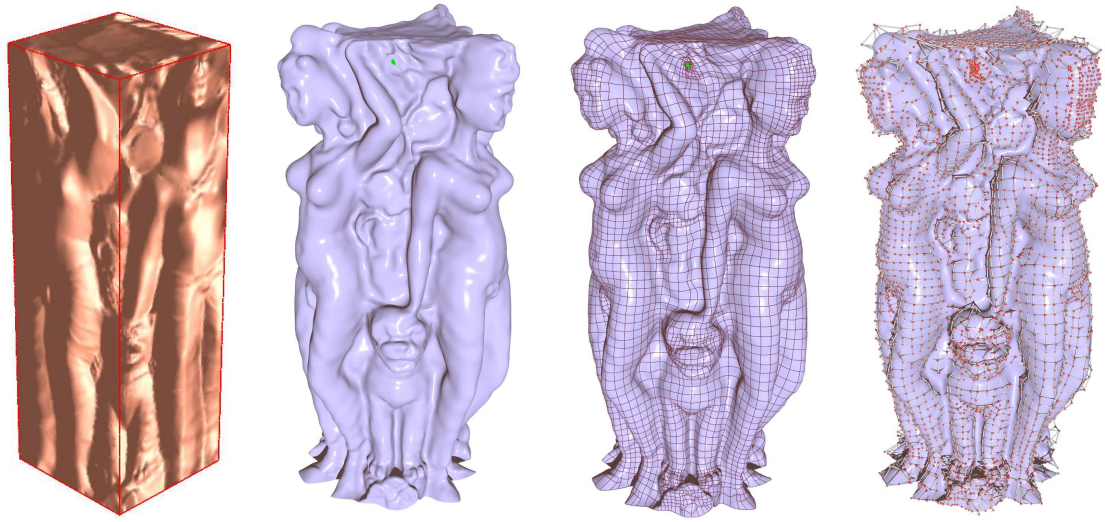
ones in $M$. The distribution of the angle ratio is illustrated in Figure 57. We can see that the ratio is mainly concentrated around 1 (which should be ideal) for our new method, and it shifts away from 1 for the method in [147] due to the affine map used to align the fundamental polygons.

Furthermore, the introduced polycube map usually induces lower area distortion between the surface and the polycube. For the examples shown in Figure 55 and Figure 56, area distortion for the method employed in [147] is 8.7548, while it is 0.8477 for the user-controllable polycube map. The area distortion is calculated as $\sum_{\Delta \in M}((\frac{area(\Delta')}{area(\Delta)}-1)^2 \times area(\Delta))$. The areas of the input model $M$ and the resulting polycube $P$ are normalized to 1 before the computation. $\Delta'$ is the image of the triangle $\Delta$ of $M$ on $P$. Lower area distortion ensures that the holes on the resulting spline surface will be small by making the holes around corner points on the domain polycube small.

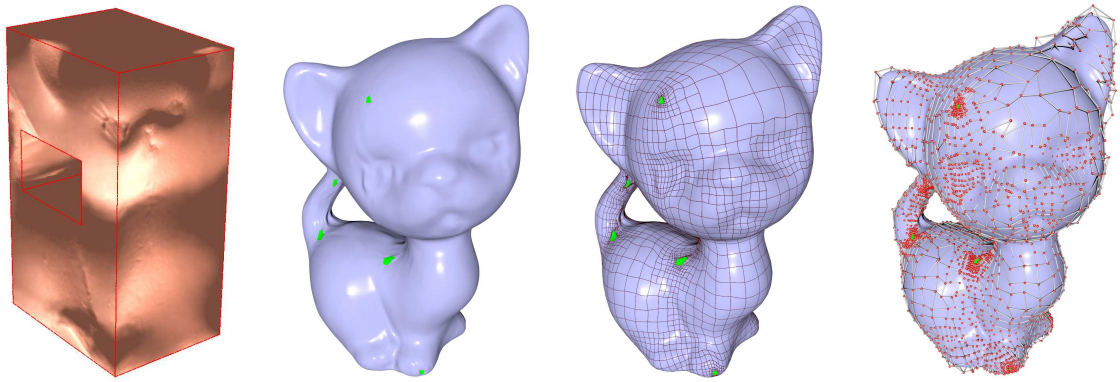
Another advantage of the proposed new method for spline construction is that it is capable of handling surfaces with more complicated topologies, such as high genus surfaces or open surfaces, without introducing extra overhead. In reality these surfaces are very difficult to handle by using conventional methods. Figure 52 shows the polycube map for David body model which is a genus-one surface with two boundaries. It is difficult to construct the polycube map for it by using the traditional method. The only feasible way is to use double covering technique which converts the open surfaces into closed ones. However, this technique is far less efficient because it at least doubles the topology complexity and is not practical for complicated, large-scale datasets. By allowing selecting the corner points on the surfaces interactively, the construction of the polycube map for these complicated surfaces is now as easy as that of simple surfaces. Figure 52 shows the spline surface built upon the resulting polycube map. Figure 58 show more examples of polycube maps and spline surfaces for open surfaces and high genus surfaces, respectively. The statistics of the examples are shown in Table 4.

**Table 4:** Statistics of various test examples (after 5 iterations): $g$, genus of polycube $P$; $N_b$, # of boundaries; $N_s$, # of singularities; $N_v$, # of vertices in the input polygonal mesh; $N_c$, # of control points; *rms*, root-mean-square error; $L_\infty$, maximal fitting error; $d_{area}$, area distortion.

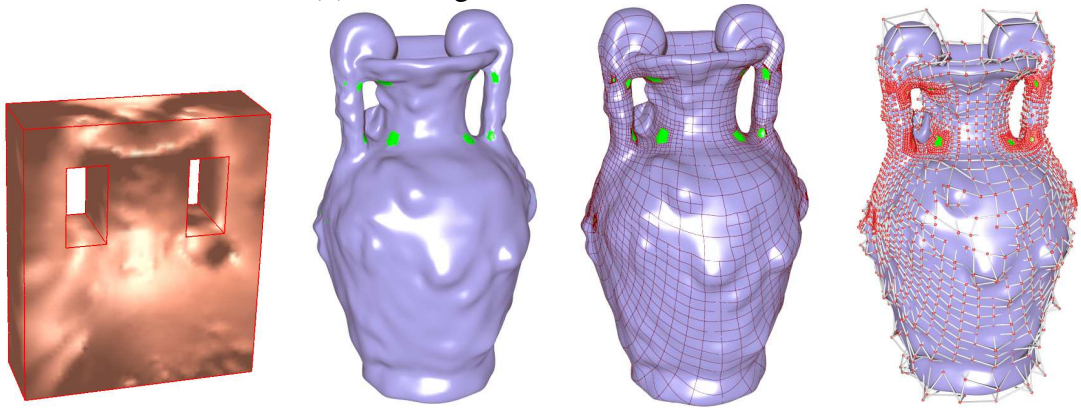
| Object | $g$ | $N_b$ | $N_s$ | $d_{area}$ | $N_v$ | $N_c$ | *rms* | $L_\infty$ |
|---|---|---|---|---|---|---|---|---|
| Buddha (Figure 54) | 0 | 0 | 16 | 0.6785 | 72$K$ | 8842 | 0.06% | 0.42% |
| Three people (Figure 58) | 0 | 1 | 4 | 0.3529 | 100$K$ | 9402 | 0.21% | 0.66% |
| Kitten (Figure 58) | 1 | 0 | 16 | 0.8477 | 45$K$ | 4678 | 0.13% | 0.62% |
| David body (Figure 52) | 1 | 2 | 24 | 0.5378 | 100$K$ | 9781 | 0.32% | 0.71% |
| Amphora (Figure 58) | 2 | 0 | 24 | 1.02 | 65$K$ | 6791 | 0.08% | 0.46% |



(a) 3-people: genus-zero open surface.



(b) Kitten: genus-one surface.



(c) Amphora: genus-two surface.

**Figure 58:** Various examples of polycube T-splines.

# Chapter 6

# Automatic Polycube Map Construction

## 6.1 Introduction and Motivation

As we introduced in Chapter 2.1, polycube map is a novel cross-surface parameterization technique where the parametric domain is a polycube (a.k.a. cubical complex). Compared with other global parameterization techniques, the quality of a polycube map (in terms of angle and area distortion) can be quantitatively controlled by designing the polycube which resembles the geometry of the input shape and shares the same topology [142]. Because of their highly regular structure (i.e., each face is a square or poly-square) and the nature of the "one-piece" global parametric domain (i.e., no cutting and abutting), polycube maps have shown great promise in texture mapping and synthesis [95, 142], shape morphing [38], spline constructions [147, 148] and harmonic volumetric mapping [77, 98].

Despite many promising properties and great modeling potentials of polycube maps, polycube maps have not yet been widely applied to real-world applications. The underlying reasons are two-fold: 1) Polycubes are usually constructed manually with great care and specific domain knowledge. Designing polycubes for shapes of complicated geometry and topology remains to be very tedious and labor intensive. 2) Once the polycube is devised, the existing techniques to construct the map between the given 3D shape and polycube require either projection of the

vertices from 3D shape to the polycube (e.g., [142] which is an extrinsic method) or computing a global surface parameterization (e.g., [147] which is an intrinsic method). As a result, many technical challenges can not be easily overcome with all the existing methods. For example, the extrinsic method may not produce a valid one-to-one map if the polycube differs from the modeled shape significantly. The intrinsic method, though theoretically sound to guarantee a bijection, may not be practically useful for a topologically complicated surface since the rounding error will cause serious numerical problems in computing the hyperbolic parameterization and the fundamental domain. In [96, 147], the 3D surfaces with negative Euler characteristics are required to reduce the number of faces significantly before computing the hyperbolic parameterization. Therefore, many geometry details are lost in the resulting polycube maps. In order to arrive at a high-fidelity polycube map, particularly for a complicated real-world object, our goal is to develop more efficient and accurate methods for producing polycube maps for shapes of complicated geometry and arbitrary topology with far less user intervention towards a full automation.

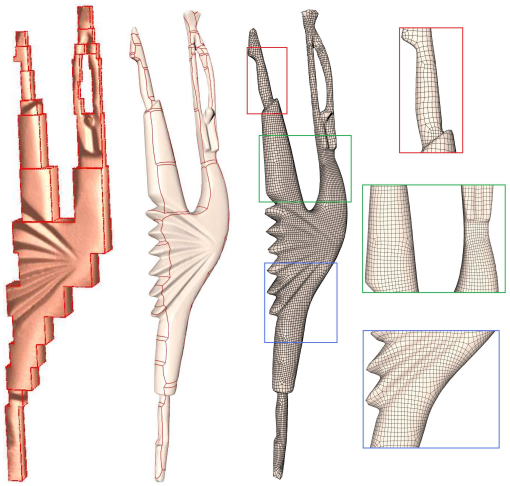



**Figure 59:** Polycube map for the genus-1 Dancer model. (a) shows the constructed polycube map. The red curves in (b) illustrate the polycube structure. (c) shows the quadrilateral mesh generated using the polycube map. (d) shows some close-up views of the quadrilateral mesh. Note that both the polycube and polycube map are constructed automatically.

In particular, this work tackles the aforementioned technical challenges and

develops a novel method to construct polycube maps of arbitrary topology. Compared with the existing methods which usually require tremendous amount of effort from users to design and build the polycubes, our method, in sharp contrast, is automatic. The user may choose to specify two parameters to control how close the polycube mimics the geometry of the input shape, then our algorithm can construct both the polycube and the one-to-one map between the polycube and input shape automatically. As an example, Fig. 59 shows the automatically-constructed polycube map for the genus-1 Dancer model, as well as the quadrilateral remeshing using polycube maps.

The contributions of this work include:

1. We develop an automatic method to construct polycube maps of complicated topology and geometry. The polycube map is theoretically sound to guarantee a bijection between the 3D model surface and the polycube domain.
2. We compare our method with the existing polycube map construction techniques and show that the constructed polycube maps have lower angle and area distortions, and hence, are of high-quality.
3. We apply the constructed polycube maps to various applications, such as polycube T-splines, seamless texture synthesis, and quadrilateral and hexahedral mesh generation, and demonstrate the efficacy of our method in real-world examples.

The remainder of the chapter is organized as follows: Section 6.2 details our algorithmic pipeline of automatic polycube map construction. Section 6.3 shows the experimental results corresponding to different applications, such as quad mesh generation, polycube T-splines, and texture synthesis. We thoroughly compare the newly-developed method with existing techniques in Section 6.4. Finally, discussions are presented in Section 6.5.

## 6.2 Automatic Polycube Map Construction

This section details the theory and algorithmic pipeline of automatic polycube map construction. As mentioned earlier, our method is intrinsic in that it avoids the projection of vertices from a 3D surface to the polycube domain. Therefore, the

major goal is to map the input model and polycube to the canonical domains and then find the map between the canonical domains. The existing intrinsic method proposed by Wang et al. [147] requires the global parameterization, i.e., mapping the models with positive Euler characteristic $\chi > 0$ to sphere $\mathbb{S}^2$, models with $\chi = 0$ to Euclidean plane $\mathbb{E}^2$ and models with $\chi < 0$ to hyperbolic disk $\mathbb{H}^2$. It is known that embedding models with negative Euler characteristic is error-prone when the point is very close to the boundary of the Poincaré disk due to the numerical rounding error. Therefore, Wang et al.'s method is not practical and much less numerically stable to construct polycube maps of large-scale models with negative Euler characteristics.

To construct intrinsic polycube maps in a more robust and practical way, we use a divide-and-conquer strategy, i.e., segmenting the polycube and the given 3D surface into multiple disjoint components, then constructing the piecewise polycube map for each component, and finally computing a globally smooth map for the entire polycube domain. The key reason that we use the divide-and-conquer approach is to avoid the time consuming and error-prone global parameterization since parameterizing each segmented component (of genus-0) to the planar domain is relatively easier, more efficient, and more robust than working directly on the global shape in its entirety. Note that the straightforward gluing of the individual polycube maps have only $C^0$ continuity across the cutting boundaries. Therefore, we must apply a global relaxation algorithm to the entire shape and achieve a globally smooth polycube map. Our automatic polycube map construction algorithm consists of five steps:

1. Given a 3D mesh $M$, construct a harmonic function $f : M \to \mathbb{R}$ and extract all the critical points of $f$ which reveal the topological structure of $M$ (Section 6.2.1).
2. Progressively construct the polycube $P$ using the scan-line like algorithm (Section 6.2.2).
3. Slice $M$ and $P$ into disjoint components $M_i$ and $P_i$, i.e., $M = \bigcup M_i$, $P = \bigcup P_i$, where $M_i$ and $P_i$ are genus-zero open surfaces. Compute the *uniform flat metric* for $M_i$ and $P_i$, and embed them to the multi-hole disks (Section 6.2.3).
4. Compute a one-to-one map between the $M_i$ and $P_i$ by solving a harmonic map between the multi-hole disks (Section 6.2.4).

5. Compute the globally smooth polycube map by solving the harmonic map for the individual face, edge and corner charts (Section 6.2.5).

### 6.2.1 Extracting the Topological Structure

The divide-and-conquer approach requires segmentation of the input mesh to a set of genus-0 shapes. To develop a general and automatic segmentation method, we should extract the topological structure of the input mesh $M$. To achieve this goal, we construct a harmonic function [111], $f : M \to \mathbb{R}$, such that

$$\triangle f = 0, \tag{33}$$

with the boundary condition

$$f(v_0) = 0 \text{ and } f(v_1) = 1,$$

where $\triangle$ is the Laplace-Beltrami operator under the Euclidean metric (edge length) of $M$, whereas $v_0$ and $v_1$ are the bottom-most and top-most points on $M$, respectively. Note if multiple bottom(top)-most points exist, we just pick arbitrary one.

We then find all the critical points of $f$ whose partial derivatives vanish. These critical points can be classified into four categories (see Figure 60):

- Maximal point where a new component starts;
- Saddle point where the handle splits;
- Saddle point where the handle merges;
- Minimal point where the current component ends.

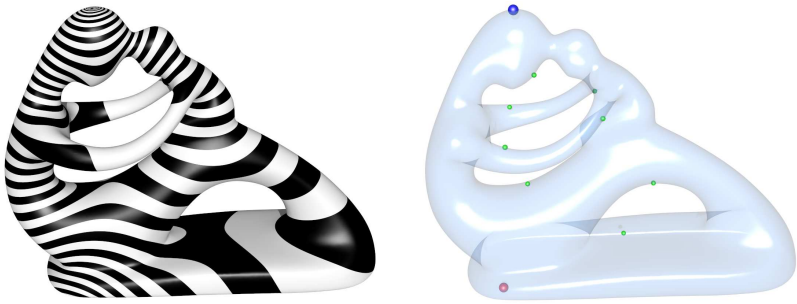
For a closed surface $M$ of genus $g$, the number of critical points satisfies the following equation

$$\#minimal - \#saddle + \#maximal = 2 - 2g.$$

Since the maximal and minimal points are specified by the boundary condition, the number of saddle points is always $2g$.

### 6.2.2 Constructing the Polycube

We sort all the critical points in an ascending order by their $z$ values. Let $v_0, c_1, c_2, \cdots, c_{2g}, v_1$ denote the sorted critical points including the bottom-most and



(a) Harmonic function (b) Critical points

**Figure 60:** Critical points of a harmonic function on a closed surface. (a) shows the harmonic function $\triangle f = 0$, $f(v_0) = 0$ and $f(v_1) = 1$ where $v_0$ and $v_1$ are the bottom-most and top-most points on the model, respectively. (b) The saddle, the global minimal, and the global maximal points are colored in green, red, and blue, respectively.

top-most points which are the global minimal and maximal points, respectively. Let $z(p)$ denote the $z$ coordinate of point $p$. Then we construct $2g+1$ horizontal cutting planes, such that

$$\begin{aligned} z_0 &= \frac{z(v_0)+z(c_1)}{2} \\ z_1 &= \frac{z(c_1)+z(c_2)}{2} \\ &\cdots \\ z_{2g} &= \frac{z(c_{2g})+z(v_1)}{2} \end{aligned}$$

Note that because of shape symmetry, two or more critical points may have the same (or nearly the same) $z$ coordinate. In such a case, only one representative point is selected.

Let $d_z$ be the user-specified parameter for the maximal distance between two adjacent cutting planes. This parameter controls how close the resulting polycube mimics the given shape. Intuitively speaking, the smaller the value of $d_z$, the larger the number of cutting planes, and thus, the more similar to the given shape the polycube approximation is. Note that if the distance between two consecutive cutting planes, say, $z_i$ and $z_{i+1}$ is greater than $d_z$, we uniformly insert $\lfloor (z_{i+1} - z_i)/d_z \rfloor$ cutting planes in-between. Since there is at least one cutting plane between two adjacent critical points, the given shape $M$ is sliced into multiple disjoint components,

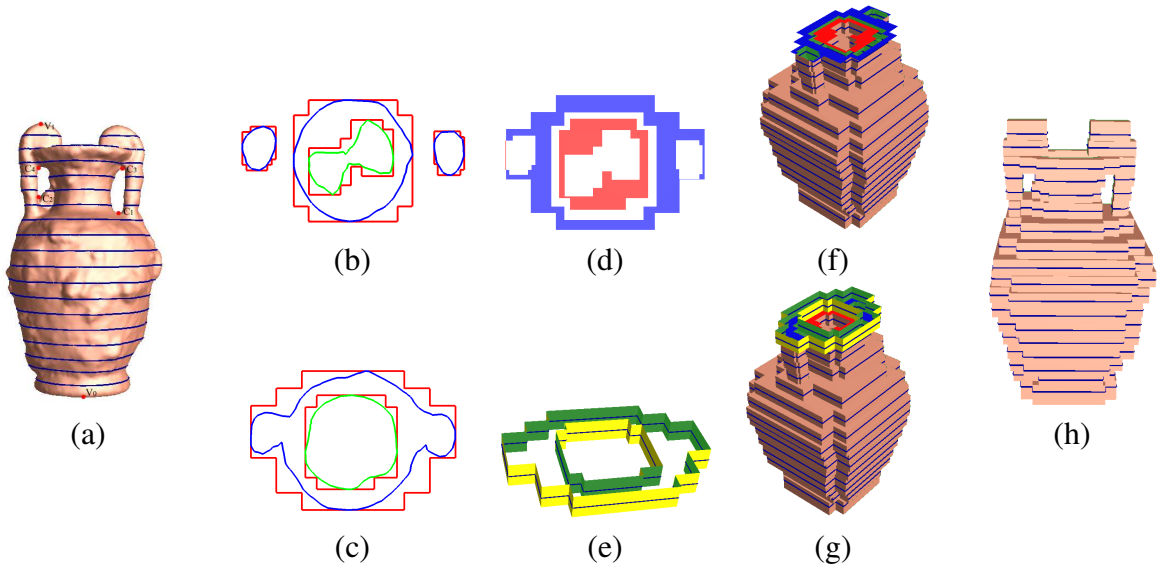



**Figure 61:** Automatic polycube construction for the genus-2 Amphora model (with parameters $d_z = 0.05$ and $d_a = 0.3$). (a) shows the genus-2 Amphora model marked with saddle points $c_1, \cdots, c_4$, and the global minimum/maximum points $v_0, v_1$ in red. A total of 18 horizontal cutting planes slice the given shape $M$ into 29 components, each of which is a genus-0 open surface. The blue curves show the cross-section contours of the horizontal scanning planes. (b) and (c) show the cross-section contours (blue lines) for scanning plane 15 (with $z$ value $z_{15}$) and 16 (with $z$ value $z_{16}$), respectively. The inner boundaries (holes) are drawn in green. Then axis-aligned polygons $Q_i$ (red polygons in (b) and (c)) are used to approximate the curved cross-section contours (blue and green curves in (b) and (c)). (d) shows the polygonal face at $z$ value $(z_{15} + z_{16})/2$, which is the union of $Q_{15} - (Q_{16} \bigcap Q_{15})$ (red) and $Q_{16} - (Q_{16} \bigcap Q_{15})$ (blue). (e) shows the side face by projecting the boundaries of $Q_{16}$ along the $z$-axis from $(z_{15} + z_{16})/2$ to $(z_{16} + z_{17})/2$. (f) shows the result by adding (d) to the partial polycube after processing the first 15 intersections. (g) shows the result by adding the side face (e) to the partial polycube in (f). (h) shows the final polycube. The blue lines in (h) correspond to the intersection contours in (a).

each of which is a genus-zero open surface. Then the polycube can be constructed automatically using the scan-line like algorithm as detailed below.

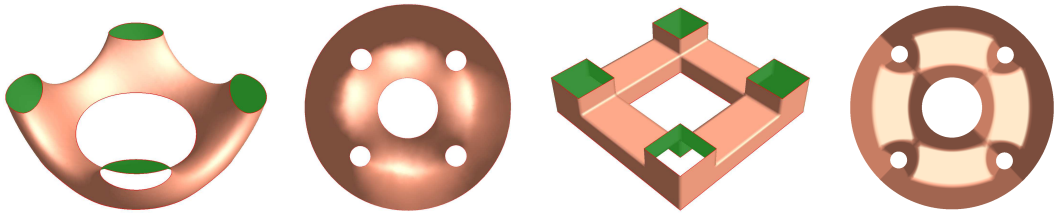
Let us use the genus-2 Amphora model to illustrate our idea and the key algorithmic components. There are four saddle points $c_1$, $c_2$, $c_3$, and $c_4$ shown in Fig. 61(a), whereas $v_0$ and $v_1$ are the bottom-most and top-most points, respectively. Note that $c_3$ and $c_4$ have similar $z$ coordinate, therefor it is not necessarily to differentiate these two points by inserting a cutting plane in-between. In our implementation, two or more critical points are considered on the same level if the difference of their z-coordinates is less than 0.01 of the height of the model. Then the $z$ range of the given model is split into 4 intervals: $[v_0, c_1]$, $[c_1, c_2]$, $[c_2, c_3]$, $[c_3, v_1]$. Next, we uniformly segment the shape by several cutting planes perpendicular to z-axis as shown by the blue lines in Fig. 61(a).

The intersection between each horizontal cutting plane and $M$ is a set of planar curves as shown in Figure 61(b)-(c). Then we approximate these intersection curves by a set of axis-aligned polygons using a quad-tree method, i.e., starting from the bounding rectangle of this polygon, and keep subdividing it until the given approximation accuracy threshold is satisfied or the maximal subdivision level is reached. The approximation accuracy of the axis-aligned polygons $p$ to the input curved contours $c$ is quantitatively measured by the normalized area difference $d_a = area(p \setminus c)/area(c)$. Note that $d_a$ is a user-specified threshold. In general, the smaller the value of $d_a$, the more accurate the approximated axis-aligned polygons to the curved contours, and thus, the more detailed the axis-aligned polygons are.

After we get the axis-aligned polygon approximation of the curved intersection contours, we can readily construct the 3D polycube by extruding the axis-aligned polygons along the $z$ axis and by performing necessary CSG operations. Suppose there are $n$ scanning planes with $z$ values $z_1 < z_2 < \ldots < z_n$ and $Q_i$ is the set of axis-aligned polygons for scanning plane $i$ with $z$ value $z_i$, all the boundaries of the polygons in $Q_i$ are extruded along the $z$ axis from $(z_i + z_{i-1})/2$ ($z(v_0)$ for the first scanning plane) to $(z_i + z_{i+1})/2$ ($z(v_1)$ for the last scanning plane) to form the side face. The polygonal face with the $z$ value $(z_i + z_{i-1})/2$ is computed as the union of $Q_{i-1} - (Q_i \bigcap Q_{i-1})$ and $Q_i - (Q_i \bigcap Q_{i-1})$. The polygon face will be $Q_1$ at the $z$ value $z(v_0)$, and $Q_n$ at the $z$ value $z(v_1)$.

Figure 61(d)-(e) show the polygon face at $(z_{15} + z_{16})/2$ and the side face for the

16-th scanning plane, respectively. Figure 61(f)-(g) show the partial polycube after combining the polygon face and side face for 16-th scanning plane. Figure 61(h) shows the final polycube; the contours for the axis-aligned polygon approximations are colored in blue.



(a) $M_i$ (b) $D_{M_i}$ (c) $P_i$ (d) $D_{P_i}$

**Figure 62:** We map each segmented component $P_i$ and $M_i$ to a multi-hole disk using uniform flat metric, where the Gaussian curvature of the interior vertices is zero and the total geodesic curvature of the boundary is constant, i.e., $2\pi$ for the outer boundary, and $-2\pi$ for each hole. Since the geometry of $P_i$ and $M_i$ are similar, their embeddings of the uniform flat metric are consistent and stable.

### 6.2.3 Uniform Flat Metric and Multi-hole Disk

After a polycube is constructed automatically and then segmented into multiple disjoint components, we are ready for the parameterization step. Note that each segmented component is of a genus-0 open surface, therefore, the idea parametric domain is the Euclidean disc. A common technique for the planar parameterization is solving a harmonic map with the user-specified boundary condition. However, the harmonic map is not suitable for this step since it is very hard to specify the position of boundary points for surface with multiple boundaries. Therefore, we use discrete Ricci flow [16, 61, 80] for the parameterization step since we only need to specify the target curvatures (rather than their positions) of the boundary points.

Suppose $S$ is a surface with a Riemannian metric $\mathbf{g}$. Let $u : S \to \mathbb{R}$ be a scalar function on $S$, then $\bar{\mathbf{g}} = e^{2u}\mathbf{g}$ is also a Riemannian metric which is conformal to $\mathbf{g}$. Let $K$ and $\bar{K}$ denote the Gaussian curvature induced by $\mathbf{g}$ and $\bar{\mathbf{g}}$, then the desired metric $\bar{\mathbf{g}}$ can be computed using

$$\frac{du(t)}{dt} = \bar{K} - K(t), \tag{34}$$

where the initial condition is $u(0) = 0$ and $K(t)$ is the Gaussian curvature induced by the metric $e^{2u(t)}\mathbf{g}$. During this curvature deformation, the metric $\mathbf{g}(t)$ is conformal to the original metric $\mathbf{g}(0)$ at any time $t$. To map the genus-0 open surface to Euclidean plane, we compute the uniform flat metric of $S$, namely, a metric $\mathbf{g}(\infty)$ which is flat everywhere inside the surface and the geodesic curvature is constant on the boundary,

$$\bar{K} = 0, \ v \notin \partial S \tag{35}$$

$$\bar{k}_g = const, \ v \in \partial S, \tag{36}$$

where $\bar{K}_v$ and $\bar{k}_v$ are the target Gaussian and geodesic curvatures. If the total geodesic curvature on each boundary is given, such a uniform flat metric exists and is unique. Using uniform flat metric, we can map genus-0 open surface to a multi-hole disk and the map is guaranteed to be a diffeomorphism.

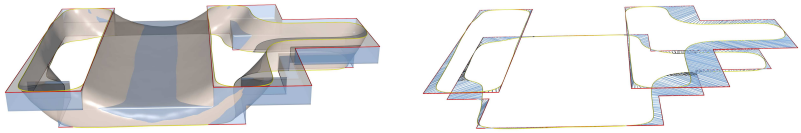
Note that both the given mesh $M$ and polycube $P$ have been segmented into multiple disjoint components $M_i$, $P_i$, $i = 1, 2, \cdots$, each of which is a genus-0 open surface with $b$ ($b \geq 1$) boundaries $C_0 \cup C_1 \cup \cdots \cup C_{b-1}$. For $b \geq 2$, $C_0$ is the boundary with the longest length. We set the target curvature of interior vertices to zero, the total geodesic curvature of the first boundary $C_0$ to $2\pi$ and the total geodesic curvature to be $-2\pi$ for each of the remaining boundaries, $C_i$, $i = 1, \cdots, b-1$. Then the total target Gaussian and geodesic curvatures satisfy the Gauss-Bonnet theorem:

$$\int_S K + \int_{\partial S} k_g = \int_S \bar{K} + \int_{\partial S} \bar{k}_g = 2\pi(2 - 2g - b), \tag{37}$$

where $g = 0$. Once the target Gaussian curvatures are given, we can compute the uniform flat metric by solving the discrete Ricci flow. Then, we embed the shape to the Euclidean plane using uniform flat metric and obtain a $(b-1)$-hole disc as shown in Figure 62.

### 6.2.4 Computing the Piecewise Map

As explained earlier, we take a "divide-and-conquer" approach in that we segment the topologically complicated shape $M$ and polycube $P$ into multiple disjoint components, $M_i$, $P_i$, $i = 1, \cdots$, each of which has simple topology. Then we construct a harmonic map between $P_i$ and $M_i$. Note that the map between $P_i$ and $M_i$



(a) (b)

**Figure 63:** Constructing the mapping between boundary curves $\partial P_i$ to $\partial M_i$. (a) The boundaries of $P_i$ and $M_i$ are planar curves. (b) The blue lines show the map $\psi_i$ between the vertices on $\partial P_i$ and $\partial M_i$ (see Equation 38).

is smooth for the interior vertices of $M_i$ and $P_i$. In general, the map of two adjacent components (components share one common boundary) may not be continuous across the boundary. In order to ensure that the two adjacent polycube maps have $C^0$ continuity across the boundary (otherwise, we can not get the globally smooth polycube map in the next step), we must impose the boundary condition of adjacent components in a consistent way.

We first construct a one-to-one map $\psi : \partial P_i \to \partial M_i$ between the boundaries of $P_i$ and $M_i$ in a piecewise fashion. Note that each boundary is a closed planar curve (on the cutting plane). For each vertex $v \in \partial P_i$, let $\psi(v) \in \partial M_i$ denote its image, then we require that the map $\psi_i$ minimizes the following distance functional

$$\min \int_{\partial P_i} \|\psi_i(v) - v\|^2. \tag{38}$$

Solving the above optimization problem gives rise to a parameterization between the boundaries of $P_i$ and $M_i$ with least distortion. Note that each cutting boundary (not the boundary in the original shape) connects two adjacent components, say, $P_i$ and $P_{i+1}$. Let $v \in \partial P_i$ and $v \in \partial P_{i+1}$, then the above map can guarantee that the images of $v$ under $\psi_i$ and $\psi_{i+1}$ are consistent, $\psi_i(v) = \psi_{i+1}(v)$. Therefore, the resulting polycube maps of two adjacent components are $C^0$-continuous across the boundary, i.e., they are seamless. Figure 63 shows an example of such a map between the boundary curves of $P_i$ and $M_i$.

Let $D_{M_i}$ and $D_{P_i}$ denote the embedding of $M_i$ and $P_i$ in the Euclidean plane using uniform flat metric, respectively. Similar to the intrinsic method proposed in [147], we want to construct the one-to-one correspondence between $P_i$ and $M_i$ by the composite map $\phi_{P_i \to M_i} = \phi^{-1}_{M_i \to D_{M_i}} \circ \phi_{D_{P_i} \to D_{M_i}} \circ \phi_{P_i \to D_{P_i}}$ as shown in the following commutative diagram:

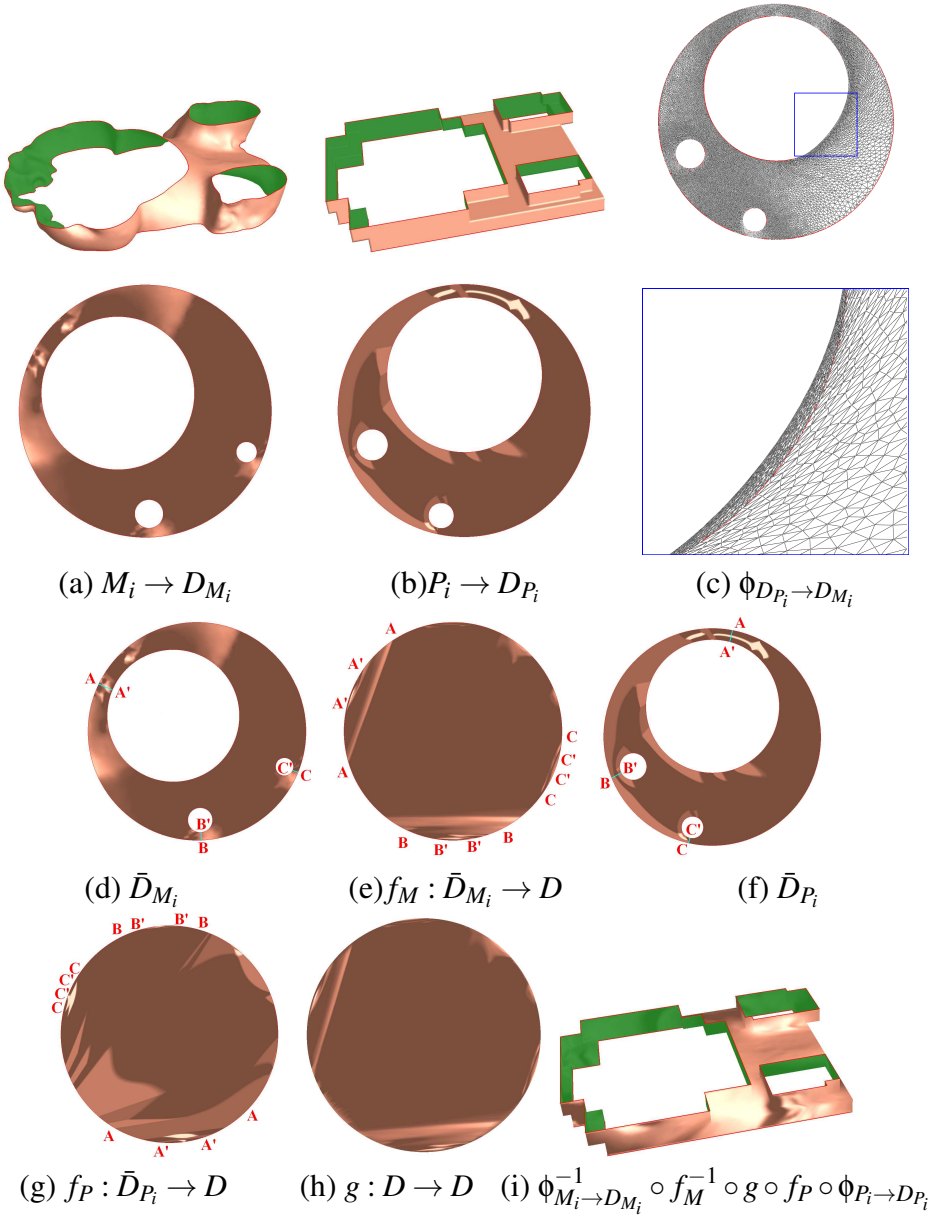
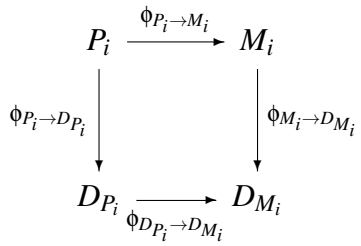



**Figure 64:** Constructing a diffeomorphism between $P_i$ and $M_i$ (see (a) and (b)). Since $D_{M_i}$ and $D_{P_i}$ are not convex, the harmonic map $\phi : D_{P_i} \to D_{M_i}$ is not one-to-one. Pay attention to the flipover in the close-up view in (c). To correct this problem, we first modify the topology of $D_{P_i}$ and $D_{M_i}$ by introducing three cuts to connect the three inner boundaries with the outer boundary (see (d) and (e)). Then we map the modified shape $\bar{D}_{P_i}$ and $\bar{D}_{M_i}$ to unit disk. Next we compute the harmonic map between two unit disks. The boundary condition is specified that the cutting loci are mapped to each other consistently, e.g., the arc $AA'$ in $\bar{D}_{M_i}$ to the corresponding arc $AA'$ in $\bar{D}_{P_i}$. Finally, the polycube map from $P_i$ to $M_i$ is the the composite map $\phi^{-1}_{M_i \to D_{M_i}} \circ f_M^{-1} \circ g \circ f_P \circ \phi_{P_i \to D_{P_i}}$.




Harmonic map is a widely used technique to compute the mapping between two 2D regions. It is well known that a harmonic map $f : A \subseteq \mathbb{R}^2 \to B \subseteq \mathbb{R}^2$ is a diffeomorphism if $\partial B$ is convex and the boundary mapping $f(\partial A) = \partial B$ is a homeomorphism. Unfortunately, both $D_{M_i}$ and $D_{P_i}$ are multi-hole discs, i.e., concave shape. Thus, solving a harmonic map between $D_{P_i}$ and $D_{M_i}$, i.e., $\triangle\phi = 0$ and $\phi(\partial D_{P_i}) = \partial D_{M_i}$, can not guarantee a bijection in general.

To address this problem, we decompose the multi-hole discs to topological discs and then compute the harmonic map between two topological disks.

1. We modify the topology of $D_{M_i}$ and $D_{P_i}$ by introducing the cuts to connect the inner circles and the outer circle such that $\bar{D}_{M_i}$ and $\bar{D}_{P_i}$ are topologically equivalent to a disk. The cuts are constructed as follows: for each inner circle, we find the shortest line to the outer circle. If the line does not pass through any other inner circles, we simply use it as cut locus; otherwise we cut through the shortest line between two inner circles to connect them. We repeat this cutting until the final shape is a topological disk.
2. We compute the harmonic maps $f_M : \bar{D}_{M_i} \to D$ and $f_P : \bar{D}_{P_i} \to D$ where $D \subseteq \mathbb{R}^2$ is a unit disk. Note that $f_M$ and $f_P$ map the boundaries $\partial\bar{D}_{M_i}$ and $\partial\bar{D}_{P_i}$ homeomorphically into the boundary of unit disk $\partial D$. Thus, $f_M$ and $f_P$ are diffeomorphisms.
3. We compute the harmonic map $g : D \to D$ between the two unit disks. The boundary condition is specified such that the cutting loci are mapped to each other consistently.
4. The composite map $\phi^{-1}_{M_i \to D_{M_i}} \circ f_M^{-1} \circ g \circ f_P \circ \phi_{P_i \to D_{P_i}}$ induces the bijection from

$P_i$ to $M_i$. The commutative diagram is shown as follows:

$$\begin{array}{ccc} P_i & \xrightarrow{\phi_{P_i \to M_i}} & M_i \\ \downarrow{\scriptstyle \phi_{P_i \to D_{P_i}}} & & \downarrow{\scriptstyle \phi_{M_i \to D_{M_i}}} \\ D_{P_i} & & D_{M_i} \\ \downarrow{\scriptstyle f_P} & & \downarrow{\scriptstyle f_M} \\ \bar{D}_{P_i} & \xrightarrow[g]{} & \bar{D}_{M_i} \end{array} \tag{39}$$

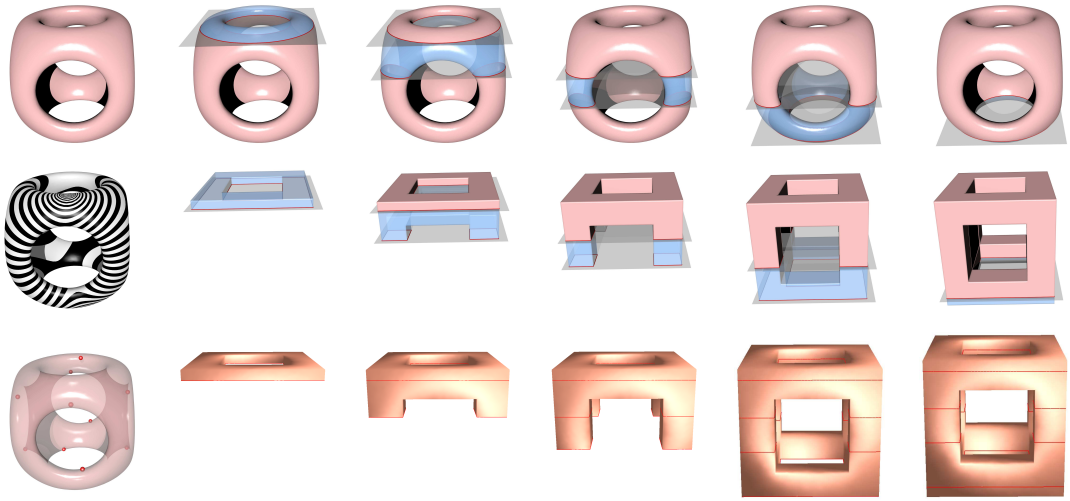
Figure 64 illustrates the idea to compute the diffeomorphism between $P_i$ and $M_i$. Note that a direct harmonic map between $D_{P_i}$ and $D_{M_i}$ is not one-to-one (see the flipover in the close-up view Fig. 64(c)). We modify the topology of $D_{P_i}$ and $D_{M_i}$ and then construct the bijection between $\bar{D}_{P_i}$ and $\bar{D}_{M_i}$ (see Figure 64(i)).

Figure 65 shows the piecewise polycube map construction for the genus-5 Decocube model, which is decomposed into 8 components. A bijective map is constructed for each component, and finally, the whole map is obtained by gluing all components together. Note that the piecewise polycube map is smooth for interior vertices and $C^0$ continuous across the cutting boundaries.

### 6.2.5 Computing the Globally Smooth Polycube Map

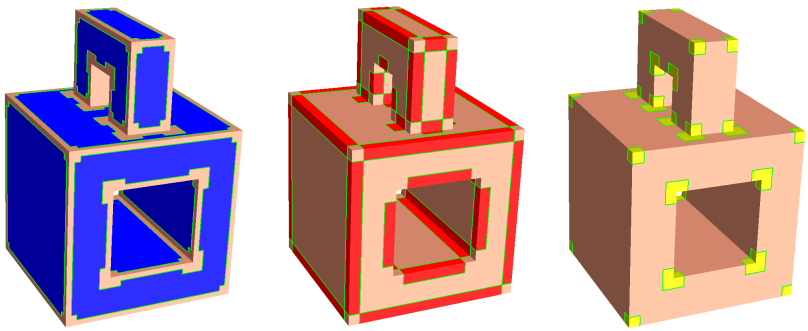
The polycube map constructed by the aforementioned steps is $C^\infty$ inside each segmented component, however, only has $C^0$ continuity across the cutting boundaries. Now, we further improve the quality of the polycube map by solving a harmonic map for the entire shape. Let $\{U_c, \psi_c\}$, $\{U_e, \psi_e\}$ and $\{U_f, \psi_f\}$ denote the set of corner, edge, and face charts, respectively. As shown in Fig. 66, the corner set $U_c$ covers the polycube corners; the edge set $U_e$ covers the interior points of the polycube edge but leaves off corner vertices; the face set $U_f$ covers the interior points of the polycube face but leaves off corner and edge vertices.

For any vertex $v \in U_f$ on the polycube face, $v$ and its neighbors are co-planar. Function $\psi_f : U_f \to \mathbb{R}^2$ is defined by an orthogonal projection along the normal of the polycube face.



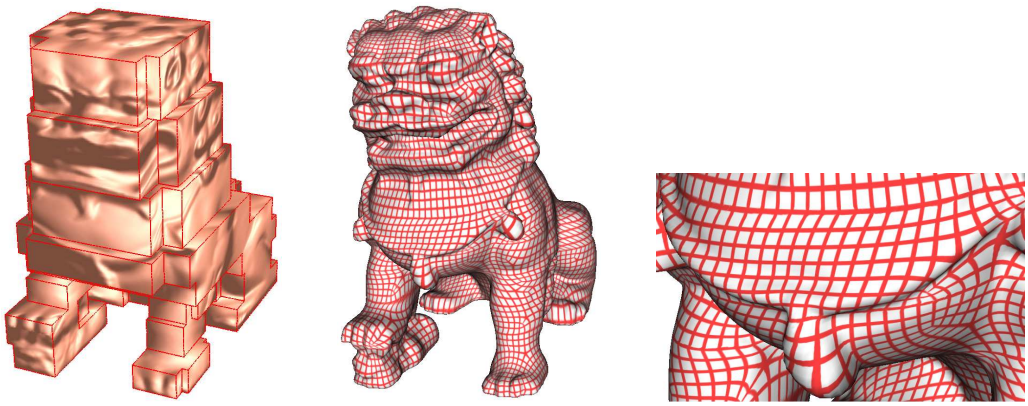
(a) (b) (c) (d) (e) (f)

**Figure 65:** Automatic polycube map construction for the genus-5 Decocube model. Because of symmetry, there are only five distinct z-values among the critical points (see (a)), so four cutting planes are used to slice the model $M$ and $P$ into 8 components, each of which is a genus-zero open surface. We construct the one-to-one map $\phi$ between $P_i$ and $M_i$, $i = 1, \cdots, 8$, respectively (see (b) to (f)). Note that each cutting boundary (instead of the boundary in the original shape) appears in two adjacent components. Since we use the consistent parameterization between $\partial P_i$ and $\partial M_i$ (see Equation 38), the boundary conditions of the harmonic map of two adjacent components are consistent. As a result, the piecewise polycube maps are $C^0$ continuous across the cutting boundaries (red curves), i.e., they are seamless.

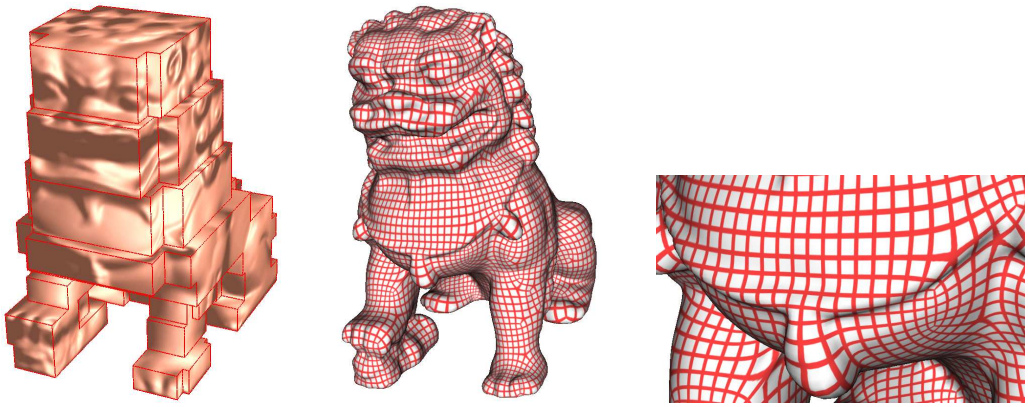


(a) Face charts (b) Edge charts (c) Corner charts

**Figure 66:** A polycube is covered by face, edge, and corner charts. Each face chart (drawn in blue) covers only the interior points of the corresponding face and leaves off all the boundary edges of the face. Each edge chart (drawn in red) covers the interior points of the edges but leaves off corner vertices. Each corner chart (drawn in yellow) covers the corner.



(a) Before improvement



(b) After improvement

**Figure 67:** Improving the polycube map by computing the harmonic map for the entire shape. The continuities across the cutting boundaries before and after improvement are $C^0$ and $C^\infty$, respectively. The angle distortions before and after the improvement are 1.141 and 1.028, respectively. Please pay attention to the quality improvement on the conformality of the checkerboard texture mapping.

For any vertex $v \in U_e$ on the polycube edge, its neighbors are on two different polycube faces. Function $\psi_e : U_e \to \mathbb{R}^2$ is defined by rotating one attached polycube face 90 degrees (i.e., making $v$ and its neighbors co-planar) followed by a projection along the normal of the un-rotated polycube face.

For any vertex $v \in U_c$ on the polycube corner, its one-ring neighbors are on three or five different polycube edges. Function $\psi_c : U_c \to \mathbb{R}^2$ maps $v$ to the origin and its one-ring neighbors to uniformly distributed points on a unit circle.

Let $\phi : P \to M$ denote the constructed piecewise polycube map and $\phi^{-1} : M \to$

**Table 5:** Statistics of the experimental results. Test models are scaled to a unit cube. $g$, genus; # $\Delta$, number of triangles in the given shape; $d_a$, the area difference between the axis-aligned contours and the curved intersection contours; $d_z$, the maximal distance between two adjacent cutting planes; $\varepsilon_{angle}$, angle distortion; $\varepsilon_{area}$, area distortion; $T$, execution time measured in minutes.

| Model | $g$ | $d_z$ | $d_a$ | # $\Delta$ | $\varepsilon_{angle}$ | $\varepsilon_{area}$ | $T$ |
|---|---|---|---|---|---|---|---|
| Amphora | 2 | 0.06 | 0.3 | 125K | 1.015 | 1.128 | 12 |
| Bimba | 0 | 0.18 | 0.3 | 200K | 1.014 | 1.143 | 9 |
| Buddha | 6 | N/A | N/A | 300K | 1.051 | 1.316 | 28 |
| Bunny | 0 | 0.08 | 0.3 | 34K | 1.026 | 1.127 | 5 |
| Dancer | 1 | 0.05 | 0.25 | 186K | 1.032 | 1.119 | 19 |
| Dragon | 0 | 0.1 | 0.25 | 200K | 1.028 | 1.118 | 20 |
| Decocube | 5 | 0.25 | 0.3 | 60K | 1.026 | 1.089 | 3 |
| Fertility | 4 | 0.08 | 0.3 | 100K | 1.020 | 1.148 | 19 |
| Gargoyle | 0 | 0.08 | 0.3 | 75K | 1.021 | 1.155 | 11 |
| Greek | 4 | 0.05 | 0.3 | 200K | 1.034 | 1.082 | 23 |
| Kitten | 1 | 0.08 | 0.2 | 134K | 1.045 | 1.153 | 12 |
| Laurana | 0 | 0.25 | 0.3 | 125K | 1.003 | 1.122 | 10 |
| Rabbit | 0 | 0.2 | 0.2 | 27K | 1.032 | 1.142 | 2 |
| Sheep | 0 | 0.14 | 0.3 | 200K | 1.004 | 1.191 | 18 |
| Squirrel | 0 | 0.06 | 0.2 | 144K | 1.004 | 1.125 | 14 |
| Totem | 0 | 0.08 | 0.25 | 217K | 1.025 | 1.055 | 23 |

$P$ the inverse map. Then we solve a harmonic map for the face, edge and corner charts, respectively. We consider the corner chart in the following, and the edge and face charts can be handled in a similar fashion.

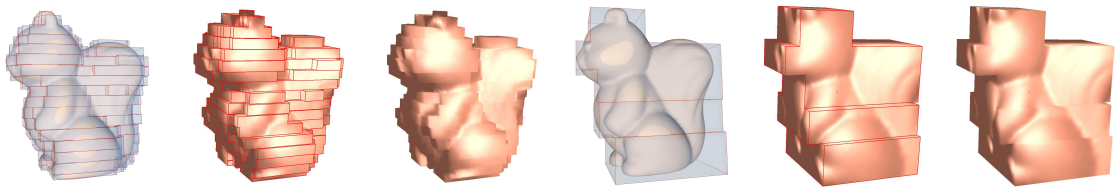
Given a point $v \in U_c$ on the polycube corner, let $p = \phi(v) \in M$ denote the point on the 3D model $M$. The composite map $\psi \circ \phi^{-1} : M \to \mathbb{R}^2$ maps a 3D point $p$ and its neighborhood to the planar domain. We can solve a harmonic map $h : \phi(U_c) \to \mathbb{R}^2$

$$\triangle h(p) = \sum_{q_i \in Nb(p)} \omega_i (h(p) - h(q_i)) = 0, \tag{40}$$

where $Nb(p)$ is the set of one-ring neighbors of $p$ and $\omega_i$ is the *cotan* weights induced by the metric of the given mesh $M$. The vertices on the boundary of corner chart $\partial U_c$ are fixed, i.e., $h(\phi(\partial U_c)) = \psi(\partial U_c)$.

Solving harmonic map for each individual face, edge and corner chart significantly improves the conformality for each chart and the charts cover the whole

polycube domain, thus, the quality of the polycube map can be improved significantly. We should point out that all the cutting boundaries are entirely covered by the face charts, as a result, the resulting polycube map has $C^\infty$ continuity along the cutting locus as demonstrated in Fig. 67.



(a) (b) (c) (d) (e) (f)

**Figure 68:** The user can easily control the shape of the polycube by specifying two parameters, $d_z$, the maximal distance between two consecutive cutting planes, and $d_a$, the area difference between the axis-aligned contours and the curved intersection contours. The parameters for the Squirrel model are $d_z = 0.06$, $d_a = 0.2$ ((a) to (c)) and $d_z = 0.16$, $d_a = 0.3$ ((d) to (f)). The model is scaled to a unit cube.

## 6.3 Experimental Results

We conducted extensive tests of our algorithm over a large variety of models ranging from genus zero to genus six. Computation time were measured in minutes on a workstation with 3.0GHz CPU and 3GB memory. Among all of the five steps in Section 6.2, computing the uniform flat metric takes nearly 80% of the entire time. Within our framework, the user may choose to simply specify two parameters, $d_z$, the maximal distance between two adjacent scanning planes, and $d_a$, the threshold of the normalized area difference between the axis-aligned polygons on $P$ and the curved intersection contours on $M$ (see Section 6.2.2 for the details). Figure 68 shows how the user can easily control the shape of the polycube by specifying the above two parameters. The quality of the polycube map is measured by the angle distortion $\varepsilon_{angle}$ and area distortion $\varepsilon_{area}$ [29],

$$\varepsilon_{angle} = \sum_i \frac{\cot\alpha a^2 + \cot\beta b^2 + \cot\gamma c^2}{4A(\triangle_i)} A(\phi(\triangle_i)), \tag{41}$$

$$\varepsilon_{area} = \frac{1}{2}\sum_i \left(\frac{A(\triangle_i)}{A(\phi(\triangle_i))} + \frac{A(\phi(\triangle_i))}{A(\triangle_i)}\right) A(\phi(\triangle_i)), \tag{42}$$

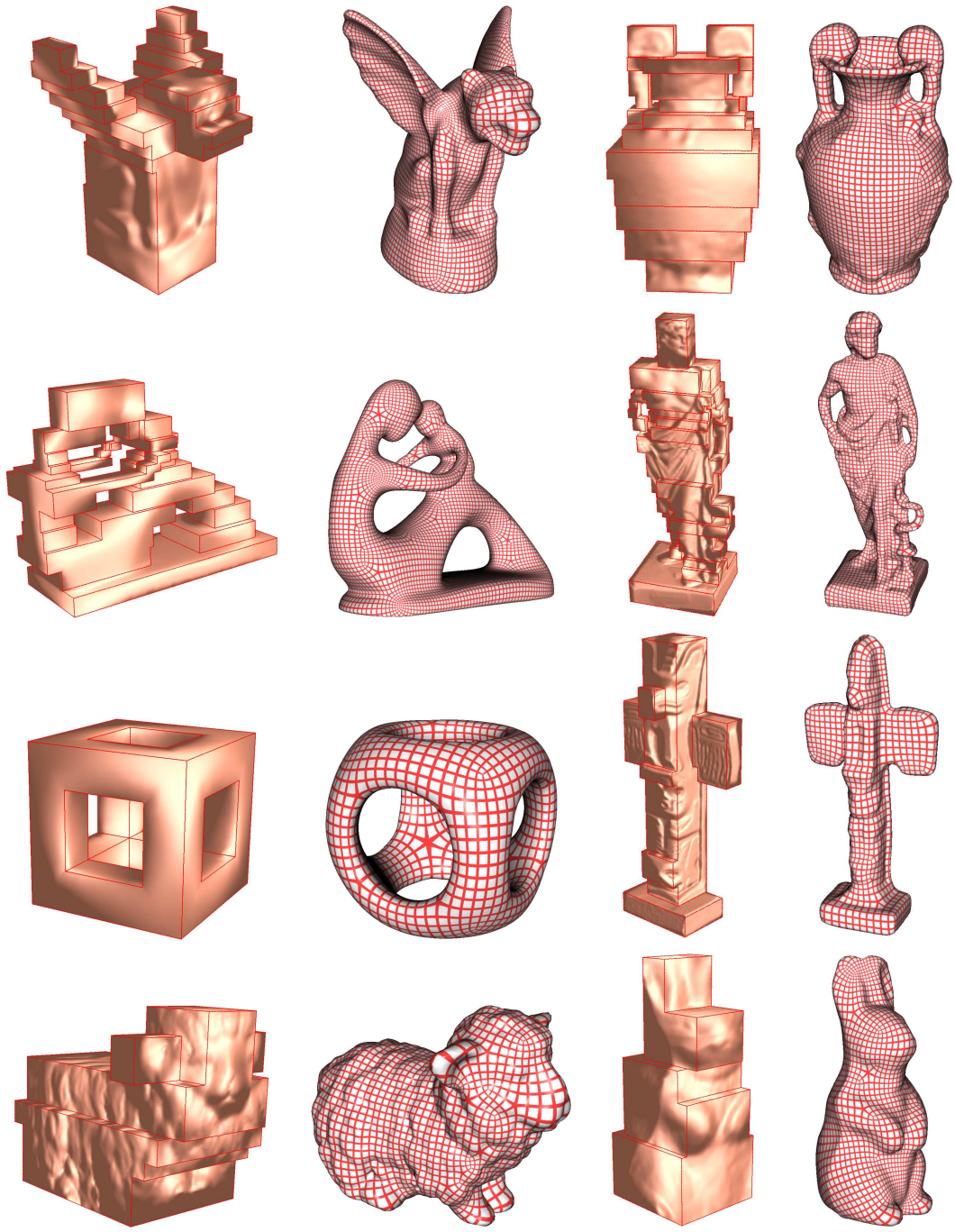


**Figure 69:** Automatically constructed polycube maps of complicated topology and geometry.

where $\triangle_i \in P$, $\phi(\triangle_i) \in M$, $a,b,c,\alpha,\beta,\gamma$ are the side length and angles of $\triangle_i$, and $A(\cdot)$ denotes the area. In the isometric map, $\varepsilon_{angle} = 1$ and $\varepsilon_{area} = 1$. Therefore, the closer the values of $\varepsilon_{angle}$ and $\varepsilon_{area}$ to 1, the better the quality of the constructed polycube maps. The statistics and performance of test cases are reported in Table 5, whereas the corresponding constructed polycube maps are shown in Figure 69. Note that our method can produce polycube maps with very small area and angle distortions.

We have applied the constructed polycube maps to a wide range of applications, such as quadrilateral mesh generation, T-spline construction, seamless texture synthesis, and volumetric parameterization, as demonstrated in Fig. 70 and 71.

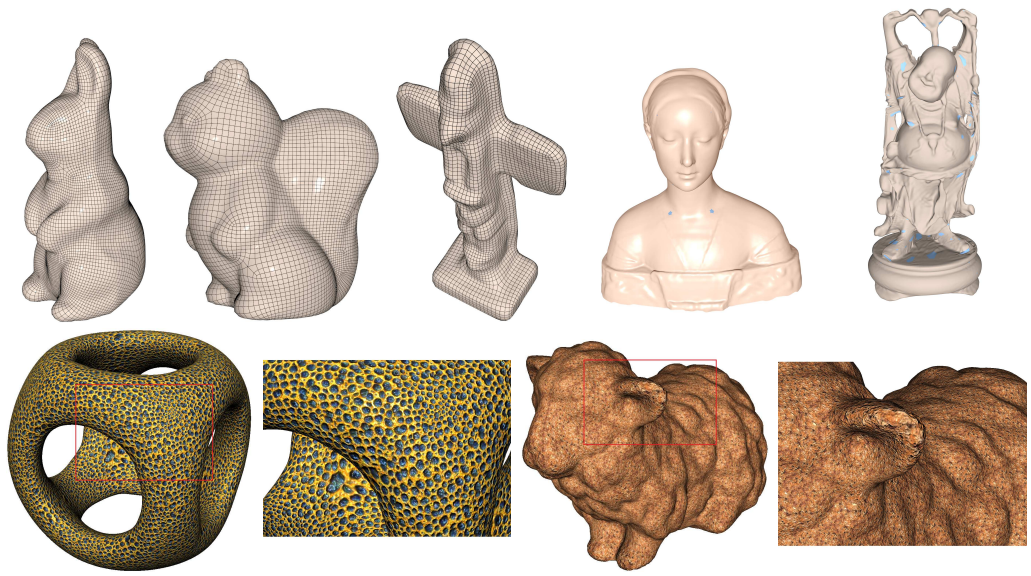


**Figure 70:** Polycube maps applied to quadrilateral remeshing, T-splines and tile-based texture synthesis.

## 6.4 Comparisons

In this section, we compare our method with the existing approaches and show its advantages and disadvantages. Table 6 summarizes the key differences between our new approach and the existing methods.

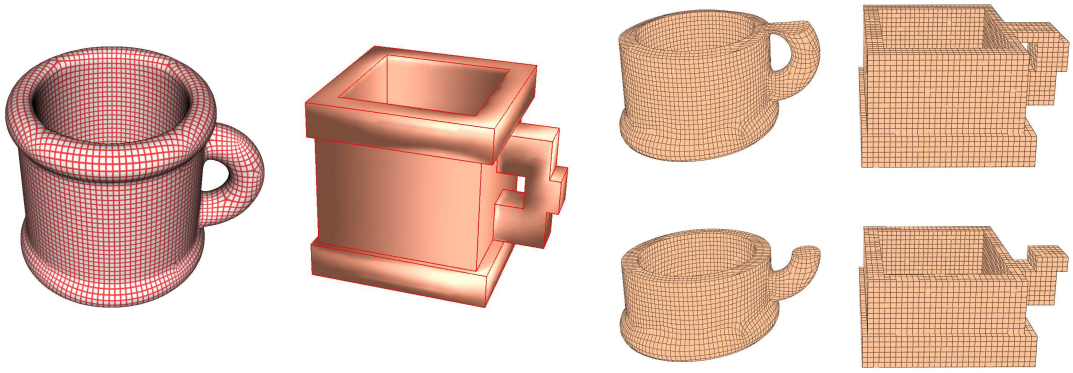


**Figure 71:** Polycube serves a natural parametric domain for volumetric parameterization. Therefore, we can generate all-hexahedral meshes without any extraordinary points and T-junctions [77].

**Table 6:** Comparison with existing polycube map construction techniques.

| Methods | Polycube Construction | Bijective | Performance | Limitation |
|---|---|---|---|---|
| Tarini *et al.* [142] | Manual (The polycube should mimic the given shape) | No | Efficient | Difficult for surfaces with complicated topology and geometry due to vertex projection from the given shape to the polycube |
| Wang *et al.* [147] | Manual (The polycube can differ from the given shape significantly) | Yes | Computationally expensive | Difficult for surfaces with complicated topology and geometry due to the numerical unstableness in hyperbolic embedding |
| Wang *et al.* [148] | Manual (The user directly specifies the polycube structure on the given shape) | Yes | Many user interactions | Not practical for surfaces with complicated topology |
| Lin *et al.* [101] | Automatic (The user specifies several parameters to control the Reeb graph embedding and surface segmentation) | N.A. | Efficient | Not practical for surfaces with complicated topology and geometry |
| Our method | Automatic (The user may set two parameters to specify how close the polycube mimics the given shape) | Yes | Efficient | Non-axis-aligned branches or handles will usually result in a geometrically complicated polycube |

### 6.4.1 Comparison with [142]

In [142], Tarini *et al.* first constructed the polycube manually and then warped the polycube close to the given mesh. Next, the vertices on the given mesh are projected onto the warped polycube. Finally, the polycube is warped back. This method is *extrinsic*, since it requires the projection of the vertices of the input shape $M$ to the polycube domain $P$. Therefore, this method requires the user to design the polycube $P$ manually and carefully such that it closely resembles the geometry of the input shape $M$, otherwise, it is difficult to warp the polycube close to $M$ and the resulting polycube map may not be bijective. Our method is *intrinsic* in that it guarantees the bijection between the given shape and the polycube. Figure 72 shows the comparison between Tarini *et al.*'s method and our method on the Laurana model.

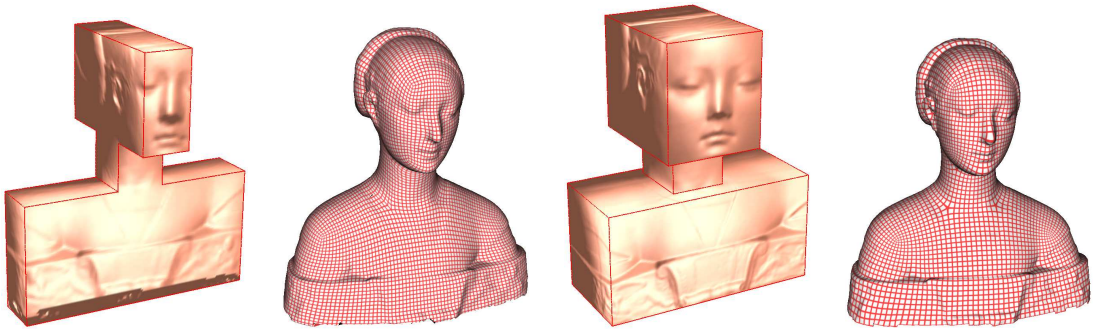


**Figure 72:** Comparison with Tarini *et al.*'s method [142], where the polycube is constructed manually to mimic the given shape (Data courtesy of Dr. Marco Tarini). The angle and area distortions of the polycube map are $\varepsilon_{angle} = 1.102$, $\varepsilon_{area} = 1.140$ (Tarini *et al.*'s method [142], left) and $\varepsilon_{angle} = 1.003$, $\varepsilon_{area} = 1.122$ (our method, right), respectively. Note that our new method is more flexible in that the user can easily control the shape of polycube and reduce the area and angle distortion.

### 6.4.2 Comparison with [147]

Following Tarini *et al.*'s pioneering work, Wang *et al.* proposed an *intrinsic* method to construct a polycube map [147]. Instead of computing the map between the polycube $P$ and input shape $M$ directly, both $P$ and $M$ are first embedded into one of the three canonical domains, $\mathbb{S}^2$, $\mathbb{E}^2$, or $\mathbb{H}^2$, depending on the topology of $M$, i.e., $\pi_M : M \to D_M$ and $\pi_P : P \to D_P$ using uniformization metric, i.e., the Gaussian

curvature is constant everywhere. Then by seeking the one-to-one map between the two domains $\phi_{D_M \to D_P} : D_M \to D_P$, the composition $\phi_{M \to P} = \pi_P^{-1} \circ \phi_{D_M \to D_P} \circ \pi_M$ is the desirable polycube map from $M$ to $P$. This method is intrinsic in that it avoids the vertex projection from $M$ to $P$. However, it is known that embedding a surface with negative Euler characteristic into $\mathbb{H}^2$ is error-prone when the point is very close to the boundary of the Poincaré disk due to the numerical rounding error. Therefore, Wang *et al.*'s method is not practical and much less numerically stable to construct polycube maps of large-scale models with negative Euler characteristics. Our method uses a divide-and-conquer approach which avoids computing the uniformization metric. Note that from the point of view of numerical computation and its robustness and stability, embedding the genus-0 open surface into $\mathbb{R}^2$ using *uniform flat metric* is much more robust than embedding a surface with negative Euler characteristic into hyperbolic space $\mathbb{H}^2$ using uniformization metric. Figure 73 compares Wang *et al.*'s method [147] and our method on the Bimba model.

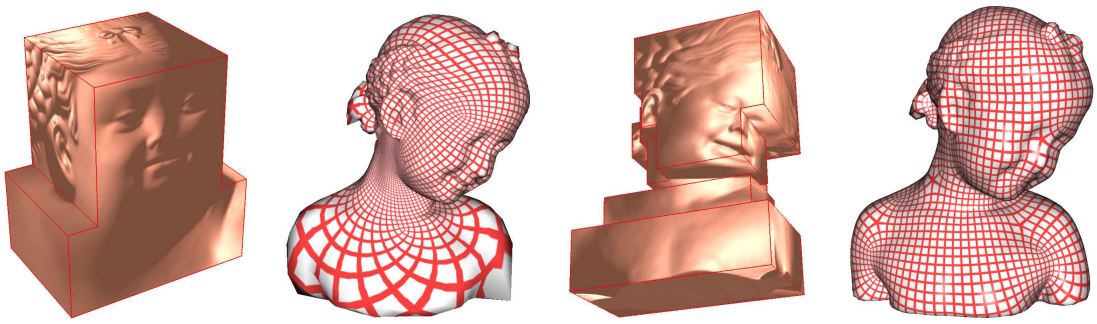


**Figure 73:** The left two columns show Wang *et al.*'s method [147] on Bima model with distortions $\varepsilon_{angle} = 1.052$ and $\varepsilon_{area} = 5.145$. The right two columns show the results using our automatic method, where $\varepsilon_{angle} = 1.014$ and $\varepsilon_{area} = 1.143$. Note that the checkerboard texture mapping of our method is much more uniform than that of Wang *et al.*'s approach.

### 6.4.3 Comparison with [148]

Wang *et al.* proposed an interactive method to improve the polycube map for high genus surfaces [148]. The key difference between this user-controllable

method and [147] is that users have full freedom to specify the number and locations of the singularities (the pre-images of polycube corners) on $M$ and their connectivity, i.e., which pair of corners forms a polycube edge, which set of polycube edges form the polycube face, etc. This method avoids the global parameterization and can nicely produce polycube maps. However, manually specifying the polycube structure on the given mesh $M$ is tedious and sometimes not feasible even for expert users with deep geometric insight and broad topological knowledge. For example, the minimal number of singularities for the genus-5 Decocube model is 48. It is rather time consuming and error-prone to specify both the locations and connectivity of the singularities on the input model of genus-5 for 48 points. Our method is more flexible in that the user plays with the parameters to specify how the polycube mimics the given shape and then produces the polycube map with low area and angle distortion. As demonstrated in Fig. 69, our method is capable of computing high-quality polycube maps for surfaces of complicated geometry and topology.

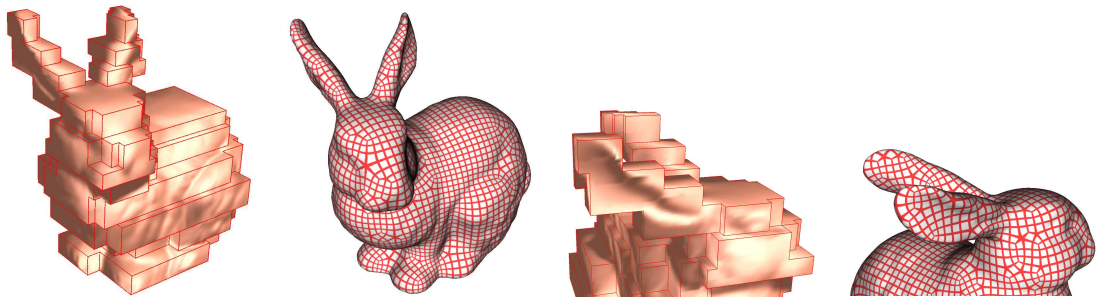


**Figure 74:** The proposed method may generate a geometrically complicated polycube for non-axis-aligned long branches or handles, such as the Bunny ears. As a result, the polycube has a large number of extraordinary points (polycube corners).

### 6.4.4 Comparison with [101]

Most recently, Lin *et al.* proposed an automatic method to construct polycube maps [101]. They first segmented the 3D model using Reeb graph and then approximate the polycube into several polycube primitives, i.e., cube, L-, O-, and U-shapes. They demonstrated their approach on bunny, 3-hole torus and horse models. However, Lin *et al.*'s approach may not work for the surfaces with complicated

topology and geometry. For example, if the Reeb graph has a node whose degree is more than 6, (.e.g, the hub of a bicycle wheel is attached with many spokes) then it is impossible to use the above polycube primitives to approximate the shape. Note that our approach can generate a polycube for this case, but may have large number of extraordinary points, which will be discussed in the next section. Furthermore, there is no guarantee that Lin *et al.*'s approach produces a bijection. According to the report in [101], the angle and area distortion of Bunny model is 1.12 and 1.15, respectively. Our approach results in polycube map with smaller angle and area distortion 1.026 and 1.127 (see Fig. 74).

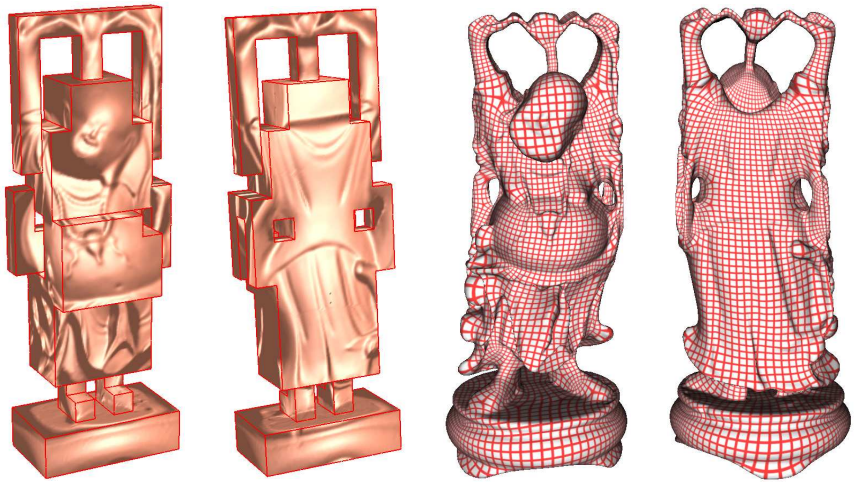


**Figure 75:** Our method also applies to manually constructed polycubes.

## 6.5 Discussions

### 6.5.1 Manually vs. Automatically Constructed Polycubes

In the existing techniques of constructing polycube maps [142] [147] [148], the polycube maps are constructed manually. Although manual constructions work well for the models with simple topology, it is extremely tedious and time consuming to construct polycube with complicated topology. The proposed approach

(Sec. 6.2.2) can generate polycubes for complicated topology and geometry. However, it usually generates polycubes which are more complicated (based on the number of corners and faces) than the manually-built ones. We should also point out that the current polycube construction stage can be simplified/replaced by any alternative method (either automatic or manual approach) in order to produce a polycube with less complexity. Figure 75 shows the genus-6 Happy Buddha model, whose polycube approximation is constructed manually. Note that using our new method we can still construct the high-quality polycube map automatically and efficiently, while accommodating the varying complexity.

### 6.5.2 Bijection of Our Automatic Polycube Map

In this section, we show that our method generates a bijection between the polycube and 3D model. Here we assume that given the user specified parameters $d_a$ and $d_z$, a *valid* polycube $P$ is constructed in step 2.

Given a close surface of genus $g$, we solve the Laplace's equation using the z-coordinate of the top-most and bottom-most points as the boundary condition in step 1. The Laplace's equation results in $2g$ saddle points. For each handle, one saddle point corresponds to the case in which the handle splits, and the other saddle point corresponds to the case in which the handle merges.

In step 2, the saddle points are sorted in the ascending order of z-coordinate. Then we construct $2g+1$ horizontal cutting planes,

$$\begin{aligned} z_0 &= \frac{z(v_0)+z(c_1)}{2} \\ z_1 &= \frac{z(c_1)+z(c_2)}{2} \\ &\cdots \\ z_{2g} &= \frac{z(c_{2g})+z(v_1)}{2} \end{aligned}$$

Note that using the top and bottom cutting planes $z_0$ and $z_{2g}$, the model will be segmented into 3 parts, the top part (a genus-0 open surface with 1 boundary), the middle part (a genus-$g$ open surface with 2 boundaries), and the bottom part (a genus-0 open surface with 1 boundary). Next, $2g-1$ cutting planes, $z_1, \cdots, z_{2g-1}$, are used to slice the middle part into $2g$ layers, each of which contains a set of

disjoint genus 0 surfaces with at least 2 boundaries. Note that if two or more saddle points are on the same cutting plane, the number of layers decreases.

In step 3, we compute the uniform flat metric of each segmented component $P_i$ or $M_i$ using discrete Ricci flow. Ricci flow is theoretically sound to guarantee the diffeomorphism between the genus-0 surface and the multi-hole disk.

In step 4, we construct the harmonic map between $P_i$ and $M_i$ using the following commutative diagram.

$$\begin{array}{ccc} P_i & \xrightarrow{\phi_{P_i\to M_i}} & M_i \\ \downarrow{\scriptstyle \phi_{P_i\to D_{P_i}}} & & \downarrow{\scriptstyle \phi_{M_i\to D_{M_i}}} \\ D_{P_i} & & D_{M_i} \\ \downarrow{\scriptstyle \phi_{D_{P_i}\to \bar{D}_{P_i}}} & & \downarrow{\scriptstyle \phi_{D_{M_i}\to \bar{D}_{M_i}}} \\ \bar{D}_{P_i} & \xrightarrow[\phi_{\bar{D}_{P_i}\to \bar{D}_{M_i}}]{} & \bar{D}_{M_i} \end{array} \tag{43}$$

The uniform flat metric computed using discrete Ricci flow is guaranteed to induce a diffeomorphism between $P_i$ to $D_{P_i}$ (and $M_i$ to $D_{M_i}$) [16]. Note that a harmonic map $f : A \subset \mathbb{R}^2 \to B \subset \mathbb{R}^2$ is a diffeomorphism if $\partial B$ is convex and the boundary condition $f(\partial A) = \partial B$ is a homeomorphism. Since $\partial \bar{D}_{P_i}$ and $\partial \bar{D}_{M_i}$ are circular, $\phi_{D_{P_i}\to \bar{D}_{P_i}}$, $\phi_{D_{M_i}\to \bar{D}_{M_i}}$, and $\phi_{\bar{D}_{P_i}\to \bar{D}_{M_i}}$ are diffeomorphism. Then the piecewise polycube map $\phi : P \to M$ is given by $\phi = \bigcup_i \phi_{P_i\to M_i}$.

In step 5, we further improve the polycube map quality by solving the harmonic maps for face, edge, and corner charts respectively. The polycube $P$ is covered by face, edge and corner charts, $\{U, \psi\}$, where $U \in P$ is an open set of $P$ and $\psi : U \to \mathbb{R}^2$ maps $U$ to the planar domain (see Fig. 66). The definition of $\psi$ is given in Section 6.2.5.

Given a point $v \in U$ on a chart, let $p = \phi(v) \in M$ denote the point on the 3D model $M$. Then the composite map $\psi \circ \phi^{-1} : M \to \mathbb{R}^2$ maps a 3D point $p$ to the planar domain. We solve a Laplace's equation $h : \phi(U) \to \mathbb{R}^2$ such that

$$\triangle h(p) = \sum_{q_i \in Nb(p)} \omega_i (h(p) - h(q_i)) = 0,$$

where $Nb(p)$ is the set of one-ring neighbors of $p$ and $\omega_i$ is the cotan weights

induced by the metric of the given mesh $M$. The boundary conditions are given by

$$h(\phi(\partial U)) = \psi(\partial U).$$

For the edge and corner charts, $\psi$ maps $\partial U$ to a rectangle and a unit circle, respectively. Note that $\psi(\partial U)$ is a homeomorphism and the boundary of $\psi(\partial U)$ is convex, and thus, $h$ is a diffeomorphism.

For the face charts, $\psi$ is defined as an orthogonal projection of the polycube face along the normal direction. Thus, $\psi$ maps $\partial U$ to itself and $h$ is also a diffeomorphism.

Since the improved map $h$ is a diffeomorphism for every point inside the face, edge and corner charts, and a homeomorphism for the points on the boundary of face, edge and corner charts. Thus, $h$ induces a bijection between the polycube $P$ and the 3D model $M$.

### 6.5.3 Limitations

Our proposed method has certain limitations and demands further improvement in the future. First, the constructed polycube depends on the orientation of the 3D model. Different orientations may result in very different polycubes. In our implementation, we require the user to align the model before the polycube construction. Second, the proposed method will generate a geometrically complicated polycube for non-axis-aligned long branches or handles, such as the ears of the bunny model (see Fig. 74). As a result, it may cause difficulty in some applications, e.g., spline construction, since each corner of polycube is an extraordinary point. Third, the polycube construction relies on the user inputs, i.e., $d_a$ and $d_z$. For a shape with complicated geometry and topology, the global parameters may not generate a valid polycube. Thus, the local adaptive parameters must be used, which will result in a much more complicated implementation.

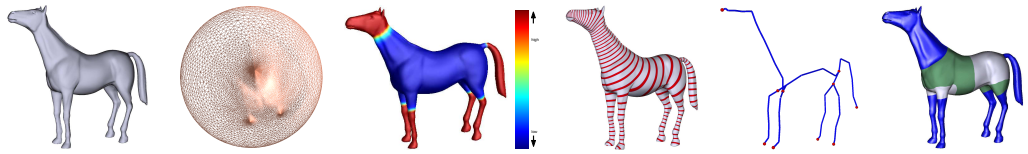
# Chapter 7

# Geometry-Aware Domain Decomposition

## 7.1 Introduction and Motivation

Manifold T-splines presented in [75] is a natural and necessary integration of T-splines and manifold splines, which naturally extends the concept and the currently available algorithms/techniques of the popular planar tensor-product NURBS and T-splines to arbitrary manifold domain of any topological type. It can be directly defined over the manifold of arbitrary topology to accurately represent various shapes with complicated geometry/topology, and naturally inherits all the attractive properties from T-splines defined over a planar domain, including the powerful local refinement capabilities and the hierarchical organization for LOD control. Despite this earlier success, certain drawbacks of manifold T-splines still remain: (i) There must be singularities for any closed manifold except tori, and in practice small holes must be punched around the singularities in order to enable the easy construction of manifold splines in the finite dimension space. No efforts for hole-filling in the vicinity of singular points were made in [75]; (ii) It is impossible to specify the locations of all the singularities on the domain manifold given the fact that the number of singularities is actually fixed, but their positions are somehow globally related; (iii) The proposed domain construction method is far from

sufficient for surfaces with boundaries or surfaces with long branches. For surfaces with long branches (for example the horse model in Figure 76), the existing global parameterization methods usually introduce extremely large area distortion and therefore make it even harder and numerically unstable for spline fitting process later on. The only feasible way is to introduce additional cuts in these areas to make it a surface with boundaries and then use double covering method to achieve a better parameterization result. However, this technique will at least double the time complexity and not practical for a large scale complex dataset.



(a) (b) (c) (d) (e) (f)

**Figure 76:** Genus-zero horse model with long branches. (b) shows the spherical conformal map of the genus-zero closed surface shown in (a). (c) highlights the area distortion using colormap. The Reeb graph of the given model shown in (e) is computed based on the harmonic function $f$ defined on the given surface. (d) shows the isolines of $f$ in red. The segmentation result based on the Reeb graph representation is shown in (f): four base patches (colored in green and gray), and six long branches (colored in blue).

Polycube T-splines proposed by us in [147] unifies T-splines and manifold splines to define a new class of shape representations for surfaces of arbitrary topology by using polycube map as its parametric domain. Instead of further reducing the number of singular points as Gu et al's work in [58], we aimed to reduce the total area distortion of the parameterization by introducing more singular points (corners of the polycube) to facilitate a better spline surface fitting. In [148] we advanced our work by introducing the user's interaction into the process of polycube map construction. By allowing the user to directly select the corner points of the polycubes on the original 3D surfaces in an interactive manner, the location of singularities of the polycube map can be interactively controlled. Therefore, the subsequent hole-filling process and better data-fitting results can be easily accomplished by placing the singularities at regions where no rich geometric features exist. However, our interactive polycube map construction framework has the following limitations: (i) Domain knowledge from users is required to select a feasible set of corner points

which gives rise to a polycube map with high quality (that has small angle distortion and area distortion); (ii) The resulting polycube map is $C^0$ continuous across polycube edges that connect corners, which may introduce unpleasing results for later spline surface fitting; (iii) It is not possible to properly handle surfaces with long and thin branches (refer to Figure 76 for an example) because users can not easily specify corner points in long and thin branches and the computation of straight lines connecting corner points in these parts will be numerically unstable and error-prone.

In this work our objective is to further improve the existing work in data modeling, which overcomes the aforementioned drawbacks, and is much more efficient, robust, and applicable in real-world applications and industrial CAD environments. We propose a geometry-aware framework for manifold T-spline construction, which first decomposes any given surface into three categories: long branch (genus-zero patch with one boundary), handle (genus-zero patch with three boundaries) and base patch (genus-zero patch with at least three boundaries) by using the pants decomposition method and exploiting the skeleton representation of the surface, then locally parameterize all of these patches into regular domains using Ricci flow, build the domain manifold for each patch independently, and finally glue them together to form a complete domain manifold for later spline fitting. Note that, the T-junctions are allowed along the patch boundaries to ensure certain continuity. The proposed construction pipeline is extremely flexible: (i) it can be made fully automatic, which is therefore very useful and applicable in industrial settings; (ii) users' interaction is also allowed (refer to section 7.3 for details) during the process to arrive at a result they prefer. Figure 76 shows the horse model with long thin branches, which is very difficult to handle by using existing data modeling techniques, but can be handled elegantly by our new method. Figure 83 shows the manifold T-spline surface for this model constructed by using our proposed algorithm.

The specific contributions of this work are as follows:

1. We provide a systematic way to segment any given surface into three categories (branches, handles, and base patches), and handle each category using different strategies to ensure high-quality parameterization and fitting results. Object segmentation and local parameterization enhance the system's flexibility while improving time/space performance by avoiding time-consuming and error-prone global parameterization.

2. We show that the number of extraordinary points equals $2 * n_{branch} + n_{handle} + 2 * n_{base}$ and the resultant spline surface is $C^2$ everywhere except $C^1$ at the extraordinary points.
3. The entire object segmentation always leads to a set of four-sided patches for any input surface with diverse topological types. Tensor-product B-splines or T-splines are naturally serving as basic building blocks, bridging the large gap between NURBS-centric existing CAD software in industry and manifold surface modeling algorithms.
4. The entire construction pipeline is flexible: it can be made fully automatic, which makes the proposed framework very valuable in industrial settings. Users' interaction can also be enabled in certain parts of the pipeline to lead to a user-controllable object segmentation and local parameterization that respects both feature alignment and geometric constraints simultaneously.
5. The entire data processing pipeline enables the flexible and accurate modeling of manifold surfaces within the currently-available industrial CAD environment. The rectangular structure of each modeled piece completely avoids the trimming operation, while ensuring the "one-piece" representation for manifold surfaces satisfying high-order continuity requirements.

The remainder of this chapter is organized as follows. We review the related work on skeleton extraction, and handle/tunnel loops computation in Section 7.2. In Section 7.3 we present the detailed algorithms for our geometry-aware domain decomposition pipeline. Finally, experimental results are demonstrated in Section 7.4.

## 7.2 Related Work

This section briefly reviews prior research on handle/tunnel loop computation and skeleton extraction. Refer to Chapter 2.1 and Chapter 3 for previous work on parameterization and splines.

### 7.2.1 Handle/Tunnel Loop Computation

The handle and tunnel loops can be defined as follows (see also [31] for the definition): a loop $b_i$ on a surface $M$ is a *handle* if it spans a disk in the bounded space $\mathbb{I}$; if one cuts $M$ along $b_i$ and fills the boundary with that disk, one eliminates a handle. A loop $a_i$ on a surface $M$ is a *tunnel* if it spans a disk in the unbounded space $\mathbb{O}$, whose its removal eliminates a tunnel. These loops characterize important topological information of the surface, and automatic detection of these loops are necessary in many applications such as topology repair of 3D models, surface parameterization , and feature recognition.

Various algorithms for computing different types of non-trivial loops on surfaces have been proposed in recent years. They either do not guarantee detecting handle and tunnel loops [37] [15] [36]; or need some graph structures built from the input model to compute the handles and tunnels such as Reeb graph [134], medial axis [153], or curve skeletons [31]. More recently, Dey et al. proposed a persistence based algorithm to compute well defined handle and tunnel loops for a 3D model in [32]. The algorithm provides a mathematical guarantee on detecting handle and tunnel loops and does not require computing any extra structures.

### 7.2.2 Skeleton Extraction

Curve-skeletons are 1D structures that represent a simplified version of the geometry and topology of a 3D object. The extraction of curve-skeletons from 3D models is a fundamental problem in computer graphics and visualization, which has received a lot of attention in recent decades. We refer the readers to [19] for a detailed overview of curve-skeleton properties, applications and algorithms.

Methods for curve-skeleton extraction can be classified into two main categories, volumetric and geometric, depending on whether an interior representation or only the surface representation is used [6]. Most existing curve-skeleton extraction methods make use of a volumetric discrete representation, either a regularly partitioned voxelized representation [102, 113, 150] or a discretized field function defined in the 3D space [69, 154]. They share the common drawbacks of potential loss of details and numerical instability caused by inappropriate discretization resolution. Geometric methods work directly on polygonal meshes or point

sets [4, 6, 7, 33]. Reeb-graph-based methods are also geometric approaches which have gained much attention in recent years. The Reeb graph [123] is a fundamental data structure that captures the topology of a compact manifold by following the evolution of the level sets of a real-valued function defined on the respective manifold. It is obtained by contracting to a point the connected components of the level-sets of a function defined on a mesh. A lot of algorithms have been proposed to compute Reeb graph of an object using various real-value functions. Aujay et al. [7] proposed a harmonic Reeb graph that uses the harmonic function, found by solving the Laplace equation. A robust on-line algorithm for computing Reeb graphs was presented in [114].

## 7.3 Algorithm

### 7.3.1 Algorithm Overview

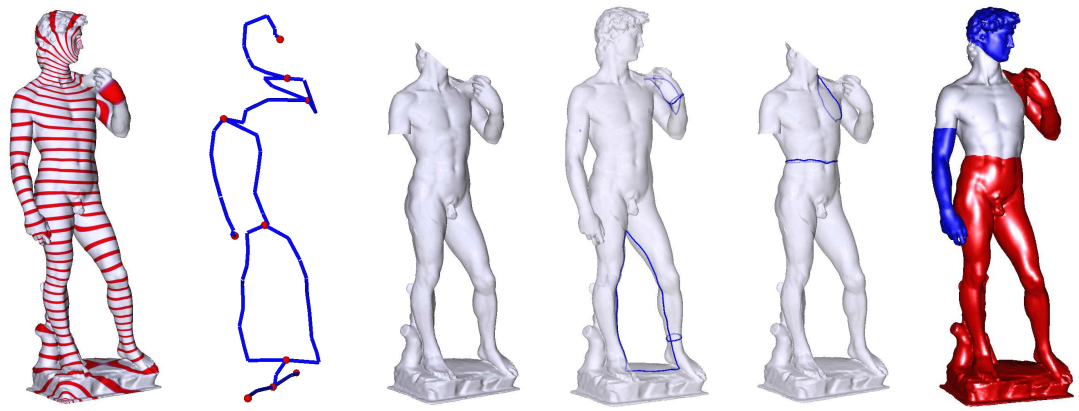
As discussed in Section 7.1, the key idea of our proposed approach is the geometry-aware object segmentation, by which the given surface is first decomposed into a group of disjoint components: branches, handles and base patches. We then apply conformal parameterization and construct the domain manifold for each individual component. Finally the domain manifold for each component can be glued together to form a complete domain manifold followed by a global relaxation for later spline fitting. The proposed construction pipeline is flexible and robust: it can be made fully automatic, which makes the proposed framework very valuable in industrial settings. Users' interaction can also be enabled during certain parts of the pipeline to lead to a user-controllable object segmentation and local parameterization that respects both feature alignment and geometric constraints.

The manifold T-spline construction pipeline for a given surface $M$ is as follows:

1. Segment the surface to branches, handles and base patches.
2. Parameterize branches, handles and base patches using discrete Ricci flow.
3. Construct the domain manifold and set the knots.
4. Fit manifold T-spline and handle extraordinary points.

Figure 77 shows the decomposition procedure for the genus-2 David model. We define different segmented components for a given surface $M$ as follows:

- A *branch* of $M$ is a genus-zero patch with single boundary, which is a region of $M$ corresponding to the arc in its Reeb graph representation with two end nodes of degree-1 and degree-3 respectively(refer to section 7.3.2 for details). A *long branch* of $M$ with respect to a given threshold $\varepsilon$ is the branch of $M$ with arc length (in its Reeb graph representation) longer than $\varepsilon$ (blue components in Figure 77(f));
- A *handle* of $M$ is a region of $M$ with genus-one and single boundary (red components in Figure 77(f));
- A *base patch* of $M$ is a genus-zero patch with at least three boundaries. By removing all the handles and long branches, the remaining region of $M$ is a base patch (gray component in Figure 77(f)); if the remaining region of $M$ is further decomposed into a set of *pants patches* (refer to section 7.3.3 for details), each pants patch is a base patch of $M$ (green and gray components in Figure 76(f));
- All the branches, handles and base patches of $M$ are disjoint, and their union equals $M$.



(a) (b) (c) (d) (e) (f)

**Figure 77:** Decomposition of genus-2 david model. The Reeb graph of the given model shown in (b) is computed based on the harmonic function $f$ defined on the given surface shown in (a). (c) shows the result after long branches removal. (d) highlights in blue the handle and tunnel loops computed by using the method in [32], and the loops used to remove the handles are shown in blue in (e). (f) shows the decomposition result: two branches colored in blue, one base patch colored in gray, and two handles colored in red.

### 7.3.2 Branch Segmentation

For surfaces with long branches, the existing global parameterization methods usually introduce very large area distortion and therefore make it numerically unstable for spline fitting process later on (refer to Figure 76 for an example). To reduce the parameterization distortion, we first remove long branches from the given surface, and then parameterize them separately.

Reeb graph [123] is an ideal tool to detect the long branches from a given surface. It is a 1D structure whose nodes are critical points (maxima, minima, and saddles) of a real-value function $f$ defined on the model surface. It encodes the topology of the model and can be constructed by contracting the connected components of the isolines (level sets or contours) of $f$ to a point. Given its intrinsic properties Reeb graph becomes an ideal tool for us to find and remove the long branches from a given surface $M$: each branch of $M$ corresponds to an arc in its Reeb graph representation with two end nodes of degree one and degree three, respectively. Given the property that each arc $A$ in the Reeb graph represents a family of contours $C_A$ that do not change topology, we can easily remove the long branch $B$ (suppose its corresponding arc is $A$) from the given surface by cutting the surface along one contour $c \in C_A$ (the set of contours of arc $A$). In practice, the long branches to be removed and the corresponding cut contours used to remove the branches from the given surface can be either specified interactively by the user from its Reeb graph representation; or decided by a preset length threshold $\varepsilon_{length}$, and a removal ratio $r$, such that all the branches with corresponding arc length larger than $\varepsilon_{length}$ will be removed by the given ratio $r$. In the latter way the branch removal process will be automatic.

Many algorithms for Reeb graph computation have been proposed during the recent decades. We adopt the on-line algorithm presented in [114] to compute the Reeb graph presentation of a given surface because of its robustness and scalability. Figure 77(b) shows the Reeb graph of the genus-2 David model based on the harmonic function $f$ shown in Figure 77(a). Figure 77(c) shows the remaining region after removing the long branches.

### 7.3.3 Handle and Base Patch Segmentation

In [97] a consistent pants decomposition algorithm was presented which takes as the input the handle and tunnel loops of the surface, and then segments the given surface into a set of *pants patches* (genus-zero patch with three boundaries) in a consistent manner. In our construction pipeline, we use the similar algorithm to remove the handles from the given surface with long branches removed.

The handle and tunnel loop information is required for automatic pants decomposition of the 3D surfaces [97]. There are various existing algorithms for computing critical loops on surfaces, but many of them do not guarantee detecting handle and tunnel loops. We use the persistence based algorithm proposed by Dey *et al.* in [32] which computes well defined handle and tunnel loops for a 3D model, and guarantees that the resulting handle and tunnel loops are topologically correct and geometrically small. The handle and tunnel loop computation is conducted on the original surface instead of the remaining patch with branches removed since the algorithm in [32] requires that the input surface is a closed one. Figure 77(d) highlights in blue curves the handle and tunnel loops computed by using the algorithm in [32].

Once the indexed $g$ handle and tunnel loops $(a_i, b_i, 0 \leq i < g)$ of the given surface $M$ with genus $g$ are computed, we first map them to the remaining patch $M'$ ($M$ with long branches removed), and then conduct a subsequent decomposition on $M'$ to obtain a set of *handles* (genus-one patch with one boundary) and one *base patch* (genus-zero patch with at least three boundaries). The algorithm is detailed in [97], here we briefly outline the idea:

*Step 1* Slice/remove all handles from $M'$. Repeat the following steps until all handles are removed:

(1.1) Compute a loop bounding the handle-$i$ (topologically, such a loop $c_i = a_i^1 \circ b_i^1 \circ a_i^{-1} \circ b_i^{-1}$).

(1.2) Shrink $c_i$ homotopically to the shortest loop $w_i$ (Figure 78(a), blue loops).

(1.3) Remove the handle-$i$ from $M'$ by slicing the loop $w_i$.

*Step 2* (The remaining patch $M''$ is a topological sphere with at least three holes) Decompose $M''$ into pants patches (Figure 78(b)(c)).

(2.1) Put all boundaries $w_i$ of $M''$ into a queue $Q$.

(2.2) If $Q$ has $\leq 3$ boundaries, end; else goto (2.3).

(2.3) Compute shortest loop $w'$ homotopic to $w_i \circ w_j$.

(2.4) ($w'$, $w_i$ and $w_j$ bound a pants patch $p_{w'}$) Remove $p_{w'}$ from $M''$. Remove $w_i$ and $w_j$ from $Q$. Put $w'$ into $Q$. Goto (2.2).

After step 1, we get a set of handles and one base patch $M''$ which is a genus-zero patch with at least three boundaries. Step 2 is optional in our construction pipeline: we can either parameterize $M''$ directly, or further decompose it into a set of *pants patches* using the algorithm in step 2, then parameterize each pants patch using the method presented in section 7.3.6. In Figure 76(f), the remaining region of horse model after removing six long branches is further decomposed into four base patches (colored in green and gray).

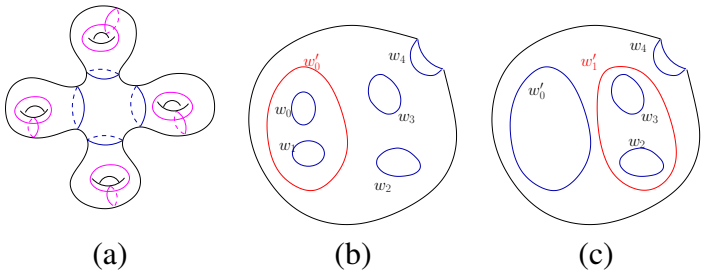


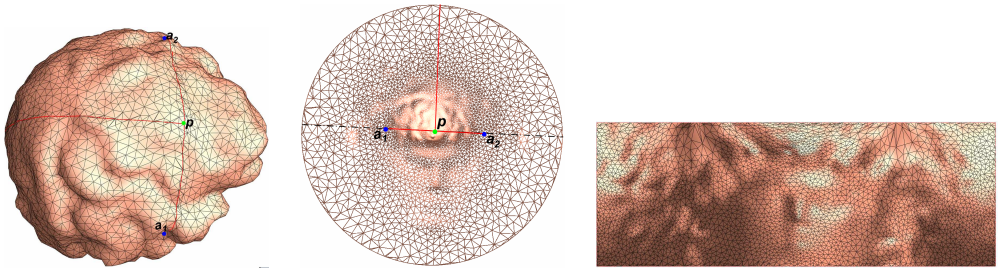

**Figure 78:** Pants Decomposition. (a) Remove handle patches. (b, c) Decompose base patch: (b) Slice $w_0'$, get a new pants patch. Boundary number decreases by 1. (c) Set $w_0'$ as a new boundary, go on to compute $w_1'$.

### 7.3.4 Branch Parameterization

Each branch $B$ of the given surface is a genus-zero patch with one boundary. Figure 79 shows the parameterization procedure for a branch from the David model shown in Figure 77. The algorithm is as follows:

*Algorithm 1:* Branch parameterization.
In: Branch $B$ with boundary length $l_B$.
Out: A rectangular domain $D$ of $B$.

1. Find a point $p$ which is the farthest point to the boundary of $B$ (Figure 79(a)).




(a) (b) (c)

**Figure 79:** Parameterizing the branch. (a) shows the pivot point $p$ (colored in green) on the given branch $B$ which is the farthest vertex from the boundary. In (b) the branch $B$ is conformally mapped to a unit disk $\Omega$ with $p$ as the disk center. Two line segments $c$ and $c'$ are selected on $\Omega$ which correspond to two smooth lines on $B$ (colored in red in (a) and (b)). (c) shows the rectangle domain $D$ obtained by running Ricci flow after slicing $B$ by using $c$ and $c'$. $a_1$ and $a_2$ marked in blue in (a) and (b) are the extraordinary points for the branch.

2. Conformally map $B$ to a unit disk $\Omega$ [64]. If $p$ is not the center of $\Omega$, use a Möbius transformation to move $p$ to the center (Figure 79(b)).
3. Find a diameter $d$ of $\Omega$, which separates the disk to two halves with minimal area difference. Choose two points $a_1$ and $a_2$ on $d$ with equal distance to $p$ in $\mathbb{R}^3$, such that the line segment $c$ connecting $a_1$ and $a_2$ passes through $p$, and its length $l_c$ in $\mathbb{R}^3$ satisfies $|2 * l_c - l_B| < \varepsilon$ . Find another line segment $c'$ perpendicular to $c$ which starts from $p$ and intersects with the boundary of $\Omega$. $c$ and $c'$ correspond to two smooth curves on $B$.
4. Slice $B$ using $c$ and $c'$, and run Ricci flow to get the rectangle domain $D$ (Figure 79(c)).

### 7.3.5 Handle Parameterization

Each handle $H$ is a genus-one surface with single boundary. The handle and tunnel loop information is computed by using Dey's algorithm [32] on the original surface and mapped to $H$. Figure 80 shows an example for the procedure of handle parameterization. The algorithm is as follows:

*Algorithm 2:* Handle parameterization.
In: Handle $H$ with computed handle and tunnel loops (Figure 80(a)).
Out: A set of four rectangle domains $D_i$ of $H$ $(0 \leq i \leq 3)$.

1. Slice $H$ along the handle and tunnel loops , and map it to a rectangle domain $D$ with one inner circle using Ricci flow (Figure 80 (b)): The inner circle corresponds to the original boundary of $H$, and the four corners of $D$ are all images of $c$ (common points of the handle and tunnel loop).
2. Find a point $p$ on the tunnel loop so that $p$, its image $p'$ and the center $o$ of the inner circle are as colinear as possible. Draw straight lines from $o$ to $p$ and $p'$ with two intersection points $a$ and $b$ with the inner circle (Figure 80(b)) which partition the domain $D$ into two parts $D'$ and $D''$ (Figure 80(d)) shows one part).
3. For $D'$, find an arc $A$ passing through $p$ and $p'$ such that $A$ has no other intersection points with $D'$ except $p$ and $p'$. $D'$ can be further divided into two parts by slicing along $A$. Find another arc $A'$ for $D''$ with the same property, and $D$ is finally decomposed into four parts by the lines $\overline{op}$ and $\overline{op'}$, and the arcs $A$ and $A'$.
4. Parameterize each of the four parts from the step 3 into a rectangle with corner points from $a, b, p, c$ using Ricci flow.

### 7.3.6 Base Patch Parameterization

Each base patch is a genus zero patch with at least three boundaries. Figure 81 (a) shows an example of the base patch from David model in Figure 77. Given a base patch $B$ with $k$ boundaries, it can be parameterized into a set of $2*k$ rectangles using the following algorithm:

*Algorithm 3:* Base Patch Parameterization.
In: Base patch $B$ with $k$ boundaries.
Out: $2*k$ rectangle domains $D_i$ of $B$ ($0 \leq i \leq 2*k-1$).

1. Find two center points $c_1$ and $c_2$, and then draw $k$ curves from each which are perpendicular to the $k$ boundaries:
   (1.1) For each vertex $v$ on base patch, compute its shortest distance to the $k$ boundaries: $d_1$, $d_2$, ..., $d_k$. Compute $c_1$ as the one with the minimum range of distances to the boundaries (Figure 81(a)).
   (1.2) Remove m-ring neighbors of $c_1$ from $B$, and map the remaining patch $B'$ to a circle $\Omega$ with $k$ holes (circles) inside, which correspond to the original

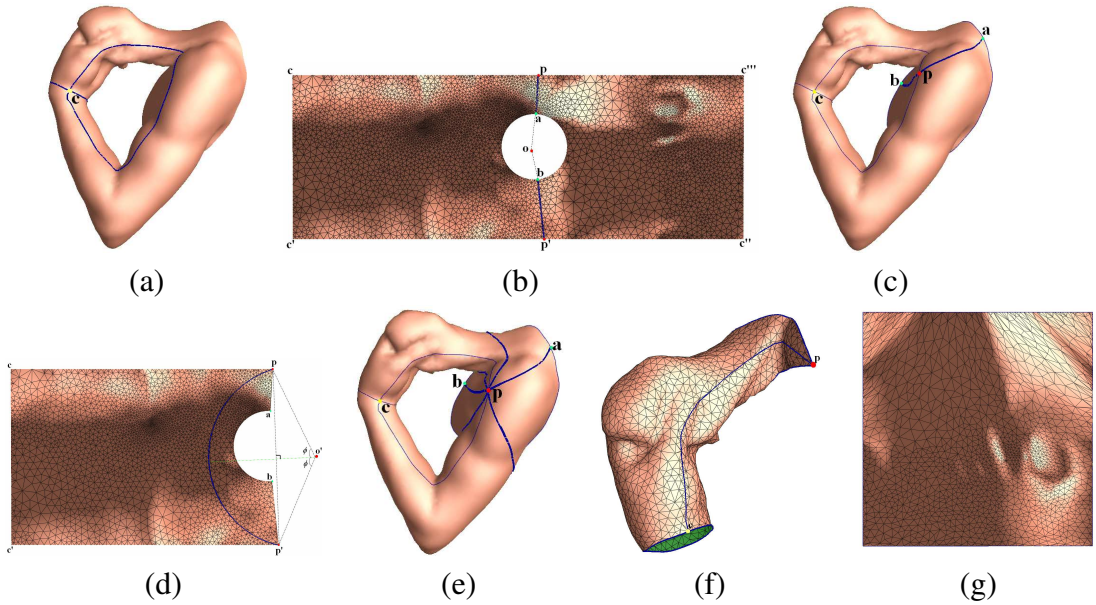



**Figure 80:** Parameterizing the handle. (a) shows one handle from David model with handle and tunnel loops highlighted in blue, and the common point $c$ of the loops shown in yellow. Point $p$ in Step 2 can be found by traversing all vertices on the tunnel loop to minimize the angle difference $|\angle pop' - \pi|$. (c) shows on the original handle patch the preimages of the cut lines (in thick blue) connecting the center of the circle and point $p$ in the rectangular domain. The arc with the property in Step 3 is not unique. One feasible way to find such an arc is to simplify the problem into finding an angle $\phi$ so that the resulting arc satisfies the required property (refer to (d)). In practice, $\phi$ can be either decided automatically by iterating all possible values to find the one with the required property and minimizing the area difference of the two parts obtained by slicing along the corresponding arc, or specified by the user in an interactive fashion to achieve a user-preferred segmentation result. (e) highlights in thick blue the lines by which the original handle is decomposed into four pieces. (f) shows one piece of the decomposition, and (g) shows its corresponding rectangular domain. $p$ is the extraordinary point for the handle.

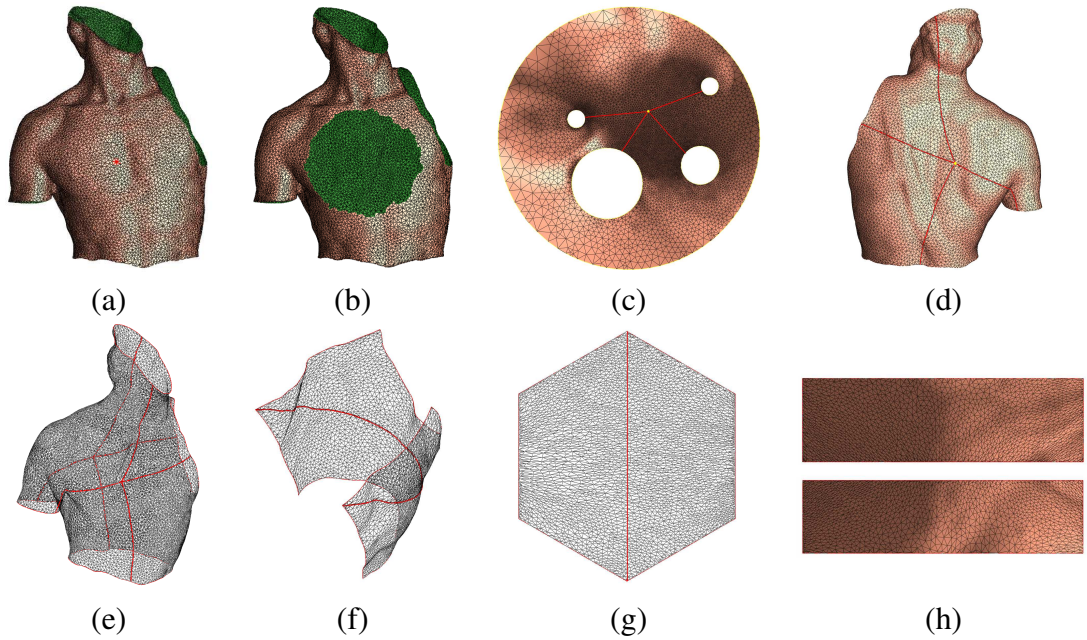



**Figure 81:** Parameterizing the base patch. (a) shows the base patch (with four boundaries) of the David model with $c_1$ marked with sharp edges, and in (b), a new boundary $b'$ is introduced by removing m-ring neighbors of $c_1$. The remaining patch is then parameterized into a disk $\Omega$ with four inner circles ((c)) which correspond to the original four boundaries. The outmost boundary of $\Omega$ corresponds to $b'$. (c) highlights in red the lines connecting $c_2$ and the centers of four inner circles, and (d) shows their preimages on the original base patch. (e) shows the eight curves from $c_1$ and $c_2$ and (f) shows one of the eight patches by slicing the original base patch using these curves. The patch in (f) is then parameterized into a regular hexagon (g), and further decomposed into two parts by slicing along the line connecting $c_1$ and $c_2$ on the hexagon, each of which is parameterized into a rectangle as shown in (h). $c_1$ and $c_2$ are the two extraordinary points for the base patch.

$k$ boundaries, and the outmost boundary of $\Omega$ corresponds to the hole introduced by removing the m-ring neighbors of $c_1$ (Figure 81(b)(c)).
(1.3) Compute $c_2$ as the one with the minimum distance range to the $k$ boundaries on $\Omega$, and draw lines from $c_2$ to each center of the $k$ circles(holes) inside $\Omega$ (Figure 81(c)).
(1.4) Remove n-ring neighbors of $c_2$ from $B$, and map the remaining patch to a circle $\Omega'$ with $k$ inner circles, draw $k$ curves on $\Omega'$ from $c_1$ to the center of each inner circle (Figure 81(e)).

2. Slice $B$ into $k$ patches using the $2*k$ curves computed from step 1, each of which contains $c_1$ and $c_2$, and four intersection points with the original $k$ boundaries. Parameterize each patch into a regular hexagon, and partition it into two parts by the line connecting $c_1$ and $c_2$ (Figure 81(f)(g)).
3. Finally $B$ is decomposed into $2*k$ patches, each of which is parameterized into a rectangle (with four corners: $c_1$, $c_2$, and two of the $2*k$ intersection points with the $k$ boundaries) using Ricci flow.

### 7.3.7 Domain Manifold Construction

Since the branches, handles and base patches are parameterized individually (refer to section 3.4 - 3.6), the parameterization may not be consistent along the shared cutting boundaries, i.e., the same cutting boundary of the 3D mesh is mapped to lines of different length by the parameterization of different patches. This inconsistency in the parametric domain causes significant troubles in constructing manifold splines, since the knot vectors of adjacent patches do not meet along the boundaries. We apply a post-processing to eliminate these inconsistency.

Note that we map all patches (branches, handles and base patches) to rectangles. Let $\phi : P \to D$ denote the parameterization, where $P$ is the 3D patch and $D$ is the rectangle on the parametric domain. The four corners of $D$ are $v_0$, $v_1$, $v_2$, and $v_3$. Then we solve a harmonic map $\psi : D \to D$ such that $\triangle\psi = 0$ with following boundary conditions:

(1) $\psi(v_i) = v_i$, $i = 0,1,2,3$;

(2) $\psi(v) = (1-\alpha)v_i + \alpha v_{i+1}$ for any boundary vertex $v \in (v_i, v_{i+1})$ and $\alpha = \frac{length(\phi^{-1}(v_i),\phi^{-1}(v))}{length(\phi^{-1}(v_i),\phi^{-1}(v_{i+1}))}$. The function $length(\mathbf{p},\mathbf{q})$ measures the arc length of

the boundary curves with end points **p** and **q**.

Solving the above harmonic map for each individual parameterization can guarantee that the shared boundary for two adjacent patches is mapped to two straight lines (side of the rectangle) that only differ by a translation, a rotation and a scaling. In other words, given two parameterized rectangles $ABCD$ and $A'B'EF$ where $AB$ and $A'B'$ are the sides corresponding to the same cutting boundary of the original mesh, we can find an affine transformation (a composite map of one translation, one rotation and one scaling) such that $AB$ and $A'B'$ coincide.

To facilitate the implementation, we scale all parameterized rectangles to make sure that two adjacent patches have the same side lengths on the parametric domain. We should also point out that T-junctions are allowed along the boundaries of the patches. Thus, the resultant T-mesh is ready to serve as the domain manifold for a manifold T-spline, where the knot interval of each edge is just its length on the parametric domain.

Given the fact that each patch is parameterized individually, thus, the above T-mesh may result in angle/area distortion along the boundaries. To reduce the distortion, we use the following technique.

For each cutting boundary, we extract the $k$-ring neighbors ($k$ is the user-specified parameter, $k$=2 in our implementation) and then map it to a rectangle. Then we solve a harmonic map for the rectangle where the vertices along the rectangle sides are fixed. This harmonic map is helpful to reduce the angle distortion of the parameterization. Figure 82(a) shows the domain manifold for David model in Figure 77 after the global relaxation.

### 7.3.8 Surface Fitting

Once the domain manifold $M$ with conformal structure $f : M \to R^2$ is given, we proceed to solve the problem of finding a good approximation of a given polygonal mesh $P$ with vertices $\{p_i\}_{i=1}^m$ by a manifold T-spline. We adopt the same strategy presented in [75] to minimize a linear combination of interpolation and fairness functionals, i.e.,

$$minE = E_{dist} + \lambda E_{fair}. \tag{44}$$

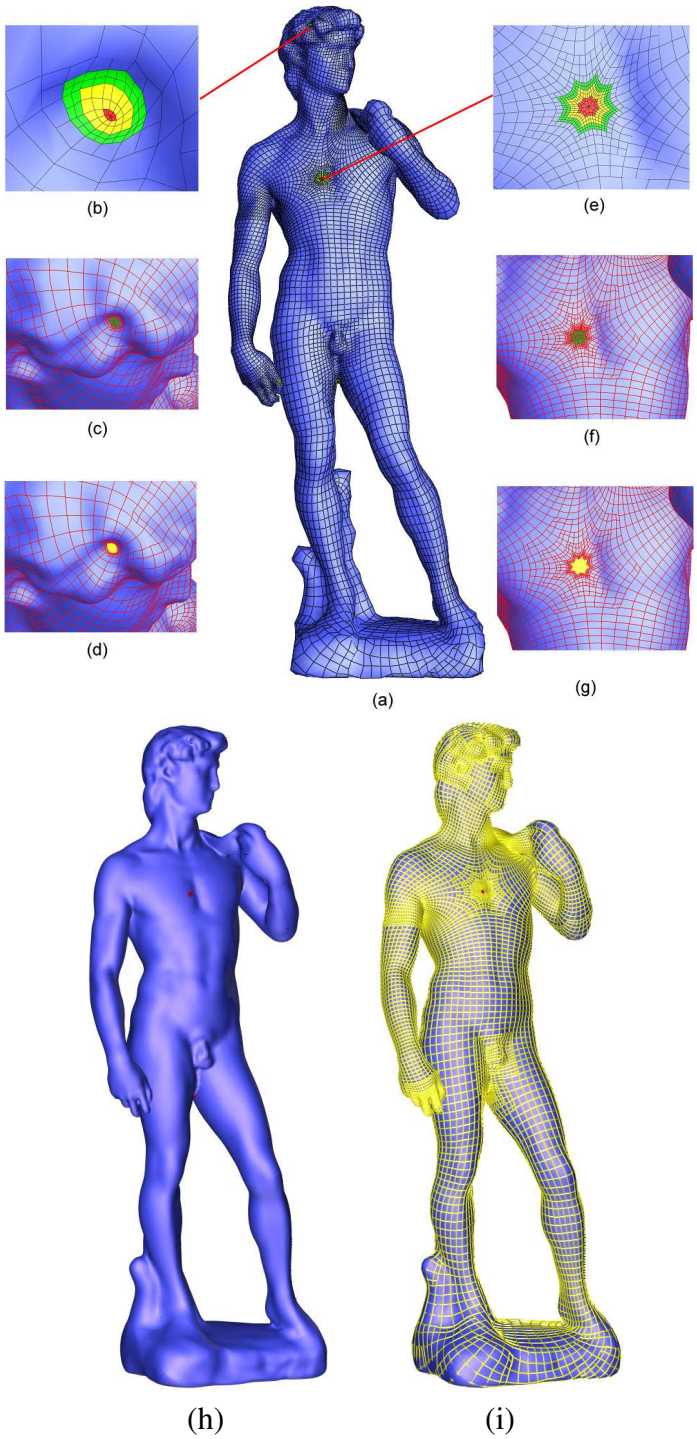



**Figure 82:** Surface fitting for David model. Extraordinary points ((b) and (e)) on the domain manifold correspond to the holes on the spline surface (shown in (c) and (f)). For each hole, we construct a Catmull-Clark subdivision surface with high order continuity along their shared boundaries. (d) and (g) show the results after hole-filling (hole areas are colored in yellow). (h) shows the manifold T-spline surface. The yellow curves on the spline surface in (i) highlights the T-junctions (singularities are colored in red).

The first part is

$$E_{dist} = \sum_{i=1}^{m} \|\mathbf{F}(\mathbf{u}_i) - \mathbf{p}_i\|^2,$$

where $\mathbf{u}_i \in M$ is the parameter for $\mathbf{p}_i$, $i = 1, ..., m$. The second part $E_{fair}$ in (44) is a smoothing term with a fairness weight $\lambda \geq 0$. In our proposed framework, the parameterizations for all decomposition components are quasi-conformal which leads to a set of good initial values for the control points, so we can obtain satisfactory results using simply a small, constant $\lambda$ as suggested in [35]. We choose $\lambda = 0.2$ in our experimentation. We refer readers to [75] for the detailed definitions of $\mathbf{F}(\mathbf{u}_i)$ and $E_{fair}$. Both parts are quadratic functions of the unknown control points, leading to a linear system. We solve Eq. (44) for unknown control points using conjugate gradient method. The value and gradient of the interpolation functional and fairness functional can be computed straightforwardly.

As discussed in section 7.3.7, our parameterization ensures the consistency along the shared boundaries of different segmented components on the parametric domain. Furthermore, our method guarantees the transitions among local charts on the domain manifold to be affine. According to manifold spline theory [60], the construction leads to an affine atlas, and the continuity along the shared boundaries is ensured automatically. The resulting spline surface is $C^2$ everywhere except at the extraordinary point.

**Handling the Extraordinary Point:** In [60], Gu *et al.* proved that manifold splines *MUST* have singularities if the domain manifold is closed and not a torus. The number of extraordinary points in our geometry-aware manifold T-spline construction pipeline equals $2 * n_{branch} + n_{handle} + 2 * n_{base}$, and they can be classified into three categories: (1) extraordinary points for handles: each handle has a single extraordinary point with valence 8 ($p$ in Figure 80(e)); (2) extraordinary points for branches: each branch has two extraordinary points with valence 2 ($a_1$ and $a_2$ in Figure 79(a) and (b)); (3) extraordinary points for base patches: each base patch with $k$ boundaries has two extraordinary points with valence $2 * k$ ($c_1$ and $c_2$ in Figure 81). Figure 82 shows one extraordinary point with valence 2 in (b) for the branch shown in Figure 79, and one extraordinary point with valence 8 in (e) for the base patch shown in Figure 81(a) with four boundaries.

Although the singularities are just points on the domain manifold, in practice,

**Table 7:** Statistics of various test examples (after 5 iterations): $g$, genus of polycube $P$; $N_{branch}$, # of branches; $N_{handle}$, # of handles; $N_{base}$, # of base patches; $N_e$, # of extraordinary points; $N_v$, # of vertices in the input polygonal mesh; $N_c$, # of control points; *rms*, root-mean-square error; $L_\infty$, maximal fitting error.

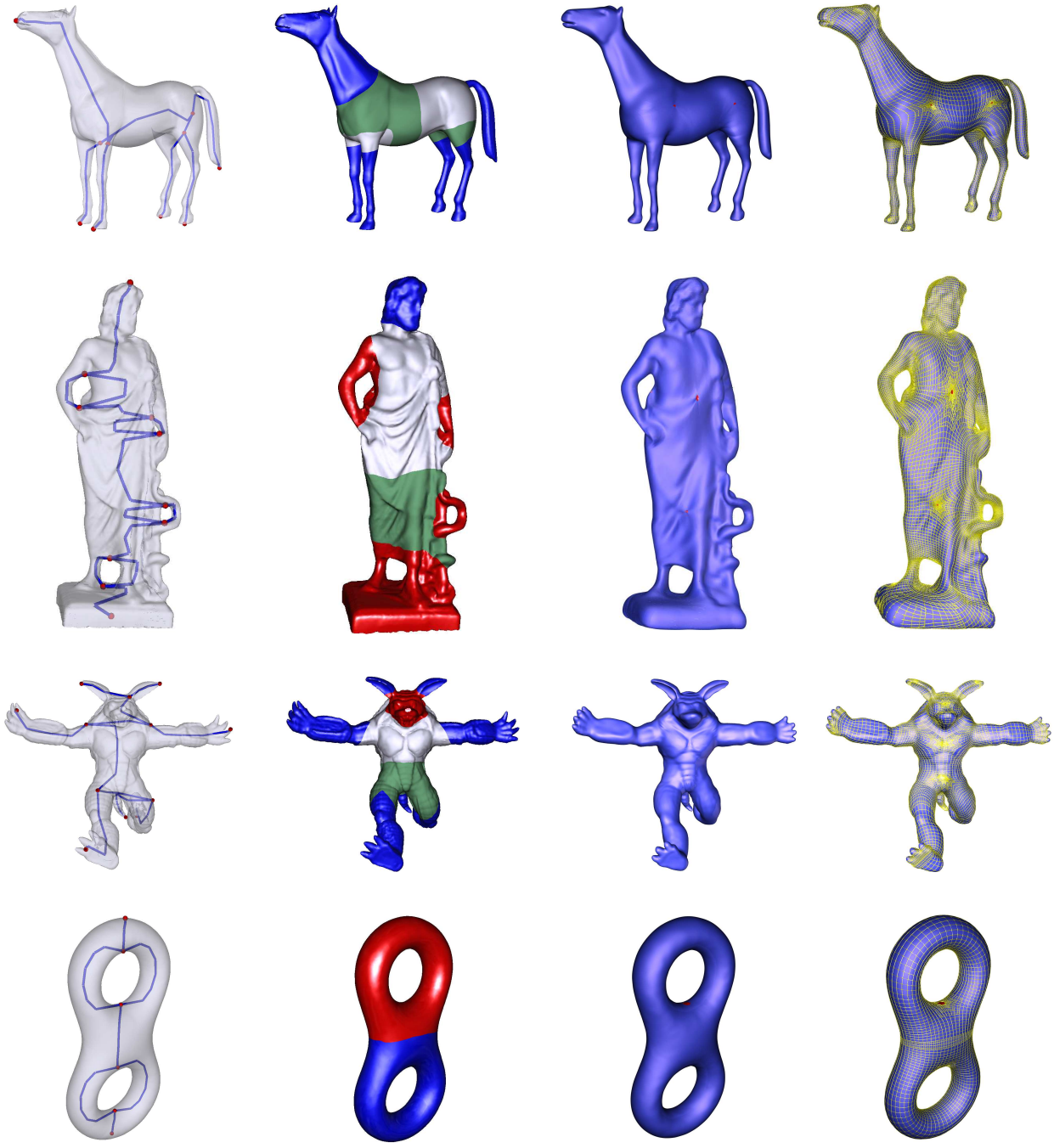
| Object | $g$ | $N_{branch}$ | $N_{handle}$ | $N_{base}$ | $N_e$ | $N_v$ | $N_c$ | *rms* | $L_\infty$ |
|---|---|---|---|---|---|---|---|---|---|
| David (Fig. 82) | 2 | 2 | 2 | 1 | 8 | 142$K$ | 15765 | 0.12% | 0.71% |
| Horse (Fig. 83) | 0 | 6 | 0 | 4 | 20 | 98$K$ | 12016 | 0.06% | 0.55% |
| Greek (Fig. 83) | 4 | 1 | 4 | 2 | 10 | 129$K$ | 16778 | 0.13% | 0.61% |
| Armadillo (Fig. 83) | 0 | 7 | 0 | 3 | 20 | 124$K$ | 15654 | 0.09% | 0.65% |
| Eight (Fig. 83) | 2 | 0 | 2 | 0 | 2 | 61$K$ | 6745 | 0.02% | 0.21% |

we have to remove these points and their 1-ring or 2-ring neighbors. As a result, the holes are unavoidable in the spline surface. Thus, we need to find a blending surface patch to fill the holes smoothly. In our implementation, we use a Catmull-Clark subdivision to fill each hole such that the surface is $C^2$ everywhere except $C^1$ at the extraordinary point. For each extraordinary point, we remove its 2-ring neighbors from the domain manifold (red quads in Figure 82(b)(e)), and use its 6-ring neighbors as the domain for the Catmull-Clark subdivision surface to fill the introduced hole. The T-spline surface is evaluated without using the yellow quads but taking into account their contributions (to the green quads). The continuity along the shared boundary between the T-spline surface and the Catmull-Clark subdivision surface (i.e., the shared boundary between yellow quads and green quads in Fig. 7(b) and (e)) is ensured naturally due to the same set of control points for that shared boundary on both the T-spline domain manifold and the domain of the Catmull-Clark subdivision surface. Figure 82 (h) shows the manifold T-spline surface built upon the domain manifold shown in Figure 82 (a). In Figure 82 (i) the yellow curves highlight the T-junctions on the spline surface. The singularities are colored in red in Figure 82 (h) and (i).

## 7.4 Implementation and Results

Our prototype system is implemented in C++ on a MS Windows XP PC with dual Intel Xeon 2.6GHz CPUs and 2GB RAM. We built a complete system for the Reeb graph computation, handle and tunnel loops detection, surface decomposition

and parameterization, and T-spline surface fitting. We tested our algorithms on various models with complicated topologies. More examples are shown in Figure 83. The results demonstrate both the theoretic rigor and feasibility in practice. The statistics of the examples are shown in Table 7.



(a)Reeb graph (b)Surface decomposition (c)T-Spline (d)T-junctions

**Figure 83:** Experimental results.

# Chapter 8

# Conclusion and Future Work

In this dissertation, we present our recent research results, ongoing research and future research directions within our general spline-based data modeling framework. We seek novel modeling techniques based on tensor-product spline schemes (the current industry standard) that would allow designers to directly define continuous spline models over any manifolds (serving as parametric domains). Our framework contributes to the research of **modeling 3D objects of arbitrary geometry and topology over regular domains**. Theoretically, it brings fundamental progress in understanding, analyzing and solving geometric modeling problems. We also demonstrate its great potential in many valuable real-world applications.

## 8.1 Contribution Summary

Our contributions in the methodologies of geometric modeling over regular domains include:

- We propose a new concept of polycube splines and develop novel modeling techniques for using the polycube splines in solid modeling and shape computing. The polycube splines are naturally built upon the polycube map which serve as its parametric domain. The use of polycubes for spline surface definition and construction is the first attempt to take advantage of the rectangular structure over the boundary of polycubes, allowing the parametric domain to actually mimic the geometry of the modeled objects with lower

area distortion while enforcing their topological consistence.

- We present a systematic way to construct polycube maps for surfaces of arbitrary topology based on global conformal parameterization. The proposed algorithm is intrinsic in that we do not need to compute the projection of the points from the 3D shape to the polycube, thus, the polycube can be flexibly constructed at any resolution and complexity.
- To control the location of extraordinary points, and further improve the mapping quality, we propose a novel framework to construct user-controllable polycube maps. The newly-proposed method allows users to directly select the corner points of the polycubes on the original 3D surfaces in an interactive manner, then constructs the polycube maps by using discrete Euclidean Ricci flow. It can construct polycube maps easily for high genus surfaces and open surfaces, which are usually difficult to be handled by the traditional methods.
- To avoid the time consuming and error-prone global parameterization for intrinsic polycube maps construction, we develop an effective method to construct polycube maps for surfaces with complicated topology and geometry in an automatic fashion. More specifically, we use a divide-and-conquer strategy, which first segments the polycube and the given 3D surface into multiple disjoint components, then constructs the piecewise polycube map for each component, and finally computes a globally smooth map for the entire polycube domain. Our algorithm can both construct a similar polycube of high geometric fidelity and compute a high-quality polycube map. In addition, our method is theoretically guaranteed to output a one-to-one map.
- We propose a geometry-aware domain decomposition algorithm for T-spline-based manifold modeling by which an arbitrarily complicated surface model can be decomposed into a group of disjoint components. Such a domain decomposition simplifies objects of arbitrary topological type into a family of genus-zero/one open surfaces, each of which can be conformally parameterized into a set of rectangles. In contrast to the conventional decomposition approaches, our method can guarantee that the cutting locus are consistent on the parametric domain. As a result, the resultant T-splines of decomposed components are automatically glued and have high-order continuity everywhere except at the extraordinary points. We show that objects with arbitrary

topology (especially objects with long branches) can be modeled elegantly by the proposed algorithm.

- We provide singularity-handling strategies: cubic triangular spline and subdivision surface based hole-filling schemes.

Practically, we demonstrate their power in many valuable applications, and show their great potential as enabling tools serving for research in broad areas of computer graphics, geometric modeling and processing, vision, visualization.

## 8.2 Future Research Directions

There are many more immediate and valuable research topics based on our current framework. Here are some research topics that directly extend from work we have done in this dissertation.

We would like to further improve the current stage of our automatic polycube map construction framework. Our proposed method has certain limitations and demands further improvement in the future. First, the constructed polycube depends on the orientation of the 3D model. Different orientations may result in very different polycubes. In our implementation, we require the user to align the model before the polycube construction. Second, the proposed method will generate a geometrically complicated polycube for non-axis-aligned long branches or handles. As a result, it may cause difficulty in some applications, e.g., spline construction, since each corner of polycube is an extraordinary point. One potential solution is to first compute the axis-independent Reeb graph representation of the given 3D surface (for example, harmonic 1-form based Reeb graph proposed in [76]), then regularize the Reeb graph so that each branch is parallel to one coordinate axis. Finally we construct the polycube from the rectified Reeb graph and associated radius information. The polycube map can then be computed in a divide-and-conquer fashion as what we do in our original work. By doing so, the number of corner points (singularities) of the resulting polycube will be much fewer, which will thereby leads to better spline surface fitting, texture mapping and synthesis, and quadrilateral remeshing results.

We would like to further strengthen our current user-controllable polycube

map framework. Within the existing framework, users are allowed to directly specify the extraordinary (corner) points of the polycubes on the input 3D surfaces. The location of singularities can be interactively placed where no important geometric features exist in order to facilitate the subsequent hole-filling process. However, domain knowledge from users is required to select a feasible set of corner points which gives rise to a polycube map with high quality. We will try to provide meaningful help to the users on corner point selection. One possible way is to exploit the topology information provided by the skeleton representation of the given surface. We also plan to seek more efficient and effective methods to improve the quality of the polycube maps.

We also expect to extend our current bivariate polycube splines to trivariate volumetric splines through the polycube volumetric parameterization of solid objects, and seek potential applications in volume modeling, simulation, finite element analysis and scientific visualization. The volume modeling framework will provide representations for the design, testing, and manufacturing of complicated mechanical objects, and will also facilitate the specification of material distributions that may be smoothly graded, discontinuous, and/or layered.

Techniques such as geometry image [57] can be used for mesh compression. Such a regular grid structure is compact and drastically simplifies the rendering pipeline and allows direct application of pixel-based image processing methods. Despite its obvious importance for efficient rendering and compression, geometry image reveals a few drawbacks due to the inevitable surface cutting: each geometry image has to be homeomorphic to a disk, therefore closed or high-genus models have to be cut along a cut graph to extract either a polygonal schema or an atlas. Finding a ”smart” cut graph (i.e. minimizing a notion of distortion) is a delicate issue and introduces a set of artificial boundary curves. Via cross surface parameterization (e.g. polycube maps), we can also re-sample (and get rid of mesh connectivity) surfaces on canonical domains without cutting. Without tearing surfaces apart, compression and reconstruction could be more effective.

In addition, the regular structure of polycubes will for sure facilitate the GPU based applications like surface/volume rendering optimization, flow simulation, FEM, etc. GPUs are probably today’s most powerful computational hardware, because the highly data-parallel nature of graphics computations (in vertex, geometry,

and fragment shaders) enable GPUs to use additional transistors more directly for computation, achieving higher arithmetic intensity with the same transistor count. The arithmetic power of GPU results from a highly specialized architecture evolved over years to extract maximum performance on the highly parallel tasks of traditional computer graphics, so general computations must be recast into graphics-specific terms in order to utilize the underlying hardware. However, not every scientific computation can take full advantage of the GPU-acceleration, especially the modeling of complex geometric shapes of arbitrary topology, due to the lack of inherent regularity structure (or parametric domain). We propose to bridge this gap by introducing surface/volume polycube mapping of complex shapes onto regular parametric domain, such that the complex geometric models can be represented as 2D/3D geometric-texture in order for the GPUs to perform the general data registration, modeling, and visualization tasks in a highly parallel fashion. The GPU-centric data formats and models will enable the efficient implementation of shape registration, surface and solid modeling, multi-scale data modeling via reverse engineering, simulation/analysis, and model visualization. Meanwhile, the efficient GPU-based algorithm will enhance existing algorithmic functionalities with improved parallel performance in order to handle large-scale, complicated models.

## 8.3 Concluding Remarks

These directions for future work, and the many other open problems that exist, are sure to encourage interesting and exciting research in geometric modeling for years to come. As technical difficulties are overcome, and existing computational algorithms are improved, the applications of geometric modeling will increase in variety and number. We are pleased to have taken the first step in uncovering the heretofore untapped potential of geometric modeling by presenting our framework to the graphics and visual computing. It is our hope that this integrated approach and demonstrated applications will foster continued interest and research in this area. We look forward to the continued exploration of geometric modeling and predict a successful future for it.

# Bibliography


[1] N. Ahuja and J.-H. Chuang. Shape representation using a generalized potential field model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):169–176, 1997.

[2] P. Alfeld, M. Neamtu, and L. L. Schumaker. Bernstein-bezier polynomials on spheres and sphere-like surfaces. *Computer Aided Geometric Design*, 13(4):333–349, 1996.

[3] P. Alliez, M. Meyer, and M. Desbrun. Interactive geomety remeshing. In *SIGGRAPH 02*, pages 347–354, 2002.

[4] N. Amenta, S. Choi, and R. K. Kolluri. The power crust. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266, New York, NY, USA, 2001. ACM.

[5] S. Angenent, S. Haker, A. Tannenbaum, and R. Kikinis. Conformal geometry and brain flattening. In *MICCAI*, pages 271–278, 1999.

[6] O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee. Skeleton extraction by mesh contraction. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–10, New York, NY, USA, 2008. ACM.

[7] G. Aujay, F. Hétroy, F. Lazarus, and C. Depraz. Harmonic skeleton for realistic character animation. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 151–160, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[8] U. Axen. Topological analysis using morse theory and auditory display. In *PhD thesis, UIUC*, 1998.

[9] U. Axen. Computing morse functions on triangulated manifolds. In *In SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[10] B. A. Barsky. Parametric bernstein/bezier curves and tensor product surfaces. Technical Report UCB/CSD-90-571, EECS Department, University of California, Berkeley, 1990.

[11] H. Blum. A transformation for extracting new descriptors of shape. In W. W. Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967.

[12] A. I. Bobenko and B. A. Springborn. Variational principles for circle patterns and koebe's theorem. *Transactions of the American Mathematical Society*, 356:659, 2004.

[13] B. Chazelle, D. P. Dobkin, N. Shouraboura, and A. Tal. Strategies for polyhedral surface decomposition: an experimental study. *Comput. Geom. Theory Appl.*, 7(5-6):327–342, 1997.

[14] B. Chazelle and L. Palios. Decomposing the boundary of a nonconvex polyhedron. In *SWAT '92: Proceedings of the Third Scandinavian Workshop on Algorithm Theory*, pages 364–375, London, UK, 1992. Springer-Verlag.

[15] C. Chen and D. Freedman. Quantifying homology classes. In *Sympos. Theoretical Aspects Comput. Sci.*, pages 169–180, 2008.

[16] B. Chow and F. Luo. Combinatorial Ricci flows on surfaces. *J. Differential Geom.*, 63(1):97–129, 2003.

[17] J.-H. Chuang, N. Ahuja, C.-C. Lin, C.-H. Tsai, and C.-H. Chen. A potential-based generalized cylinder representation. *Computers & Graphics*, 28(6):907–918, 2004.

[18] J.-H. Chuang, C.-H. Tsai, and M.-C. Ko. Skeletonization of three-dimensional object using generalized potential field. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1241–1251, 2000.

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

[20] H. B. Curry. Review. *Math. Tables Aids Comput.*, 2:167–169, 211–213, 1947.

[21] W. Dahmen and C. A. Micchelli. *Approximation Theory IV*, chapter Recent progress in multivariate splines, pages 27–121. Academic Press, New York, 1983.

[22] W. Dahmen, C. A. Micchelli, and H.-P. Seidel. Blossoming begets *B*-spline bases built better by *B*-patches. *Mathematics of Computation*, 59(199):97–115, 1992.

[23] W. A. Dahmen and C. A. Micchelli. On the linear independence of multivariate *B*-splines. I. Triangulations of simploids. *SIAM J. Numer. Anal.*, 19(5):993–1012, 1982.

[24] W. A. Dahmen and C. A. Micchelli. On the linear independence of multivariate *B*-splines. II. Complete configurations. *Math. Comp.*, 41(163):143–163, 1983.

[25] C. de Boor. On calculating with B-splines. *J. Approx. Theory*, 6(1):50–62, 1972.

[26] C. de Boor. Splines as linear combinations of *B*-splines. A survey. In *Approximation theory, II (Proc. Internat. Sympos., Univ. Texas, Austin, Tex., 1976)*, pages 1–47. Academic Press, New York, 1976.

[27] P. de Casteljau. *Formes á Pôles*. Andre Citroen, Paris, 1959.

[28] P. de Casteljau. *Outillages méthodes calcul*. Andre Citroen, Paris, 1959.

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

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

[31] T. K. Dey, K. Li, and J. Sun. On computing handle and tunnel loops. In *Proc. IEEE NASAGEM workshop*, pages 357–366, 2007.

[32] T. K. Dey, K. Li, J. Sun, and D. Cohen-Steiner. Computing geometry-aware handle and tunnel loops in 3D models. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–9, New York, NY, USA, 2008. ACM.

[33] T. K. Dey and J. Sun. Defining and computing curve-skeletons with medial geodesic function. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 143–152, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

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

[35] M. Eck and H. Hoppe. Automatic reconstruction of B-spline surfaces of arbitrary topological type. In *Proceedings of SIGGRAPH 96*, pages 325–334, 1996.

[36] Éric Colin de Verdière and F. Lazarus. Optimal system of loops on an orientable surface. In *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 627–636, Washington, DC, USA, 2002. IEEE Computer Society.

[37] J. Erickson and K. Whittlesey. Greedy optimal homotopy and homology generators. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1038–1046, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[38] Z. Fan, X. Jin, J. Feng, and H. Sun. Mesh morphing using polycube-based cross-parameterization. In *Computer Animation and Virtual Worlds*, volume 16, pages 499–508, 2005.

[39] M. S. Floater. Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.

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

[41] M. S. Floater. One-to-one piecewise linear mappings over triangulations. *Mathematics of Computation*, 72(242):685–696, 2003.

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

[43] P. Fong and H.-P. Seidel. Control points for multivariate *B*-spline surfaces over arbitrary triangulations. *Computer Graphics Forum*, 10(4):309–317, 1991.

[44] D. R. Forsey and R. H. Bartels. Hierarchical B-spline refinement. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 205–212, New York, NY, USA, 1988. ACM Press.

[45] M. Franssen, R. C. Veltkamp, and W. Wesselink. Efficient evaluation of triangular *B*-spline surfaces. *Computer Aided Geometric Design*, 17(9):863–877, 2000.

[46] M. Garland, A. Willmott, and P. S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 49–58, New York, NY, USA, 2001. ACM.

[47] A. Golovinskiy and T. Funkhouser. Consistent segmentation of 3D models. *Computers and Graphics (Shape Modeling International 09)*, 33(3):262–269, June 2009.

[48] C. Gonzalez-Ochoa and J. Peters. Localized-hierarchy surface splines (less). In *SI3D*, pages 7–15, 1999.

[49] W. J. Gordon and R. F. Riesenfeld. Bernstein-bézier methods for the computer-aided design of free-form curves and surfaces. *J. ACM*, 21(2):293–310, 1974.

[50] R. Gormaz. *B*-spline knot-line elimination and Bézier continuity conditions. In *Curves and surfaces in geometric design*, pages 209–216. A K Peters, Wellesley, MA, 1994.

[51] R. Gormaz and P.-J. Laurent. Some results on blossoming and multivariate *B*-splines. In *Multivariate approximation: from CAGD to wavelets (Santiago, 1992)*, volume 3 of *Ser. Approx. Decompos.*, pages 147–165. World Sci. Publishing, River Edge, NJ, 1993.

[52] C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3D meshes. *ACM Transactions on Graphics (TOG)*, 22(3):358–363, 2003.

[53] A. Gregory, A. State, M. C. Lin, D. Manocha, and M. A. Livingston. Interactive surface decomposition for polyhedral morphing. *The Visual Computer*, 15(9):453–470, December 1999.

[54] G. Greiner and H.-P. Seidel. Modeling with triangular *B*-splines. *IEEE Computer Graphics and Applications*, 14(2):56–60, Mar. 1994.

[55] K. P. Grinspun, E. and P. Schröder. Charms: A simple framework for adaptive simulation. In *ACM Transactions on Graphics 21*, pages 3 (July), 281–290, 2002.

[56] X. Gu. Parametrization for surfaces with arbitrary topologies. In *PhD thesis, Harvard University*, 2002.

[57] X. Gu, S. J. Gortler, and H. Hoppe. Geometry images. *ACM Trans. Graph.*, 21(3):355–361, 2002.

[58] X. Gu, Y. He, M. Jin, F. Luo, H. Qin, and S.-T. Yau. Manifold splines with single extraordinary point. In *Symposium on Solid and Physical Modeling*, pages 61–72, 2007.

[59] X. Gu, Y. He, and H. Qin. Manifold splines. In *Symposium on Solid and Physical Modeling*, pages 27–38, 2005.

[60] X. Gu, Y. He, and H. Qin. Manifold splines. *Graph. Models*, 68(3):237–254, 2006.

[61] X. Gu, S. Wang, J. Kim, Y. Zeng, Y. Wang, and H. Qin. Ricci flow for 3D shape analysis. In *IEEE International Conference on Computer Vision (ICCV 2007)*, pages 1–8, 2007.

[62] X. Gu, Y. Wang, T. F. Chan, P. M. Thompson, and S.-T. Yau. Genus zero surface conformal mapping and its application to brain surface mapping. In *Proceedings of Information Processing in Medical Imaging*, pages 172–184, 2003.

[63] X. Gu, Y. Wang, T. F. Chan, P. M. Thompson, and S.-T. Yau. Genus zero surface conformal mapping and its application to brain surface mapping. *IEEE Transactions on Medical Imaging*, 23:949–958, 2004.

[64] X. Gu and S.-T. Yau. Global conformal surface parameterization. In *ACM Symposium on Geometry Processing*, pages 127–137, 2003.

[65] X. Gu and S.-T. Yau. Computing conformal structures of surafces. In *Communication of Informtion and Systems*, pages 121–146, December 2002.

[66] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle. Conformal surface parameterization for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):181–189, 2000.

[67] R. S. Hamilton. The Ricci flow on surfaces. *Mathematics and general relativity*, 71:237–262, 1988.

[68] J. Hart. Morse theory for implicit surface modeling. In *Mathematical Visualization*, pages 257–268. Springer-Verlag, 1997.

[69] M. S. Hassouna and A. A. Farag. Robust centerline extraction framework using level sets. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 458–465, Washington, DC, USA, 2005. IEEE Computer Society.

[70] Y. He, X. Gu, and H. Qin. Fairing triangular *B*-splines of arbitrary topology. In *Proceedings of Pacific Graphics (PG '05) short paper*, pages 153–156, 2005.

[71] Y. He, X. Gu, and H. Qin. Rational spherical splines for genus zero shape modeling. In *Proceedings of Shape Modeling International (SMI '05)*, pages 82–91, 2005.

[72] Y. He, X. Gu, and H. Qin. Automatic shape control of triangular B-splines of arbitrary topology. *Journal of Computer Science and Technology*, 21(2):232–237, 2006.

[73] Y. He, M. Jin, X. Gu, and H. Qin. A $C^1$ globally interpolatory spline of arbitrary topology. In *Proceedings of the 3rd IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision*, pages 295–306, 2005.

[74] Y. He and H. Qin. Surface reconstruction with triangular *B*-splines. In *Proceedings of Geometric Modeling and Processing (GMP '04)*, pages 279–290, 2004.

[75] Y. He, K. Wang, H. Wang, X. Gu, and H. Qin. Manifold T-spline. In *In Proceedings of Geometric Modeling and Processing*, pages 409–422, 2006.

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

[77] Y. He, X. Yin, F. Luo, and X. Gu. Harmonic volumetric parameterization using green's functions on star shapes. In *Eurographics Symposium on Geometry Processing (SGP '08)*, 2008.

[78] K. Höllig. Multivariate splines. *SIAM J. Numer. Anal.*, 19(5):1013–1031, 1982.

[79] K. Hormann. Theory and applications of parameterizing triangulations. In *PhD thesis, Department of Computer Science, University of Erlangen, November*, 2001.

[80] M. Jin, J. Kim, and X. D. Gu. Discrete surface Ricci flow: Theory and applications. In *IMA Conference on the Mathematics of Surfaces*, pages 209–232, 2007.

[81] M. Jin, J. Kim, S. Lee, and X. Gu. Conformal surface parameterization using Ricci flow. Technical Report, 2006.

[82] M. Jin, F. Luo, and X. Gu. Computing surface hyperbolic structure and real projective structure. In *SPM '06: Proceedings of the 2006 ACM symposium on Solid and physical modeling*, pages 105–116, New York, NY, USA, 2006. ACM.

[83] M. Jin, Y. Wang, S.-T. Yau, and X. Gu. Optimal global conformal surface parameterization. In *In IEEE Visualization*, pages 267–274, 2004.

[84] J. Jost and R. R. Simha. Compact Riemann surfaces: An introduction to contemporary mathematics. 1997.

[85] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 279–286, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[86] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *SIGGRAPH '03: ACM SIGGRAPH 2003*, pages 954–961, New York, NY, USA, 2003. ACM.

[87] L. Kharevych, B. Springborn, and P. Schröder. Discrete conformal mappings via circle patterns. *ACM Trans. Graph.*, 25(2):412–438, 2006.

[88] A. Khodakovsky, N. Litke, and P. Schroder. Globally smooth parameterizations with low distortion. *ACM Trans. Graph.*, 22(3):350 – 357, 2003.

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

[90] R. KRAFT. Adaptive and linearly independent multilevel b-splines. In *Surface Fitting and Multiresolution Methods*, pages 209–218. A. L. Mhaut, C. Rabut, and L. L. Schumaker, Eds., vol. 2. Vanderbilt University Press, 1998.

[91] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 313–324. ACM Press, 1996.

[92] M. J. Lai and P. Wenston. Bivariate splines for fluid flows. In *Computers and Fluids 33*, pages 1047–1073, 2004.

[93] Y. Lee, H. Kim, and S. Lee. Mesh parameterization with a virtual boundary. *Computers and Graphics*, 26(5):677–686, 2002.

[94] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. In *SIGGRAPH 02*, pages 362–371, 2002.

[95] H. Li, K.-Y. Lo, M.-K. Leung, and C.-W. Fu. Dual Poisson-disk tiling: An efficient method for distributing features on arbitrary surfaces. In *IEEE Transactions on Visualization and Computer Graphics*, volume 14, pages 982–998, 2008.

[96] X. Li, Y. Bao, X. Guo, M. Jin, X. Gu, and H. Qin. Global optimal surface mapping for shapes of arbitrary topology. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 14(4):805–819, 2008.

[97] X. Li, X. Gu, and H. Qin. Surface matching using consistent pants decomposition. In *SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 125–136, New York, NY, USA, 2008. ACM.

[98] X. Li, X. Guo, H. Wang, Y. He, X. Gu, and H. Qin. Harmonic volumetric mapping for solid modeling applications. In *SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 109–120, New York, NY, USA, 2007. ACM.

[99] X. Li, T. W. Woon, T. S. Tan, and Z. Huang. Decomposing polygon meshes for interactive applications. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 35–42, New York, NY, USA, 2001. ACM.

[100] J.-M. Lien, J. Keyser, and N. M. Amato. Simultaneous shape decomposition and skeletonization. In *SPM '06: Proceedings of the 2006 ACM symposium on Solid and physical modeling*, pages 219–228, New York, NY, USA, 2006. ACM.

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

[102] C.-M. Ma, S.-Y. Wan, and J.-D. Lee. Three-dimensional topology preserving reduction on the 4-subfields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(12):1594–1605, 2002.

[103] J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. In *SIGGRAPH 93*, pages 27–34.

[104] G. Malandain and S. Fernández-Vidal. Euclidean skeletons. *Image and Vision Computing*, 16(5):317–327, 1998.

[105] A. P. Mangan and R. T. Whitaker. Partitioning 3D surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.

[106] M. D. McCool. Analytic antialiasing with prism splines. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 429–436. ACM Press, 1995.

[107] J. W. Milnor. On the existence of a connection with curvature zero. *Comm. Math. Helv.*, 32:215–223, 1958.

[108] J. W. Milnor. Morse theory. In *Princeton University Press*, 1963.

[109] M. Mortara and G. Patanè. Shape-covering for skeleton extraction. *International Journal of Shape Modeling*, 8(2):139–158, 2002.

[110] M. Neamtu. Bivariate simplex *B*-splines: A new paradigm. In *SCCG '01: Proceedings of the 17th Spring conference on Computer graphics*, pages 71–78, 2001.

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

[112] R. Ogniewicz and M. Ilg. Voronoi skeletons: Theory and applications. In *Proc. Conf. on Computer Vision and Pattern Recognition*, pages 63–69, 1992.

[113] K. Palágyi and A. Kuba. A parallel 3D 12-subiteration thinning algorithm. *Graph. Models Image Process.*, 61(4):199–221, 1999.

[114] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust on-line computation of reeb graphs: simplicity and speed. In *SIGGRAPH '07: ACM SIGGRAPH 2007*, page 58, New York, NY, USA, 2007. ACM.

[115] R. Pfeifle and H.-P. Seidel. Fitting triangular *B*-splines to functional scattered data. In *Graphics Interface '95*, pages 26–33, 1995.

[116] R. Pfeifle and H.-P. Seidel. Spherical triangular *B*-splines with application to data fitting. *Computer Graphics Forum*, 14(3):89–96, 1995.

[117] L. A. Piegl and W. Tiller. *The NURBS Book*. Springer, 2 edition, 1996.

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

[119] M. J. D. Powell and M. A. Sabin. Piecewise quadratic approximations on triangles. *ACM Trans. Math. Softw.*, 3(4):316–325, 1977.

[120] L. Ramshaw. Blossoming: A connected-the-dots approach to splines. Technical report, Digital Systems Research Center, Palo Alto, 1987.

[121] L. Ramshaw. Béziers and B-splines as multiaffine maps. In *Theoretical Foundations of Computer Graphics and CAD*, pages 757–776. Springer, 1988.

[122] L. Ramshaw. Blossom are polar forms. *Computer-Aided Geom. Design*, 6(4):323–358, 1989.

[123] G. Reeb. Sur les points singuliers d'une forme de Pfaff complétement intégrable ou d'une fonction numérique. *Comptes Rendus de L'Académie ses Sciences*, pages 847–849, 1946.

[124] M. Reuter, S. Biasotti, D. Giorgi, G. Patanè, and M. Spagnuolo. Discrete laplace-beltrami operators for shape analysis and segmentation. *Computers & Graphics*, 33:381–390, 2009.

[125] I. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions, part a: On the problem of smoothing or graduation, a first class of analytic approximation formulas. *Quart. Appl. Math.*, 4:45–99, 1946.

[126] I. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions, part b: On the problem of osculatory interpolation, a second class of analytic approximation formulae. *Quart. Appl. Math.*, 4:112–141, 1946.

[127] J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe. Inter-surface mapping. *ACM Trans. Graph.*, 23(3):870 – 877, 2004.

[128] T. W. Sederberg, D. L. Cardon, G. T. Finnigan, N. S. North, J. Zheng, and T. Lyche. T-spline simplification and local refinement. *ACM Trans. Graph.*, 23(3):276–283, 2004.

[129] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-nurccs. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 477–484, New York, NY, USA, 2003. ACM.

[130] H.-P. Seidel. Symmetric recursive algorithms for surfaces: B-patches and the de boor algorithm for polynomials over triangles. *Constr. Approx.*, 7:257–279, 1991.

[131] H.-P. Seidel. An introduction to polar forms. In *IEEE Transaction on Computer Graphics*, pages 38–46, 1993.

[132] H.-P. Seidel. Polar forms and triangular *B*-spline surfaces. In D.-Z. Du and F. Hwang, editors, *Euclidean Geometry and Computers, 2nd Edition*, pages 235–286. World Scientific Publishing Co., 1994.

[133] L. Shapira, A. Shamir, and D. Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis. Comput.*, 24(4):249–259, 2008.

[134] D. W. Shattuck and R. M. Leahy. Automated graph-based analysis and correction of cortical volume topology. In *IEEE Trans. Med. Imaging 20*, pages 1167–1177, 2001.

[135] A. Sheffer. Spanning tree seams for reducing parameterization distortion of triangulated surfaces. In *Proceedings of Shape Modeling International*, 2002.

[136] A. Sheffer and E. de Sturler. Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers*, 17(3):326–337, 2001.

[137] A. Sheffer, C. Gotsman, and N. Dyn. Robust spherical parameterization of triangular meshes. *Computing*, 72(1-2):185–193, 2004.

[138] S. Shlafman, A. Tal, and S. Katz. Metamorphosis of polyhedral surfaces using decomposition. *Comput. Graph. Forum*, 21(3):219–228, 2002.

[139] K. Stephenson. *Introduction To Circle Packing*. Cambridge University Press, 2005.

[140] G. Strang and G. Fix. An analysis of the finite element method. Prentice Hall, 1973.

[141] A. Tagliasacchi, H. Zhang, and D. Cohen-Or. Curve skeleton extraction from incomplete point cloud. *ACM Transactions on Graphics, (Proceedings SIGGRAPH 2009)*, 28(3), 2009.

[142] M. Tarini, K. Hormann, P. Cignoni, and C. Montani. Polycube-maps. *ACM Trans. Graph.*, 23(3):853–860, 2004.

[143] W. Thurston. *Geometry and Topology of 3-manifolds*. Princeton lecture notes, 1976.

[144] J. Tierny, J. P. Vandeborre, and M. Daoudi. 3D mesh skeleton extraction using topological and geometrical analyses. In *14th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2006)*, pages 85–94, 2006.

[145] J. Tierny, J.-P. Vandeborre, and M. Daoudi. Topology driven 3D mesh hierarchical segmentation. In *IEEE International Conference on Shape Modeling and Applications (SMI 2007)*, pages 215–220, Lyon, France, 2007.

[146] K. J. Versprille. *Computer-Aided Design Applications of the Rational B-Spline Approximation Form*. PhD thesis, Syracuse University, 1975.

[147] H. Wang, Y. He, X. Li, X. Gu, and H. Qin. Polycube splines. In *Symposium on Solid and Physical Modeling*, pages 241–251, 2007.

[148] H. Wang, M. Jin, Y. He, X. Gu, and H. Qin. User-controllable polycube map for manifold spline construction. In *SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 397–404, New York, NY, USA, 2008. ACM.

[149] L. Wang, X. Gu, K. Mueller, and S.-T. Yau. Uniform texture synthesis and texture mapping using global parameterization. In *The Visual Computer (Special issue of Pacific Graphics '05)*, pages 801–810, 2005.

[150] Y.-S. Wang and T.-Y. Lee. Curve-skeleton extraction using iterative least squares optimization. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):926–936, 2008.

[151] F. Weller and H. Hagen. Tensor product spline spaces with knot segments. in mathematical methods for curves and surfaces. In *Mathematical Methods for Curves and Surfaces*, pages 563–572. M. Daehlen, T. Lyche, and L. L. Schumaker, Eds. Vanderbilt University Press, Nashville, 1995.

[152] F.-C. Wu, W.-C. Ma, R.-H. Liang, B.-Y. Chen, and M. Ouhyoung. Domain connected graph: the skeleton of a closed 3D shape for animation. *Vis. Comput.*, 22(2):117–135, 2006.

[153] Q.-Y. Zhou, T. Ju, and S.-M. Hu. Topology repair of solid models using skeletons. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):675–685, 2007.

[154] Y. Zhou and A. W. Toga. Efficient skeletonization of volumetric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):196–209, 1999.

[155] M. Zöckler, D. Stalling, and H.-C. Hege. Fast and intuitive generation of geometric shape transitions. *The Visual Computer*, 16(5):241–253, June 2000.

[156] E. Zuckerberger. Polyhedral surface decomposition with applications. *Computers and Graphics*, 26(5):733–743, October 2002.