Efficient Bijective Spherical Parameterizations

高效无翻转球面参数化

Huanyu Chen

Bachelor's Thesis
Mathematics and Applied Mathematics
University of Science and Technology of China

Supervisor: Prof. Ligang Liu

May 24, 2021

English translation, November 2026. The original thesis is in Chinese; in case of any discrepancy, the Chinese original prevails. Original and translation: `https://huanyuusc.github.io/spherical/`

Acknowledgments

After continual inquiry and exploration, this project, which has taken more than a year, has finally produced good results. The topic was assigned to me by Prof. Ligang Liu when he taught the course Computer Graphics in the spring of 2020. Under Prof. Liu's guidance I read nearly a hundred papers, implemented dozens of algorithms, and came up with a great many new ideas, then kept on trying and failing until I found the right direction and was able to complete this project. Here I would first like to express my most sincere gratitude to Prof. Liu, for leading me through the door of computer graphics and onto the path of research, and for his constant guidance and inspiration. I would also like to thank Prof. Xiao-Ming Fu and my senior labmate Chunyang Ye for patiently answering my many questions throughout this process.

Four years of college have passed in the blink of an eye, and I have gained a great deal during them. I thank the university for the many courses it offers, which gave me a solid foundation in mathematics and physics; for the academic lectures it organizes, which broadened my horizons; and for the activity facilities it provides, which enriched my life outside class. I would like to thank my class advisor, Mr. Tie Zhang, for constantly encouraging me and helping me stay true to my original aspiration and hold on to my dream of doing research; the academic secretaries, Ms. Ying Qi and Mr. Wei Zhang, for answering my questions about academic planning; and the dormitory staff for their help in daily life. I thank my classmates and roommates for working hard alongside me and making progress together.

On my way to graduate school, I would like to thank Prof. Ligang Liu, Prof. Juyong Zhang, and Prof. Yang Cao for their recommendations, which allowed me to receive fully funded direct-PhD offers from several renowned universities abroad. I would also like to thank Prof. Renjie Chen and my senior schoolmate Gao Chen for helping me plan my coursework and studies, and Prof. Yongqi Liang and Prof. Xiao-Ming Fu for their guidance and advice in choosing a school.

Finally, I want to give my heartfelt thanks to my parents. I thank them for their selfless devotion and for raising me; for their attentive care, which allowed me to receive a good education and become a well-read and versatile student; for cultivating my ability to think and live independently; for giving me the freedom to choose and room to grow; and for shaping my values and outlook, which made me a person with a strong sense of justice who is glad to help others. In the future, I will devote myself to scientific research and strive to become a scholar who is both principled and professionally accomplished, and who unites theory with practice, to repay what my parents, my university, and my country have given me.

Abstract

In computer graphics, surfaces are often represented as triangle meshes. Mesh parameterization has long been one of the fundamental problems in digital geometry processing, and it plays an important role in a wide range of graphics applications, including texture mapping, remeshing, surface mapping, geometry images, and morphing. In real applications, meshes usually come from 3D scans of object surfaces, and thus are often closed. To parameterize such a mesh, we can first cut it open and then map it to a planar region, or we can directly map the whole mesh to some closed surface. For a closed surface mesh of genus 0, the most natural parameter domain is the unit sphere. Planar parameterization has been studied extensively, so a wide variety of solution methods are available. In contrast, spherical parameterization has received much less research attention because of the difficulties brought by the spherical constraint. In particular, planar parameterization has a universal geometric optimization pipeline that guarantees a flip-free parameterization map, but this pipeline does not work well when applied directly to spherical parameterization. The goal of this thesis is to modify this pipeline so that it also works smoothly for spherical parameterization.

This thesis first introduces the problem of low-distortion, flip-free parameterization, then elaborates on the geometric optimization pipeline and the various solvers for planar parameterization, then surveys existing spherical parameterization methods, and finally proposes a geometric optimization pipeline for spherical parameterization. Experimental results show that the geometric optimization pipeline proposed in this thesis solves the spherical parameterization problem well and produces spherical parameterization maps with low distortion.

Keywords: Geometry Processing, Mesh Parameterization, Geometric Optimization

Contents

Acknowledgments i

Abstract ii

1 Introduction 1
- 1.1 Smooth Surfaces . . . 1
  - 1.1.1 Basic Definitions [45] . . . 1
  - 1.1.2 Local Theory of Surfaces [1] . . . 2
  - 1.1.3 Global Theory of Surfaces [1] . . . 5
- 1.2 Triangle Mesh Surfaces . . . 6
  - 1.2.1 Surface Representations in Practice . . . 6
  - 1.2.2 Discrete Differential Geometry . . . 8
- 1.3 The Mesh Parameterization Problem . . . 9
  - 1.3.1 Overview of the Problem . . . 9
  - 1.3.2 Distortion Energy . . . 11
  - 1.3.3 Formulation of Low-Distortion Parameterization . . . 14
- 1.4 An Introduction to Geometric Optimization Methods . . . 14
  - 1.4.1 General Nonlinear Optimization [26, 27] . . . 14
  - 1.4.2 Geometric Optimization . . . 16

2 Geometric Optimization Methods for Planar Parameterization 18
- 2.1 Tutte's Method and Its Variants . . . 18
- 2.2 Iterative Solvers . . . 20
  - 2.2.1 Preconditioners . . . 20
  - 2.2.2 Second-order solvers . . . 24
  - 2.2.3 Line search [43] . . . 27
  - 2.2.4 Termination criterion . . . 27
- 2.3 A brief introduction to acceleration methods . . . 28
  - 2.3.1 Changing the reference triangles [23] . . . 28
  - 2.3.2 Changing the search direction [51] . . . 29
  - 2.3.3 Nesterov acceleration [25][21] . . . 29
  - 2.3.4 Anderson acceleration [2][28] . . . 29

3 Low-Distortion Flip-Free Spherical Parameterization 31
- 3.1 Introduction to Spherical Parameterization . . . 31
  - 3.1.1 Spherical Parameterization Based on Stereographic Projection . . . 31
  - 3.1.2 Spherical Parameterization Based on Gnomonic Projection . . . 32
  - 3.1.3 Spherical Parameterization Based on Conformal Geometry . . . 33
  - 3.1.4 The Colin de Verdière Equations . . . 33
  - 3.1.5 Spherical Parameterization Based on Multiresolution . . . 34
  - 3.1.6 Geometric Optimization for Spherical Parameterization . . . 35

3.2 Spherical Triangle Meshes . . . 35
3.2.1 The Flip-Free Condition for Spherical Triangles . . . 35
3.2.2 Rigorous Spherical Parameterization Maps . . . 36
3.2.3 Distortion Energies of Spherical Parameterization Maps . . . 37
3.2.4 Formulation of the Geometric Optimization Problem for Spherical Parameterization . . . 39
3.3 Geometric Optimization of Spherical Parameterizations . . . 40
3.3.1 Spherical Initialization . . . 40
3.3.2 Optimization Solvers . . . 40
3.3.3 Handling the Spherical Constraint . . . 43
3.3.4 Line Search . . . 44
3.4 Experimental Results . . . 45
3.4.1 Implementation Details . . . 45
3.4.2 Results . . . 46
3.4.3 Conclusion . . . 46

4 Summary and Reflections **51**

Chapter 1

Introduction

1.1 Smooth Surfaces

1.1.1 Basic Definitions [45]

An n-dimensional **topological manifold** is a second-countable Hausdorff space that is locally homeomorphic to n-dimensional Euclidean space. This means that for every point p in the space there exist an open neighborhood U_p and a homeomorphism

$$\phi_p : U_p \simeq V_p \subseteq \mathbb{R}^n,$$

whose inverse ϕ_p^{-1} can be regarded as a parametric representation of the manifold near that point. The pair (U_p, ϕ_p) is therefore called a **chart**, U_p is called a **coordinate neighborhood** of the point p, and ϕ_p is called the **coordinate map** on U_p. If the transition maps between two charts (U, ϕ), (V, ψ) are smooth, i.e.,

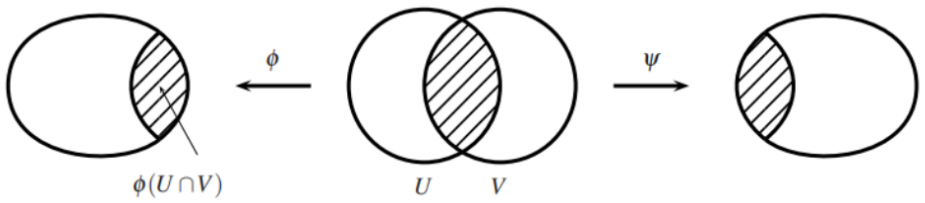


Figure 1.1: Transition maps between two charts [45]

$$\phi \circ \psi^{-1} : \psi(U \cap V) \to \phi(U \cap V), \qquad \psi \circ \phi^{-1} : \phi(U \cap V) \to \psi(U \cap V)$$

are both smooth maps, then the two charts are said to be **compatible**. An **atlas** on a manifold M is a family of pairwise compatible charts $\mathfrak{U} = \{(U_\alpha, \phi_\alpha)\}$ that covers M, i.e., $M = \bigcup_\alpha U_\alpha$. If such an atlas exists on the manifold M, then M is called a **smooth manifold**.

In differential geometry, a **surface** generally refers to a two-dimensional smooth manifold, denoted S. Hence every point on a surface has a coordinate neighborhood that can be parameterized by a planar region. Here we assume that the surface can be embedded in three-dimensional Euclidean space, i.e., there exists an embedding $f : S \hookrightarrow \mathbb{R}^3$. If we identify each point of the surface with its image, we may regard the surface directly as a subset of three-dimensional Euclidean space. In this way, every point p on the surface has a smooth parametric representation $\boldsymbol{f_p} : \Omega_p \to \mathbb{R}^3$ near it, where Ω_p is a planar region whose image under $\boldsymbol{f_p}$ is a coordinate neighborhood of p.

1.1.2 Local Theory of Surfaces [1]

1. Fundamental Forms and Curvature

To make it easier to study the local properties of a surface, we may assume without loss of generality that the surface is $\boldsymbol{f} : \Omega \to \mathbb{R}^3$, and identify a point p on the surface with its parameter coordinates (u, v). The plane spanned by

$$\boldsymbol{f}_u(p) = \frac{\partial \boldsymbol{f}}{\partial u}(p), \qquad \boldsymbol{f}_v(p) = \frac{\partial \boldsymbol{f}}{\partial v}(p)$$

at the point p is called the **tangent plane** at that point, denoted T_pS.

$$\boldsymbol{n}(p) = \frac{\boldsymbol{f}_u(p) \times \boldsymbol{f}_v(p)}{\|\boldsymbol{f}_u(p) \times \boldsymbol{f}_v(p)\|}$$

is called the **unit normal** at that point; it lies on the two-dimensional sphere S^2. The map

$$G : S \to S^2, \qquad p \mapsto \boldsymbol{n}(p)$$

is called the **Gauss map** of the surface.

The **first fundamental form** of the surface is defined as

$$I = d\boldsymbol{f} \cdot d\boldsymbol{f} = \begin{pmatrix} du & dv \end{pmatrix} \begin{pmatrix} E & F \\ F & G \end{pmatrix} \begin{pmatrix} du \\ dv \end{pmatrix},$$

which is the metric induced by three-dimensional Euclidean space on the tangent plane at a point of the surface, and characterizes the local metric properties of the surface. By definition,

$$E = \boldsymbol{f}_u \cdot \boldsymbol{f}_u, \qquad F = \boldsymbol{f}_u \cdot \boldsymbol{f}_v, \qquad G = \boldsymbol{f}_v \cdot \boldsymbol{f}_v.$$

The **second fundamental form** of the surface is defined as

$$II = -d\boldsymbol{f} \cdot d\boldsymbol{n} = \begin{pmatrix} du & dv \end{pmatrix} \begin{pmatrix} L & M \\ M & N \end{pmatrix} \begin{pmatrix} du \\ dv \end{pmatrix},$$

which characterizes the shape of the surface near a point. By definition,

$$L = \boldsymbol{f}_{uu} \cdot \boldsymbol{n}, \qquad M = \boldsymbol{f}_{uv} \cdot \boldsymbol{n}, \qquad N = \boldsymbol{f}_{vv} \cdot \boldsymbol{n}.$$

Both the first and the second fundamental forms are independent of the choice of parameters, and are invariant under orthogonal transformations of $\mathbb{R}^3$.

Given a unit tangent vector $\boldsymbol{t}$ at a point, the component along the normal vector of the curvature vector of the surface in the direction $\boldsymbol{t}$ at that point is called the **normal curvature**; it can be computed by

$$k_n(\boldsymbol{t}) = \frac{II(\boldsymbol{t}, \boldsymbol{t})}{I(\boldsymbol{t}, \boldsymbol{t})}.$$

The maximum and minimum of the normal curvatures over all directions are called the **principal curvatures**, denoted k_1, k_2; they are the eigenvalues of the matrix $\begin{pmatrix} L & M \\ M & N \end{pmatrix} \begin{pmatrix} E & F \\ F & G \end{pmatrix}^{-1}$. In particular, $H = (k_1 + k_2)/2$ is called the **mean curvature** at the point, and $G = k_1 k_2$ is called the **Gaussian curvature** at the point. The Gaussian curvature is the area-scaling factor of the Gauss map.

2. Differential Forms on Surfaces

Let $\{\boldsymbol{e}_1, \boldsymbol{e}_2\}$ be an orthonormal basis of the tangent plane at a point p on the surface S; then

$$\boldsymbol{e}_3 = \boldsymbol{e}_1 \times \boldsymbol{e}_2 = \boldsymbol{n}$$

is the unit normal, and

$$\omega_1 = d\boldsymbol{f} \cdot \boldsymbol{e}_1, \qquad \omega_2 = d\boldsymbol{f} \cdot \boldsymbol{e}_2$$

are the corresponding 1-forms. Further, let

$$\omega_{ij} = d\boldsymbol{e}_i \cdot \boldsymbol{e}_j, \qquad i, j = 1, 2, 3;$$

then

$$\omega_{ij} + \omega_{ji} = d(\boldsymbol{e}_i \cdot \boldsymbol{e}_j) = d\delta_{ij} = 0, \qquad i, j = 1, 2, 3.$$

The structure equations of the surface state that

$$\begin{cases} d\omega_1 = \omega_{12} \wedge \omega_2 \\ d\omega_2 = \omega_1 \wedge \omega_{12} \end{cases}, \qquad \begin{cases} d\omega_{12} = \omega_{13} \wedge \omega_{32} \\ d\omega_{13} = \omega_{12} \wedge \omega_{23} \\ d\omega_{23} = \omega_{21} \wedge \omega_{13} \end{cases}.$$

In addition, the geometric quantities of the surface can all be expressed in terms of these 1-forms:

$$\begin{cases} I = d\boldsymbol{f} \cdot d\boldsymbol{f} = \omega_1\omega_1 + \omega_2\omega_2 \\ II = -d\boldsymbol{f} \cdot d\boldsymbol{n} = \omega_1\omega_{13} + \omega_2\omega_{23} \\ III = d\boldsymbol{n} \cdot d\boldsymbol{n} = \omega_{13}\omega_{13} + \omega_{23}\omega_{23} \end{cases}, \qquad \begin{cases} dA = \omega_1 \wedge \omega_2 \\ d\sigma = \omega_{13} \wedge \omega_{23} \end{cases},$$

where $d\sigma = K\, du \wedge dv = K\, dA$ is the area element of the Gaussian curvature. In particular,

$$d\omega_{12} = \omega_{13} \wedge \omega_{32} = -d\sigma = -K\, dA = -K\, \omega_1 \wedge \omega_2,$$

where ω_{12} is called the **connection 1-form**. The above equation can also be written as

$$K = -\frac{d\omega_{12}}{\omega_1 \wedge \omega_2},$$

which is called the **Gauss equation** of the surface. From this equation one obtains the famous **Theorema Egregium** (Gauss' Theorema Egregium): the Gaussian curvature of a surface depends only on its first fundamental form.

3. Maps Between Surfaces

Let $\boldsymbol{f}_1 : \Omega_1 \to S_1 \subseteq \mathbb{R}^3$, $\boldsymbol{f}_2 : \Omega_2 \to S_2 \subseteq \mathbb{R}^3$ be two smooth surfaces. A map between surfaces $\phi : S_1 \to S_2$ induces a map between tangent spaces

$$d\phi : T_pS_1 \to T_{\phi(p)}S_2, \qquad \frac{\partial \boldsymbol{f}_1}{\partial u} \mapsto \frac{\partial(\phi \circ \boldsymbol{f}_1)}{\partial u}, \qquad \frac{\partial \boldsymbol{f}_1}{\partial v} \mapsto \frac{\partial(\phi \circ \boldsymbol{f}_1)}{\partial v},$$

which is called the **tangent map** at the corresponding point. If ϕ satisfies

$$I_{S_1}(\boldsymbol{t}_1, \boldsymbol{t}_2) = k\, I_{S_2}\big(d\phi(\boldsymbol{t}_1), d\phi(\boldsymbol{t}_2)\big), \qquad \forall\, \boldsymbol{t}_1, \boldsymbol{t}_2 \in T_pS_1,$$

where k depends only on p and not on $\boldsymbol{t}_1, \boldsymbol{t}_2$, then ϕ is called a **conformal map**. In particular, if $k \equiv 1$, it is called an **isometric map**. By the Theorema Egregium, the Gaussian curvature of a surface is preserved under isometric maps. Hence an isometric map can exist only between surfaces with equal Gaussian curvature. If we regard the parameter domain as a surface and the

parameterization map as a map between surfaces, we can discuss the conformality and isometry of the parameterization map itself. Consequently, only surfaces whose Gaussian curvature is identically zero can admit an isometric parameterization.

A ruled surface whose Gaussian curvature is identically zero is called a **developable surface**, for example cylinders, cones, and tangent developables. As the name suggests, these surfaces can be developed (i.e., isometrically parameterized) onto the plane. For a general surface, it can be proved that every point has a coordinate neighborhood that can be conformally mapped to a planar region. Such parameters are called **isothermal coordinates**.

4. Geodesics and the Gauss–Bonnet Formula

Given a curve on a surface, the projection of its curvature vector onto the tangent plane is called the **geodesic curvature vector**, and its magnitude is called the **geodesic curvature**, denoted k_g; it depends only on the first fundamental form. A curve on a surface whose geodesic curvature is identically zero is called a **geodesic**. It follows easily from the definition that if a surface contains a straight line, the line must be a geodesic. The image of a geodesic under an isometric map is also a geodesic. Moreover, the length of a geodesic between two points on a surface is an extremum of the lengths of all curves between the two points. On the sphere, the geodesics between two points are the great-circle arcs connecting them (both the minor arc and the major arc).

If D is a simply connected region on a surface S whose boundary is a closed piecewise smooth curve, then the **Gauss–Bonnet formula** holds:

$$\iint_D K\,dA + \int_{\partial D} k_g\,ds + \sum \alpha_i = 2\pi,$$

where K is the Gaussian curvature and the α_i are the exterior angles at the non-smooth points of the boundary of D. In particular, for a geodesic triangle on a surface,

$$\iint_D K\,dA = \beta_1 + \beta_2 + \beta_3 - \pi,$$

where $\beta_1, \beta_2, \beta_3$ are the three interior angles. For the sphere, whose Gaussian curvature is identically 1, the above formula shows that the area of a spherical geodesic (great-circle-arc) triangle is exactly $\beta_1 + \beta_2 + \beta_3 - \pi$.

5. Functions on Surfaces and the Laplace Operator

Given an orthonormal frame $\{\boldsymbol{e}_1, \boldsymbol{e}_2\}$ on a surface S and a tangent vector field $\boldsymbol{v}$, the projection of its differential onto the tangent plane is called its **covariant differential**:

$$D\boldsymbol{v} = (d\boldsymbol{v} \cdot \boldsymbol{e}_1)\,\boldsymbol{e}_1 + (d\boldsymbol{v} \cdot \boldsymbol{e}_2)\,\boldsymbol{e}_2.$$

The covariant differential is a linear differential operator.

Given a function ϕ on the surface S, its differential can be expressed as a combination of 1-forms on S:

$$d\phi = \phi_1\omega_1 + \phi_2\omega_2.$$

The **gradient** of the function ϕ is defined as

$$\nabla\phi = \phi_1\boldsymbol{e}_1 + \phi_2\boldsymbol{e}_2,$$

which is independent of the choice of frame. Its covariant differential is

$$D(\nabla\phi) = (d\phi_1 + \phi_2\omega_{21})\,\boldsymbol{e}_1 + (d\phi_2 + \phi_1\omega_{12})\,\boldsymbol{e}_2,$$

written $D\phi_1 \boldsymbol{e}_1 + D\phi_2 \boldsymbol{e}_2$. Note that $D\phi_1, D\phi_2$ are also 1-forms, so they too can be written as combinations of ω_1, ω_2:

$$D\phi_1 = d\phi_1 + \phi_2\omega_{21} = \phi_{11}\omega_1 + \phi_{12}\omega_2,$$
$$D\phi_2 = d\phi_2 + \phi_1\omega_{12} = \phi_{21}\omega_1 + \phi_{22}\omega_2.$$

The 2×2 matrix formed by $\phi_{11}, \phi_{12}, \phi_{21}, \phi_{22}$ is called the **Hessian matrix** of ϕ with respect to $\{\boldsymbol{e}_1, \boldsymbol{e}_2\}$. In particular, the trace of this matrix, $\phi_{11} + \phi_{22}$, is independent of the choice of frame and is denoted $\Delta_S\phi$. The operator Δ_S is called the **Laplace–Beltrami operator** of the surface, or simply the Laplace operator. In particular, it can be proved that

$$\Delta_S \boldsymbol{f} = -2H\boldsymbol{n}.$$

For a general parameterization, the Laplace operator of the surface can be computed by the following formula:

$$\Delta_S\phi = \frac{1}{\sqrt{g}} \frac{\partial}{\partial u^\beta} \left(\sqrt{g}\, g^{\alpha\beta} \frac{\partial \phi}{\partial u^\alpha} \right),$$

where (u^1, u^2) are the parameters, $g^{\alpha\beta}$ are the matrix entries of the first fundamental form of the surface, and g is the determinant of that matrix. The above formula uses the Einstein summation convention.

1.1.3 Global Theory of Surfaces [1]

1. Orientability

Let S be a surface and $\mathfrak{U} = \{(U_\alpha, \phi_\alpha)\}$ an atlas of it. For each chart (U_α, ϕ_α), choose a unit normal $\boldsymbol{n}_\alpha$ as its orientation. If there exists a choice such that

$$\boldsymbol{n}_\alpha = \boldsymbol{n}_\beta, \qquad \forall\, U_\alpha \cap U_\beta \neq \emptyset,$$

then the surface is called **orientable**. An orientable surface has a globally well-defined unit normal, and hence a globally well-defined second fundamental form. Common non-orientable surfaces include the Möbius strip and the Klein bottle.

2. Triangulation of Surfaces

Let S be a surface and $\mathfrak{U} = \{(U_\alpha, \phi_\alpha)\}$ an atlas of it. A **triangle** T on the surface is a simply connected region within some U_α bounded by three smooth curves on the surface. A **triangulation** of the surface is a family of triangles $\{T_\lambda\}$ on the surface such that

1) every point p on the surface belongs to some triangle, and if p is an interior point of a triangle, that triangle is unique;

2) every edge belongs to at most two triangles; two triangles share at most one edge, and if two triangles share an edge, they also share both endpoints of that edge;

3) every vertex belongs to at most finitely many triangles.

It can be proved that every bounded region on a surface with a piecewise smooth boundary admits a triangulation.

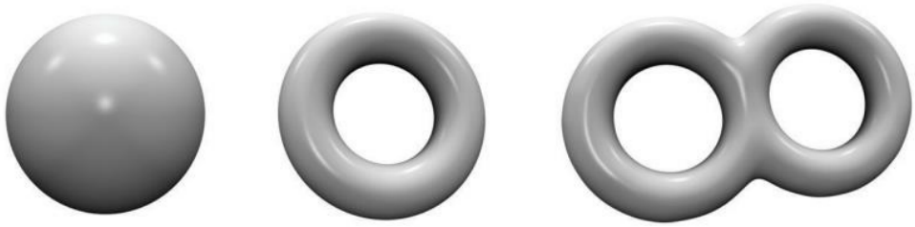


Figure 1.2: Surfaces of different genus [3]

3. Euler Characteristic

Let S be a surface and $\{T_\lambda\}$ a triangulation of it. Denote the numbers of vertices, edges and triangles by V, E, F, respectively. Euler's formula states that $V-E+F$ is a topological invariant; it is denoted by $\chi(S)$ and called the **Euler characteristic** of the surface. It equals $2-2g$, where g is the **genus** of the surface, which counts the number of holes in the surface.

Let S be an orientable compact surface without boundary. Then the global Gauss–Bonnet formula holds:

$$\int_S K\, dA = 2\pi\chi(S).$$

The classification theorem of surfaces states that a surface of genus 0 is homeomorphic to the sphere, and a surface of nonzero genus is homeomorphic to a torus with the same number of genus. Consequently, a closed surface cannot be globally parameterized onto the plane.

1.2 Triangle Mesh Surfaces

1.2.1 Surface Representations in Practice

In mathematics, there are three main kinds of surfaces that are commonly studied:

1) Explicit surfaces: $z = f(x, y)$.

2) Parametric surfaces: $(x, y, z) = \big(x(u, v), y(u, v), z(u, v)\big)$, $(u, v) \in \Omega \subseteq \mathbb{R}^2$.

3) Implicit surfaces: $\phi(x, y, z) = 0$.

Explicit surfaces can be regarded as special parametric surfaces. What these surfaces have in common is that their representations are very concise and intuitive, and the functions involved can usually be assumed to have a certain degree of smoothness. Therefore, the standard tools of analysis are well suited to studying the geometric properties of these surfaces, from which all kinds of beautiful theories are derived.

However, the surfaces found in real life are complex and varied, and are often difficult to represent with intuitive functions. Moreover, since computers store data discretely, there is no data structure that can store all the information of a general surface completely, so a surface can only be represented approximately by a finite amount of sampled information. A triangle mesh is a simple and effective linear approximation of a general surface.

More specifically, a **triangle mesh** M consists of a graph $G = (V, E, F)$ and a coordinate map $X\colon V \to \mathbb{R}^3$, where V is a set of vertices, E is a set of edges connecting these vertices, and F is a set of triangles formed by these vertices and edges; together they encode the topological relations within the mesh. In order to form a triangulation of a general orientable surface, they must satisfy the following conditions:

1) every vertex belongs to at least three edges, and two vertices are connected by at most one edge;

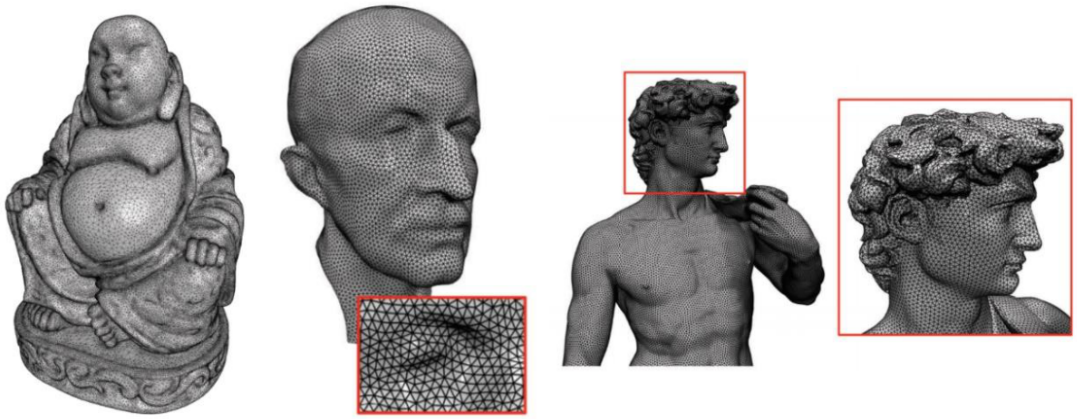


Figure 1.3: Triangle mesh surfaces [3]

2) every edge belongs to at least one and at most two triangles;

3) two triangles share at most one edge and at most two vertices;

4) there exists a consistent orientation of all the triangles.

The coordinate map X maps each vertex v to its position $\boldsymbol{x}_v$ in space, which encodes the geometric information of the mesh. Thus a triangle $f \in F$ with vertices v_1, v_2, v_3 corresponds to a triangular face in three-dimensional Euclidean space:

$$(v_1, v_2, v_3) \leftrightarrow \left\{ \lambda_1 \boldsymbol{x}_{v_1} + \lambda_2 \boldsymbol{x}_{v_2} + \lambda_3 \boldsymbol{x}_{v_3} \;\middle|\; \lambda_1, \lambda_2, \lambda_3 \geq 0,\ \lambda_1 + \lambda_2 + \lambda_3 = 1 \right\}.$$

These triangles linearly approximate the corresponding parts of the surface. For these triangles to fit together into a manifold surface, the coordinate map must satisfy:

1) the 1-ring neighborhood of every vertex (i.e., the union of all triangles it belongs to) is homeomorphic to the closed unit disk;

2) any two triangles have no intersection other than shared vertices and edges.

Finally, by gluing these triangles together we obtain a piecewise linear map as an approximation of the surface. It can be shown that, by increasing the number of vertices of the mesh, a general orientable manifold surface can be approximated to arbitrary precision.

Surfaces in practical applications usually come from two sources: computer-aided design (CAD) and 3D scanning. In computer-aided design, engineers or artists create object models with various 3D modeling software packages (such as AutoCAD, 3ds Max, Maya, Cinema 4D, Blender, Houdini, etc.). These models are often composed of basic elements (such as cuboids, spheres, cylinders, triangles, Bézier curves, etc.), so the exported models are usually stored as spline surfaces or polygonal meshes. In 3D scanning, a scanner scans the surface of an object to obtain dense sample points. These sample points alone, however, do not carry any global information about the surface, so the connectivity between the points must be constructed; this process is called **meshing**. Through meshing, the sample points become a triangle mesh.

In geometry processing algorithms, we often need to access the two endpoints or the two adjacent triangles of an edge, or to iterate over all edges, triangles, or vertices incident to a vertex; the halfedge data structure is therefore used to perform these operations efficiently. In the halfedge data structure, each edge is split into two oppositely oriented halfedges, each belonging to one of the two adjacent triangles. The halfedges within each triangle are oriented counterclockwise. Since the approximated surface is orientable, such a splitting always exists. Here we use the triangle mesh class provided by OpenMesh [4] as the basic mesh data structure, and use its API to perform the various traversal operations on the triangle mesh.

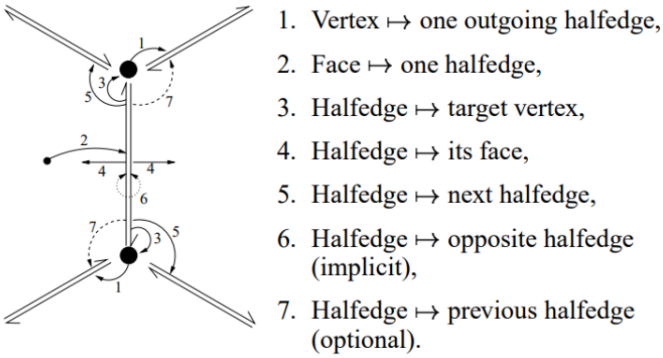


Figure 1.4: The halfedge data structure [4]

1.2.2 Discrete Differential Geometry

With the triangle mesh data structure as an approximation of a smooth surface, we can compute various geometric quantities. However, the geometric quantities of differential geometry are all defined under the assumption that the surface is sufficiently smooth, and they often involve higher-order derivatives of the surface map. A triangle mesh surface, by contrast, is piecewise linear: within each triangle the first derivatives are constant and all higher-order derivatives vanish, which clearly does not agree with the geometry of the original surface. Therefore, the concepts of the smooth setting cannot be applied directly in actual computation, and a certain discretization is required. **Discrete differential geometry** is precisely the tool designed for this purpose: it aims to compute, by simple methods on a triangle mesh surface, geometric quantities such as tangent planes, normals, area elements, curvatures, and geodesics, so that they are good estimates of the corresponding quantities of the original surface. Here we only introduce the computation of some geometric quantities relevant to this thesis; for the rest, see [6, 3, 22].

1. Triangle-based quantities

Let the positions of the three vertices of a triangle $f = \{f_0, f_1, f_2\}$, in counterclockwise order, be $\boldsymbol{x}_{f_0}, \boldsymbol{x}_{f_1}, \boldsymbol{x}_{f_2}$.

1) Area: $|f| = \frac{1}{2}\big\|(\boldsymbol{x}_{f_1} - \boldsymbol{x}_{f_0}) \times (\boldsymbol{x}_{f_2} - \boldsymbol{x}_{f_0})\big\|$;

2) Normal: $\boldsymbol{n}_f = \dfrac{(\boldsymbol{x}_{f_1} - \boldsymbol{x}_{f_0}) \times (\boldsymbol{x}_{f_2} - \boldsymbol{x}_{f_0})}{\|(\boldsymbol{x}_{f_1} - \boldsymbol{x}_{f_0}) \times (\boldsymbol{x}_{f_2} - \boldsymbol{x}_{f_0})\|}$;

3) Barycenter: $\boldsymbol{x}_f = \frac{1}{3}\big(\boldsymbol{x}_{f_0} + \boldsymbol{x}_{f_1} + \boldsymbol{x}_{f_2}\big)$.

2. Vertex-based quantities

1) Area element: $\sigma_i = \frac{1}{3}\sum_{i \in f} |f|$;

2) Vertex normal: $\boldsymbol{n}_i = \dfrac{\sum_{i \in f} |f|\, \boldsymbol{n}_f}{\| \sum_{i \in f} |f|\, \boldsymbol{n}_f \|}$.

3. Global quantities

1) Euler characteristic: $\chi(M) = |V| - |E| + |F|$;

2) Genus: $g(M) = \frac{1}{2}\big(2 - \chi(M)\big)$;

3) Total area: $A(M) = \sum_{f \in F} |f|$;

4) Global barycenter: $\boldsymbol{x}_c(M) = \frac{1}{A(M)} \sum_{f \in F} |f| \, \boldsymbol{x}_f$.

4. Functions on a mesh

A function on a triangle mesh is given by its values at the vertices; its values inside each triangle are obtained by linear interpolation of the values at the three vertices. For a function f defined on the mesh vertices, let its value at vertex $i \in V$ be $f_i \in \mathbb{R}$.

1) Integral of the function: $\int_M f \, d\sigma = \sum_{i \in V} \sigma_i f_i$;

2) Laplacian of the function: $\Delta f_i = \frac{1}{\sigma_i} \sum_{(i,j) \in E} \omega_{ij} (f_j - f_i)$, where

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

is called the **cotangent weight**. For the detailed derivation, see [3, 6]. The matrix formed by the cotangent weights,

$$L(M) = (L_{ij})_{i,j \in V}, \qquad L_{ij} = \begin{cases} \sum_{(i,k) \in E} \omega_{ik}, & i = j, \\ -\omega_{ij}, & (i,j) \in E, \\ 0, & \text{otherwise}, \end{cases}$$

is called the **Laplacian matrix** of the surface. It is a positive semidefinite matrix:

$$\sum_{i,j \in V} L_{ij} f_i f_j = -\sum_{i \in V} \sigma_i f_i \Delta f_i = -\int_M f \Delta f \, d\sigma = \int_M |\nabla f|^2 \, d\sigma \geq 0.$$

The Laplacian matrix of a surface is very useful in geometry processing, and it will also be used repeatedly in the geometric optimization introduced in the following chapters.

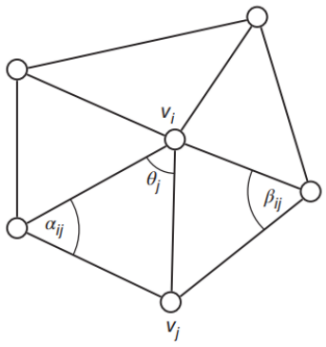


Figure 1.5: The cotangent weight [22]

1.3 The Mesh Parameterization Problem

1.3.1 Overview of the Problem

For a smooth orientable surface, the neighborhood of every point can be represented by a planar region through a parameter map. Hence every local region of a smooth surface is a parametric surface. Conversely, given a local region, the problem of finding a suitable parameter domain and

parameter map is the **parameterization** problem. More specifically, for a region U on a surface S, the parameterization problem seeks a suitable parameter domain Ω and a homeomorphism $\phi\colon U \simeq \Omega$. If U is only a local region, the parameter domain can be chosen to be a planar region; if U is the entire surface S, the parameter domain must be homeomorphic to the surface, and its choice must take into account the global properties and certain topological quantities of the surface, such as its genus and whether it is open or closed; the corresponding parameterization problem is called the **global parameterization** problem. In particular, for an open surface the global parameter domain can still be chosen to be a planar region, whereas for a closed surface of genus 0 the global parameter domain is usually chosen to be the unit sphere.

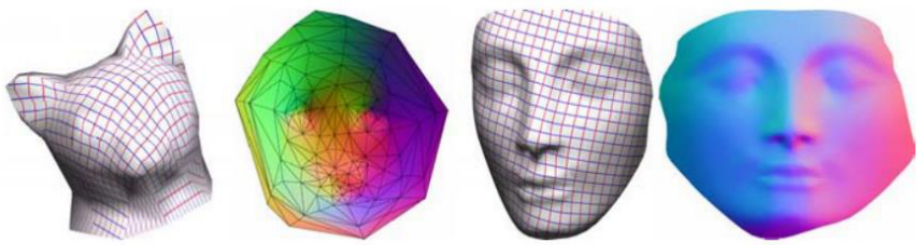


Figure 1.6: Planar parameterization of meshes [39]

In computer graphics, mesh parameterization is one of the most fundamental geometry processing problems [39, 10]. Many applications, including texture mapping, remeshing, mesh deformation, inter-surface mapping, and geometry images, require mesh parameterization as a basic tool. Since meshes in practical applications usually come from scanning the surfaces of objects, they are usually closed. One approach is to cut such a mesh into one or more open meshes (Figure 1.9) and then parameterize each of them onto a planar region. Its advantage is that a large number of high-quality planar parameterization algorithms are available; its disadvantage is that cutting the mesh makes the parameterization of the cut regions discontinuous along the cuts, which produces visible **seams** on the original mesh. Since some applications are sensitive to this discontinuity, meeting their practical requirements calls for additional **seamless constraints** at the boundaries, which are difficult to handle. In contrast, the other approach is to parameterize the mesh globally, so that the above discontinuity does not arise. Its disadvantage is that relatively few global parameterization algorithms are available.

For a function on a triangle mesh surface whose range is a linear space, the function values at points on the boundary and in the interior of each triangle can generally be approximated by linearly interpolating the values at the vertices with barycentric coordinates:

$$\phi\big(\lambda_0 \boldsymbol{x}_{f_0} + \lambda_1 \boldsymbol{x}_{f_1} + \lambda_2 \boldsymbol{x}_{f_2}\big) \approx \lambda_0 \phi(\boldsymbol{x}_{f_0}) + \lambda_1 \phi(\boldsymbol{x}_{f_1}) + \lambda_2 \phi(\boldsymbol{x}_{f_2}).$$

Therefore, to solve a planar parameterization problem it suffices to solve for the parameter coordinates at each vertex. In this sense, the parameterization map can be regarded as a **deformation** of the mesh into the plane, and the necessary and sufficient conditions for this deformation to be a homeomorphism are:

1) Local bijectivity: the map is bijective on every local region (i.e., on any two triangles sharing an edge). This is equivalent to every pair of adjacent triangles having the same orientation after the deformation, and also equivalent to there being no flipped triangles.

2) Global bijectivity: the map is bijective on the entire region to be parameterized. This is equivalent to the boundary of the region to be parameterized being free of self-intersections after the deformation.

Only a parameterization map satisfying these two conditions can be called **valid**. The basic requirement of the mesh parameterization problem is to compute such a valid parameterization

map. The advantage of the planar parameter domain lies in its linearity, which makes the parametric relation between each vertex and its surrounding vertices linear; this has given rise to a large class of linear methods. From the deformation point of view, the whole parameterization map can be regarded as a $2|V|$-dimensional vector whose components are the parameter coordinates of the vertices. The linearity of the plane then also allows the two parameter coordinates of each vertex to vary linearly, subject only to the local bijectivity constraint; this has given rise to a large class of iterative methods based on line search. For these reasons, planar parameterization has been studied extensively.

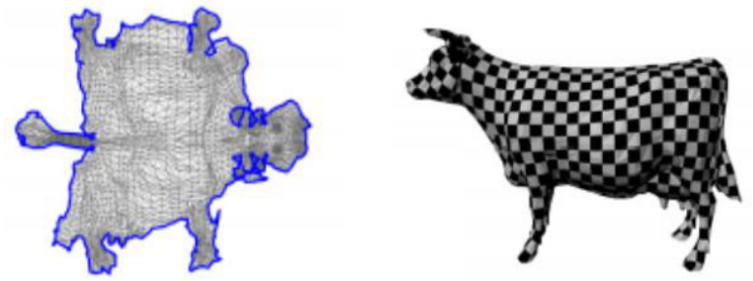


Figure 1.7: Planar parameterization and texture mapping [43]

For the global parameterization of a closed triangle mesh of genus 0, since the parameter domain is the unit sphere, the parameters inside the triangles cannot be obtained by linearly interpolating the parameters at the vertices; instead, a homeomorphism from planar triangles to spherical triangles must be specified in advance. Such a map is called a **spherical triangle map** and will be introduced in detail in Chapter 3. Once a spherical triangle map is given, the problem again reduces to computing the parameter coordinates at the vertices. Since the mesh now has no boundary, it suffices for the parameterization map to satisfy local bijectivity, i.e., there are no flipped spherical triangles. Compared with the planar parameter domain, the nonlinear constraint imposed by the sphere makes spherical parameterization harder to compute: the linear methods for the plane no longer apply, and the corresponding iterative methods are difficult to carry over directly. Consequently, spherical parameterization has received much less research attention than planar parameterization.



Figure 1.8: Spherical parameterization and texture mapping [19]

1.3.2 Distortion Energy

To quantitatively describe the distortion of a parameterization map over the whole triangle mesh, we first need to define its distortion on each triangle. Given a triangle mesh $M = (V, E, F)$ and a planar parameterization map $\boldsymbol{x}$, denote the geometric position of a vertex $v \in V$ by $\bar{\boldsymbol{x}}_v \in \mathbb{R}^3$ and its parameter coordinates by $\boldsymbol{x}_v \in \mathbb{R}^2$. Given a triangle $f = (f_0, f_1, f_2) \in F$, we use a rigid transformation to map the triangle with vertices $\bar{\boldsymbol{x}}_{f_0}, \bar{\boldsymbol{x}}_{f_1}, \bar{\boldsymbol{x}}_{f_2}$ to the plane, such that



Figure 1.9: Cutting a closed mesh [34]

$\bar{\boldsymbol{x}}_{f_0}$ is mapped to the origin, $\bar{\boldsymbol{x}}_{f_1}$ to the positive x-axis, and $\bar{\boldsymbol{x}}_{f_2}$ to the upper half-plane:

$$(\bar{\boldsymbol{x}}_{f_0}, \bar{\boldsymbol{x}}_{f_1}, \bar{\boldsymbol{x}}_{f_2}) \mapsto (\boldsymbol{u}_{f_0}, \boldsymbol{u}_{f_1}, \boldsymbol{u}_{f_2}) = \begin{pmatrix} 0 & u^x_{f_1} & u^x_{f_2} \\ 0 & 0 & u^y_{f_2} \end{pmatrix},$$

where

$$\begin{cases} u^x_{f_1} = \|\bar{\boldsymbol{x}}_{f_1} - \bar{\boldsymbol{x}}_{f_0}\| \\ u^x_{f_2} = (\bar{\boldsymbol{x}}_{f_1} - \bar{\boldsymbol{x}}_{f_0}) \cdot (\bar{\boldsymbol{x}}_{f_2} - \bar{\boldsymbol{x}}_{f_0}) / u^x_{f_1} \\ u^y_{f_2} = \|(\bar{\boldsymbol{x}}_{f_1} - \bar{\boldsymbol{x}}_{f_0}) \times (\bar{\boldsymbol{x}}_{f_2} - \bar{\boldsymbol{x}}_{f_0})\| / u^x_{f_1} \end{cases}.$$

The original triangle is then congruent to the triangle formed by the new coordinates $\boldsymbol{u}_{f_0}, \boldsymbol{u}_{f_1}, \boldsymbol{u}_{f_2}$. The purpose of this step is to reduce the space in which the triangles of the original mesh lie from three dimensions to two, which makes it convenient to compute the deformation of each triangle later. We call the flattened triangle the **source triangle**. Since we use the halfedge data structure, the orientation of each source triangle agrees with the orientation of the mesh itself. There is a unique affine transformation from the source triangle to the parameterized image of the triangle:

$$\boldsymbol{x}_{f_i} = J_f \boldsymbol{u}_{f_i} + \boldsymbol{x}_{f_0}, \qquad i = 1, 2, 3,$$

where

$$J_f = (\boldsymbol{x}_{f_1} - \boldsymbol{x}_{f_0}, \boldsymbol{x}_{f_2} - \boldsymbol{x}_{f_0})(\boldsymbol{u}_{f_1}, \boldsymbol{u}_{f_2})^{-1}$$

is the gradient of this map, called the **deformation gradient** of triangle f; it characterizes how much the triangle is deformed. When the deformation gradient is an orthogonal matrix, the image triangle is congruent to the source triangle, and triangle f is then undistorted. When the determinant of the deformation gradient is negative, the orientation of the image triangle disagrees with that of the source triangle, and triangle f is flipped; when the determinant of the deformation gradient is 0, the image triangle degenerates into a line segment. Therefore, the necessary and sufficient condition for triangle f not to be flipped is

$$\det J_f > 0,$$

which is equivalent to

$$\det(\boldsymbol{x}_{f_1} - \boldsymbol{x}_{f_0}, \boldsymbol{x}_{f_2} - \boldsymbol{x}_{f_0}) > 0.$$

Let the singular value decomposition of the deformation gradient be

$$J_f = U_f \operatorname{diag}(\sigma_{1,f}, \sigma_{2,f}) V_f^T,$$

where U_f, V_f are orthogonal matrices and $\sigma_{1,f}, \sigma_{2,f}$ are the singular values of the deformation gradient, which are the extreme values of the stretch factors of this affine transformation over all directions. Let

$$R_f = U_f V_f^T, \qquad S_f = V_f \operatorname{diag}(\sigma_{1,f}, \sigma_{2,f}) V_f^T;$$

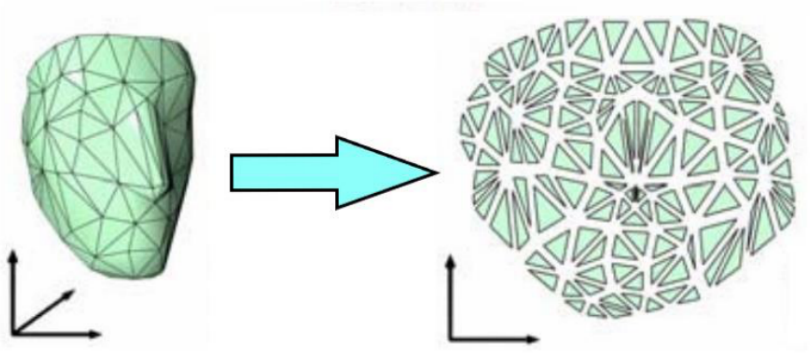


Figure 1.10: Flattening the triangles of a mesh to obtain the source triangles [24]

then $J_f = R_f S_f$ is the polar decomposition of the deformation gradient. From the above definitions, S_f is a symmetric matrix, called the **stretch tensor** of triangle f; geometrically, it represents stretching along two mutually orthogonal principal directions. R_f is an orthogonal matrix; geometrically, it represents a rigid transformation and does not contribute to the deformation of the triangle. It can be shown that, among all orthogonal matrices, R_f is the one closest to J_f in Frobenius distance. In other words, R_f is the projection of J_f onto the space of orthogonal matrices. We can therefore naturally use the Frobenius distance from J_f to R_f to describe the distortion of triangle f; this is called the ARAP (As-Rigid-As-Possible) distortion function [24]:

$$D_{ARAP}(J_f) = \frac{1}{2}\|J_f - R_f\|_F^2.$$

Note that this function can also be expressed in terms of the two singular values of the deformation gradient:

$$D_{ARAP}(J_f) = \frac{1}{2}(\sigma_{1,f} - 1)^2 + \frac{1}{2}(\sigma_{2,f} - 1)^2.$$

Clearly, this distortion function attains its minimum value 0 when the deformation gradient is an orthogonal matrix. Weighting the ARAP distortion of each triangle by its area, we obtain the ARAP distortion energy of the whole mesh [24]:

$$E_{ARAP}(M, \boldsymbol{x}) = \frac{1}{A(M)} \sum_{f \in F} |f|\, D_{ARAP}(J_f).$$

Similarly, we can define the distortion of each triangle in other ways, and then obtain the distortion energy of the whole mesh by area weighting. Commonly used distortion functions include:

1) Symmetric Dirichlet distortion [43]:

$$D_{symDir}(J_f) = \frac{1}{2}\left(\|J_f\|_F^2 + \|J_f^{-1}\|_F^2\right) = \frac{1}{2}\left(\sigma_{1,f}^2 + \sigma_{2,f}^2 + \frac{1}{\sigma_{1,f}^2} + \frac{1}{\sigma_{2,f}^2}\right).$$

This function attains its minimum value 2 when J_f is an orthogonal matrix. Because this distortion function has a simple form and works well, it is generally the objective function of choice in parameterization problems. In the rest of this thesis, the symmetric Dirichlet distortion is also used as the main objective function.

2) Conformal distortion (MIPS distortion [18]):

$$D_{MIPS}(J_f) = \frac{1}{2}\frac{\|J_f\|_F^2}{\det J_f} = \frac{1}{2}\left(\frac{\sigma_{1,f}}{\sigma_{2,f}} + \frac{\sigma_{2,f}}{\sigma_{1,f}}\right).$$

This function attains its minimum value 1 when $\sigma_{1,f} = \sigma_{2,f}$. Optimizing this distortion yields a conformal parameterization map.

3) Area-preserving distortion:

$$D_{area}(J_f) = \frac{1}{2}\Big(\det J_f + \frac{1}{\det J_f}\Big) = \frac{1}{2}\Big(\sigma_{1,f}\sigma_{2,f} + \frac{1}{\sigma_{1,f}\sigma_{2,f}}\Big).$$

This function attains its minimum value 1 when $\det J_f = 1$. Optimizing this distortion yields an area-preserving parameterization map.

4) Isometric distortion [11]:

$$D_{iso}(J_f) = \alpha D_{MIPS}(J_f) + (1-\alpha)D_{area}(J_f), \qquad 0 < \alpha < 1.$$

This function is a combination of the conformal and the area-preserving distortions. It attains its minimum value 1 when J_f is an orthogonal matrix, and the parameter α adjusts how strongly area or angle distortion is penalized.

5) AMIPS (Advanced MIPS) distortion [11]:

$$D^*_{iso}(J_f) = \exp\big(sD_{MIPS}(J_f) + sD_{area}(J_f)\big).$$

This function effectively penalizes triangles with large distortion.

1.3.3 Formulation of Low-Distortion Parameterization

Finally, we formulate the low-distortion parameterization problem as an optimization problem whose variable is the parameterization map, whose constraint is its bijectivity, and whose objective function is its distortion energy over the entire surface:

$$\begin{aligned} \boldsymbol{x}^* &= \operatorname*{argmin}_{\boldsymbol{x}} E(M, \boldsymbol{x}), \\ \text{s.t. } &\det\big(\boldsymbol{x}_{f_1} - \boldsymbol{x}_{f_0}, \boldsymbol{x}_{f_2} - \boldsymbol{x}_{f_0}\big) > 0, \ \forall f, \end{aligned}$$

where

$$E(M, \boldsymbol{x}) = \frac{1}{A(M)} \sum_{f \in F} |f|\, D(J_f).$$

This is a nonlinear optimization problem with an open-set constraint. Note that we have not included a global bijectivity constraint here, because such a constraint can be incorporated into the energy function; see [43, 44] for details. In addition, for the low-distortion spherical parameterization problem, both the energy and the constraints are slightly different; they are given in Chapter 3. Although the objective functions and constraints of this class of nonlinear optimization problems have complicated forms, they possess a distinctive geometric structure. People have therefore modified common optimization algorithms using geometric knowledge, obtaining algorithms that can effectively decrease such functions while producing results that strictly satisfy the constraints. These are called **geometric optimization** methods.

1.4 An Introduction to Geometric Optimization Methods

1.4.1 General Nonlinear Optimization [26, 27]

In nonlinear optimization theory, a general nonlinear optimization problem is usually solved by starting from some initial point and converging step by step to an extremum through many

iterations. For unconstrained convex optimization problems, commonly used methods include gradient descent and Newton's method.

Gradient descent expands the objective function to first order and computes its direction of fastest decrease (i.e., the negative gradient direction); it is therefore also called the method of steepest descent. However, since it uses only first-order derivative information of the objective function, it is a first-order method, and so its convergence rate is linear. Common acceleration techniques include preconditioning, Nesterov acceleration [25], and Anderson acceleration [2]. **Preconditioning** applies a linear transformation to the gradient through a **preconditioner** before each iteration, in order to obtain a better descent direction. In essence, it defines a new Riemannian metric on the solution space to replace the original Euclidean metric; under this new metric, the negative gradient direction of the objective function is exactly the preconditioned descent direction. Nesterov acceleration defines a momentum that accumulates all negative gradients computed along the iterations, and uses it as the descent direction. In essence, it simulates a physical motion: the objective function is interpreted as a potential energy, so its negative gradient corresponds to a conservative force; momentum is the accumulation of force over time, and the momentum determines the current direction of motion. Anderson acceleration combines information from several previous iterates to improve the new iterate. In essence, it regards each new iterate as a function of the previous iterate, and thus estimates the fixed point of this function from the information of several previous iterates. In addition, choosing a suitable step size so that each iteration decreases the objective function as much as possible can also accelerate the whole iterative process. For large-scale problems, one can also use stochastic gradient descent or batch gradient descent to avoid computing the full gradient. Although these acceleration methods can all reduce the time needed for convergence, they cannot change the essentially linear convergence rate.

Newton's method expands the objective function to second order and takes the minimizer of this second-order approximation as the next iterate. Since it makes full use of the second-order derivative information of the objective function, it is a second-order method and can therefore achieve a superlinear convergence rate. Newton's method can also be viewed as preconditioning with the Hessian matrix of the objective function as the preconditioner. The drawback of Newton's method is that computing each new iterate is time-consuming. Although the Hessian of a convex function is a symmetric positive definite matrix, so that special methods such as the LL^T decomposition can be used to accelerate solving for the descent direction, computing the Hessian and solving its quadratic programming problem is ultimately time-consuming. A common acceleration technique is to avoid computing the Hessian and instead estimate its inverse directly from gradient information; such methods are called **quasi-Newton** methods.

For constrained optimization problems, we can modify the above methods according to the type of constraint. The overall idea is to start from an initial point that satisfies the constraints and to maintain the constraints during the iterations. Linear equality constraints can be written as a linear system in the variables of the objective function; this constraint can be maintained simply by projecting each descent direction onto the null space of the coefficient matrix. For general manifold constraints, one can use parametric coordinates on the manifold, or first project the gradient onto the tangent space of the manifold and then replace the line search with a curve search on the manifold. For inequality constraints, if the set satisfying the constraint in the solution space is closed, each iterate can be projected onto this closed set in some way; if it is open, one generally uses the **interior point method**, i.e., in each iteration one computes the largest step size that does not violate the constraint and then uses a backtracking search to determine a suitable descent step size.

For nonconvex optimization problems, gradient descent is still applicable, but Newton's method may cause the objective function to increase or the iterates to diverge, because the Hessian of the objective function is indefinite. A common remedy is to replace the Hessian with a positive definite or positive semidefinite matrix, similarly to preconditioning. A typical

replacement is to add a positive multiple λ of the identity matrix to the Hessian. In essence, this adds a regularization term to the objective function; the result is that λ is added to every eigenvalue of the Hessian, so λ must be at least the absolute value of the most negative eigenvalue of the Hessian. The drawback of this method is that the number of iterations needed for convergence increases greatly, and the larger λ is, the greater the increase. Another method is to find the positive semidefinite matrix closest to the Hessian, which is called **projected Newton**. Concretely, all negative eigenvalues of the Hessian are set to 0, which requires computing the eigendecomposition of the Hessian. The drawback of this method is that for large-scale problems the Hessian is too large and computing its eigendecomposition is very time-consuming; if analytic expressions for the eigenvalues and eigenvectors are known in advance, a large speedup can be obtained. Some quasi-Newton methods guarantee a positive semidefinite solver even for nonconvex functions, which avoids the eigendecomposition and the subsequent quadratic programming problem, and can greatly reduce the time required per iteration. However, the positive semidefinite approximation obtained in this way is usually not as good as that of projected Newton, so more iterations are needed to reach convergence.

1.4.2 Geometric Optimization

In the previous section, we formulated the low-distortion flip-free parameterization problem as a constrained optimization problem. Its objective function is highly nonconvex, and its constraint is a nonlinear, nonconvex, open-set constraint, which makes the optimization problem difficult to solve. However, the objective function has the special feature that it is obtained by summing the values of a local distortion function over all triangles, weighted by area, and the constraint is also imposed on each triangle separately. Moreover, the local problems are small and contain a great deal of geometric information. Based on these features, a special class of solution methods can be designed, called geometric optimization methods. Since such objective functions and constraints arise widely in graphics problems such as deformation and maps between surfaces, geometric optimization is one of the most important tools in graphics. Below we briefly describe the general procedure for solving the geometric optimization problem posed in the previous section.

First, we need to find an initialization that satisfies the constraints. This means that we must compute a bijective parameterization map. Fortunately, for planar parameterization we can obtain such an initialization with a linear method, by solving a sparse linear system. During the iterations, since the objective function is nonconvex, we can use preconditioning, projected Newton, or quasi-Newton methods to compute the descent direction. We first compute a solver for each triangle and then assemble them into a global solver. Such a solver preserves the sparsity of the problem well, which makes the corresponding quadratic programming problem easier to solve. To maintain the flip-free constraint, we need to compute a maximal step size and then use backtracking line search to obtain a suitable step size. Since the flip-free constraint is the union of the local constraints on the individual triangles, and the boundary of each local constraint is a quadratic polynomial in the parametric coordinates of the triangle, the maximal step size of each triangle can be obtained by solving a quadratic equation, and the minimum of the maximal step sizes over all triangles is taken as the global maximal step size. In this way, no triangle flips in any iteration, and the converged result is necessarily flip-free.

In practice, however, the iterations sometimes get stuck, and no solver is able to decrease the objective function effectively. This is usually because some triangles are close to flipping, which makes the global maximal step size very small. If the objective function is not chosen appropriately, this phenomenon occurs easily. For example, the ARAP distortion function introduced in the previous section takes large values on triangles with large stretch factors (i.e., singular values of the deformation gradient), which effectively suppresses such triangles; but its values on triangles with stretch factors close to 0 are small, so triangles easily come close to flipping, and the maximal step size then becomes too small. To avoid this, the local distortion function must also penalize triangles with stretch factors close to 0 sufficiently strongly. The symmetric

Dirichlet distortion and the AMIPS distortion mentioned in the previous section satisfy this requirement, and have therefore gradually replaced the ARAP distortion. In addition, some special techniques can be used to avoid small maximal step sizes, for example interpolating an intermediate triangle between the source triangle and the image triangle and using it in place of the source triangle as the reference triangle, or exploiting the form of the constraint to construct, with the original search direction as a reference, a search direction whose maximal step size is as large as possible.

It can thus be seen that geometric optimization methods for planar parameterization have been studied extensively; there is a fairly complete pipeline as well as many techniques for acceleration. In contrast, geometric optimization methods for spherical parameterization have rarely been studied, and even the most basic flip-free initialization is hard to obtain. This project therefore aims to generalize the geometric optimization pipeline of planar parameterization to the spherical parameterization problem, so as to efficiently obtain low-distortion, flip-free spherical parameterizations. To this end, we must first understand and implement the various methods for planar parameterization, study the problems that arise in existing spherical parameterization methods, and then find ways to solve them. The remaining chapters are organized roughly as follows: Chapter 2 describes the details of geometric optimization methods for planar parameterization; Chapter 3 surveys existing spherical parameterization methods and uses the gnomonic projection as a bridge to apply the geometric optimization methods of the plane to the spherical parameterization problem. Finally, Chapter 4 concludes and points out directions for future research.

Chapter 2

Geometric Optimization Methods for Planar Parameterization

2.1 Tutte's Method and Its Variants

In 1963, W. T. Tutte, in a paper on graph theory [46], described how to embed a 2-connected graph in the plane. The procedure is roughly as follows: first map the boundary of the graph onto the boundary of a planar region, and then map the remaining vertices into the interior of the region by solving a linear system. Tutte proved that as long as the boundary of the planar region is convex, this method guarantees that no two edges of the embedded graph intersect. Later, Michael S. Floater [8] applied this method to the planar parameterization of triangle meshes with boundary and successfully obtained flip-free planar parameterizations. Floater further generalized this method and proposed a class of linear flip-free planar parameterization methods [9]. Below we introduce this class of methods.

Given a triangle, every point in it can be expressed in barycentric coordinates:

$$\boldsymbol{x} = \lambda_1 \boldsymbol{x}_1 + \lambda_2 \boldsymbol{x}_2 + \lambda_3 \boldsymbol{x}_3, \qquad \lambda_1, \lambda_2, \lambda_3 \geq 0,\ \lambda_1 + \lambda_2 + \lambda_3 = 1.$$

In particular, a point lies in the interior of the triangle if and only if $\lambda_1, \lambda_2, \lambda_3 > 0$. A similar result holds for convex planar polygons: let the positions of the n vertices of a convex planar n-gon be $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$. If a point $\boldsymbol{x}$ can be expressed as a convex combination of these vertices, i.e.,

$$\boldsymbol{x} = \lambda_1 \boldsymbol{x}_1 + \lambda_2 \boldsymbol{x}_2 + \cdots + \lambda_n \boldsymbol{x}_n, \qquad \lambda_1, \lambda_2, \cdots, \lambda_n > 0,\ \lambda_1 + \lambda_2 + \cdots + \lambda_n = 1,$$

then the point must lie in the interior of this convex n-gon. This result can be proved by induction on the number of sides of the convex polygon, starting from the triangle. A natural question is then: if the boundary of a planar triangle mesh is a convex polygon, and every interior point can be expressed as a convex combination of the vertices in its 1-ring neighborhood, is it true that no two edges of this triangle mesh intersect? Tutte gave an affirmative answer and proposed the following method of computation. Let the vertex set of a boundary of the mesh M be $B = \{v_1, \cdots, v_b\} \subseteq V$, ordered clockwise. Take any convex planar b-gon, and let its vertex positions, ordered clockwise, be $\boldsymbol{\xi}_1, \cdots, \boldsymbol{\xi}_b$. Then the planar parametric coordinates $\{\boldsymbol{x}_i\}_{i \in V}$ of the vertices of M can be obtained by solving the following system:

$$\begin{cases} \boldsymbol{x}_{v_k} = \boldsymbol{\xi}_k, & k = 1, \cdots, b \\ \sum\limits_{(i,j) \in E} \boldsymbol{x}_i - \boldsymbol{x}_j = 0, & i \notin B \end{cases},$$

which is a sparse linear system in the planar parametric coordinates. By Tutte's result [46], the resulting parameterization map is necessarily bijective. Since each vertex's parametric

coordinates lie at the barycenter of the parametric coordinates of its 1-ring neighbors, such a parameterization map is also called a **barycentric map**.

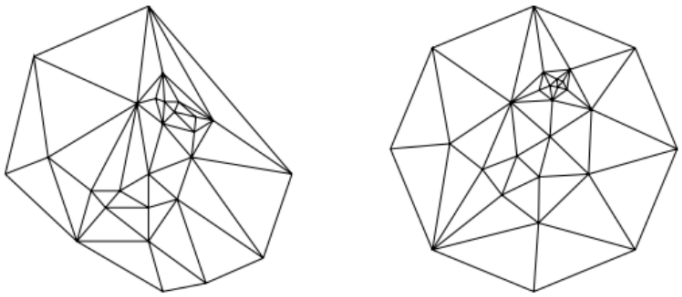


Figure 2.1: Barycentric map obtained by Tutte's method [8]

In the method above, each vertex weights its 1-ring neighbors uniformly. Floater generalized it [8] to a linear method with general weights: for each edge $(i,j) \in E$, specify a weight $\omega_{ij} > 0$. The parametric coordinates of the vertices can then be obtained by solving the following weighted system:

$$\begin{cases} \boldsymbol{x}_{v_k} = \boldsymbol{\xi}_k, & k = 1, \cdots, b \\ \sum\limits_{(i,j)\in E} \omega_{ij} \left(\boldsymbol{x}_i - \boldsymbol{x}_j\right) = 0, & i \notin B \end{cases} .$$

Floater proved [9] that the solutions of such systems are all bijective parameterization maps, and that the properties of the map can be controlled through the choice of ω_{ij}. Some particular choices of weights include:

1) Inverse edge length: $\omega_{ij} = \dfrac{1}{\|\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_j\|}$

2) Cotangent weights: $\omega_{ij} = \dfrac{1}{2}\left(\cot\alpha_{ij} + \cot\beta_{ij}\right)$

$$\frac{1}{2}\left(\cot\alpha_{ij} + \cot\beta_{ij}\right)$$

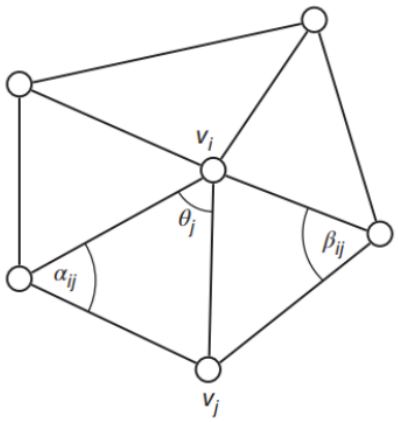


Figure 2.2: Cotangent weights

Floater pointed out in [8] that the parameterization obtained with cotangent weights has good **shape-preserving** properties. However, if the mesh contains two adjacent triangles whose opposite angles are both acute, the cotangent weight of their shared edge is negative, and the

computed parameterization map is then not necessarily bijective. A common remedy is to replace negative weights with 0 or with a very small positive number.

It is worth noting that the linear system solved in the above method is exactly equivalent to the following optimization problem [8]:

$$\begin{aligned} \boldsymbol{x}^* &= \operatorname*{argmin}_{\boldsymbol{x}_i} \frac{1}{2} \sum_{(i,j)\in E} \omega_{ij} \left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|^2, \\ \text{s.t. } \boldsymbol{x}_{v_k} &= \boldsymbol{\xi}_k, \qquad k = 1, \cdots, b. \end{aligned}$$

Hence it is in essence also a geometric optimization problem. The objective function in this problem is equivalent to the elastic potential energy of a spring system in which each edge is a spring with stiffness ω_{ij} and rest length 0; it is therefore also called the **string energy**. This means that if the boundary of such a spring system is fixed to a prescribed convex polygon, the positions of the vertices when the system finally reaches equilibrium are exactly the parametric coordinates.

In summary, Tutte's method is a simple linear method for computing bijective parameterizations, and it has a special geometric and physical meaning. However, the parameterizations it produces usually have large distortion: the parametric coordinates of some vertices cluster near the barycenter or the boundary of the convex polygon, producing extremely small or long, thin triangles in those regions. Nevertheless, we can still use it as the initialization for optimizing the distortion energy, since we only need it to satisfy the most basic flip-free condition. In the next section we describe in detail the iterative process for optimizing the distortion energy.

2.2 Iterative Solvers

With a bijective initialization in hand, it is time to enter the iterative process to decrease the distortion energy. Since the distortion energy is a nonconvex function, commonly used methods include preconditioning, projected Newton, and quasi-Newton methods. The first is generally a first-order method, while the latter two are second-order methods.

2.2.1 Preconditioners

1. Sobolev Preconditioner [24, 21]

Let us first see how to decrease the ARAP distortion energy. Recall the definition of the ARAP energy:

$$E_{ARAP}(M, \boldsymbol{x}) = \frac{1}{A(M)} \sum_{f\in F} |f|\, D_{ARAP}(J_f), \qquad D_{ARAP}(J_f) = \frac{1}{2} \left\| J_f - R_f \right\|_F^2,$$

where

$$J_f = (\boldsymbol{x}_{f_1} - \boldsymbol{x}_{f_0},\ \boldsymbol{x}_{f_2} - \boldsymbol{x}_{f_0})\,(\boldsymbol{u}_{f_1}, \boldsymbol{u}_{f_2})^{-1} = U_f \operatorname{diag}(\sigma_{1,f}, \sigma_{2,f})\, V_f^T, \qquad R_f = U_f V_f^T.$$

To optimize the distortion energy, we first need its gradient. Since the local distortion function is a function of J_f, and J_f is linear in $\boldsymbol{x}$, we only need to take the gradient with respect to J_f. However, computing the gradient of R_f with respect to J_f is not easy. Instead, we can regard R_f as an auxiliary variable and iterate by alternately updating R_f and J_f: for the current $\boldsymbol{x}$, compute each J_f and hence R_f; then fix all R_f and update $\boldsymbol{x}$, which gives new J_f; repeat this process until convergence. Since the first step is a local update and the second is a global update, this method is also called the **local–global method** [24]. This method is very effective for optimizing the ARAP energy, because once all R_f are fixed, the energy is a quadratic function

of J_f (and hence of $\boldsymbol{x}$), so the second step becomes a quadratic programming problem. Its quadratic-form matrix is

$$\Delta_{\boldsymbol{x}} E_{ARAP}(M, \boldsymbol{x}) = \frac{1}{A(M)} \sum_{f \in F} |f| \, I_2 \otimes P_f P_f^T,$$

where

$$P_f = \begin{pmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} (\boldsymbol{u}_{f_1}, \boldsymbol{u}_{f_2})^{-1}.$$

By computation, one can show that $\Delta_{\boldsymbol{x}} E_{ARAP}(M, \boldsymbol{x})$ is exactly the Laplacian matrix $L(M)$ of the mesh M. Hence the solution of the corresponding quadratic programming problem is

$$\boldsymbol{x}^* = \boldsymbol{x} - L(M)^{-1} \nabla_{\boldsymbol{x}} E_{ARAP}(M, \boldsymbol{x}).$$

Of course, to guarantee bijectivity, we should denote the descent direction by

$$\boldsymbol{d} = -L(M)^{-1} \nabla_{\boldsymbol{x}} E_{ARAP}(M, \boldsymbol{x}),$$

and then compute along this direction a step size that effectively decreases the energy without flipping any triangle; the details of this part will be discussed later.

Later, Kovalsky et al. pointed out in [21] that transforming the gradient of other distortion energies by the inverse of the Laplacian matrix also yields good descent directions, i.e., for an arbitrary distortion energy one takes in each iteration

$$\boldsymbol{d} = -L(M)^{-1} \nabla_{\boldsymbol{x}} E(M, \boldsymbol{x}),$$

which in essence amounts to locally approximating the objective function by the ARAP energy. This preconditioner is therefore called the **Sobolev preconditioner**, and the corresponding preconditioning method is called Sobolev gradient descent. It is worth noting that this preconditioner does not depend on the iterate, i.e., it stays fixed throughout the iterations, so it can be prefactorized, which greatly reduces the time required per iteration. Moreover, this preconditioner is independent of the objective function, so for general geometric energies it can be used in place of ordinary gradient descent for acceleration. Kovalsky et al. [21] further accelerated it with Nesterov's method and proposed their AQP method. The details of this part are explained in Section 3.

2. SLIM (Scalable Locally Injective Mappings) Preconditioner [32]

Although the ARAP distortion energy is simple, as mentioned in the introduction, it does not penalize nearly flipped triangles strongly enough, so the maximal step size along the descent directions it produces is often very small, and the iterations easily get stuck. For other energies, although the Sobolev preconditioner is universal, we would like to tailor a similar preconditioner to each energy so as to decrease the chosen target energy more effectively. Following this idea, Rabinovich et al. [32] proposed a new class of preconditioners, called SLIM preconditioners. Rabinovich et al. pointed out [32] that a general geometric energy can be locally approximated by a quadratic function better than the ARAP function. They achieve this by multiplying the ARAP distortion function by a weight matrix:

$$P_W(J_f) = \|W_f \, (J_f - R_f)\|_F^2, \qquad \text{s.t. } \nabla_{J_f} P_W(J_f) = \nabla_{J_f} D(J_f).$$

In this approximating quadratic function, we regard only J_f as a variable and treat W_f and R_f as constants. Computing the gradient of this approximating function, we obtain

$$\nabla_{J_f} P_W(J_f) = 2 W_f^T W_f \, (J_f - R_f) = \nabla_{J_f} D(J_f),$$
$$\Rightarrow W_f = \left(\frac{1}{2} \nabla_{J_f} D(J_f) \, (J_f - R_f)^{-1} \right)^{1/2}.$$

In particular, for the symmetric Dirichlet distortion function,

$$\nabla_{J_f} D_{symDir}(J_f) = U_f \operatorname{diag}\left(2\sigma_{1,f} - \frac{2}{\sigma_{1,f}^3},\ 2\sigma_{2,f} - \frac{2}{\sigma_{2,f}^3}\right) V_f^T,$$

$$\Rightarrow W_f = U_f \operatorname{diag}\left(\sqrt{\frac{(\sigma_{1,f}+1)(\sigma_{1,f}^2+1)}{\sigma_{1,f}^3}},\ \sqrt{\frac{(\sigma_{2,f}+1)(\sigma_{2,f}^2+1)}{\sigma_{2,f}^3}}\right) U_f^T.$$

Thus the SLIM local preconditioner matrix $H_{SLIM,f}$ is

$$H_{SLIM,f} = W_f^T W_f \otimes P_f P_f^T.$$

The global preconditioner matrix is

$$H_{SLIM} = \frac{1}{A(M)} \sum_{f \in F} |f|\, H_{SLIM,f},$$

so the descent direction at the iterate $\boldsymbol{x}$ is

$$\boldsymbol{d} = -H_{SLIM}^{-1} \nabla_{\boldsymbol{x}} E(M, \boldsymbol{x}).$$

This preconditioner performs much better than the Sobolev preconditioner, because its weight matrices are computed specifically for each distortion energy. In addition, although this preconditioner changes during the iterations, its pattern of nonzero entries does not, so this sparsity pattern can be prefactorized in advance. Rabinovich et al. also showed experimentally [32] that this preconditioner is **scalable**, i.e., for subdivisions of the same model at different resolutions, the number of iterations required for convergence is about the same.

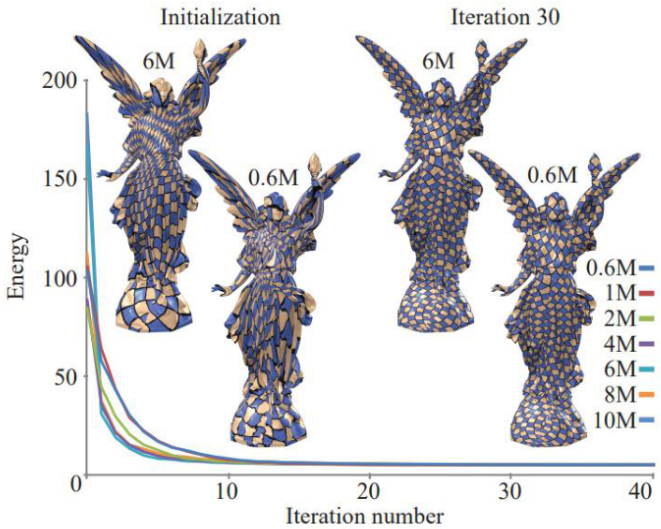


Figure 2.3: Scalability of the SLIM preconditioner [32]

3. AKVF (Approximated Killing Vector Field) Preconditioner [5]

As mentioned in the introduction, a preconditioner is in essence a Riemannian metric on the solution space. Thus, considering the solution space alone, one can also construct a preconditioner by designing a suitable Riemannian metric. Claici et al. observed in [5] that two parameterization maps differing by a global rigid transformation have the same distortion energy, and

should therefore be regarded as equivalent. A rigid transformation of the plane has three degrees of freedom—two translations and one rotation. Hence, after quotienting the original $2|V|$-dimensional Euclidean space by the equivalence relation induced by rigid transformations, the actual solution space should be a $(2|V|-3)$-dimensional manifold. Defining a Riemannian metric on this manifold means defining an inner product on the tangent space at each of its points. However, defining a Riemannian metric directly on such a manifold is difficult, and even if it were defined, computing descent directions afterwards would be cumbersome. Therefore, we can first define a pseudo-Riemannian metric (i.e., a positive semidefinite Riemannian metric) on the original $2|V|$-dimensional Euclidean space, and then obtain the Riemannian metric on the manifold as the one induced by the quotient map. This pseudo-Riemannian metric must have the three symmetry-preserving tangent directions as Killing vectors, i.e., they must be eigenvectors of the pseudo-Riemannian metric with eigenvalue 0. More concretely, a tangent direction $\boldsymbol{d}$ at a parameterization map $\boldsymbol{x}$ is in essence a tangent vector field on this planar embedded mesh, i.e., for each vertex $i \in V$, $\boldsymbol{d}_i$ is a tangent vector at $\boldsymbol{x}_i$. All symmetry-preserving tangent vector fields have the form

$$\boldsymbol{d}_i = \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix} \boldsymbol{x}_i + \boldsymbol{\xi}_i,$$

where the first term is the rotational part and the second is the translational part. A pseudo-Riemannian metric on the original $2|V|$-dimensional solution space is a symmetric positive semidefinite matrix $H(\boldsymbol{x})$ at each parameterization map $\boldsymbol{x}$. It induces a well-defined Riemannian metric on the reduced $(2|V|-3)$-dimensional manifold if and only if its null space consists exactly of the symmetry-preserving tangent vector fields. Claici et al. pointed out [5] that such tangent vector fields share a common feature: their Jacobians with respect to the parametric coordinates are all skew-symmetric matrices. However, knowing only the parametric coordinates and tangent vector at a single vertex does not uniquely determine this Jacobian, so it must be considered on triangles. For a triangle $f = (f_1, f_2, f_3) \in F$ of the mesh M, given the parameterization map $\boldsymbol{x}$ and the tangent vector field $\boldsymbol{d}$, the system

$$\boldsymbol{d}_{f_k} = A_f \, \boldsymbol{x}_{f_k} + \boldsymbol{\xi}_f, \qquad k = 1, 2, 3,$$

can be solved for a unique A_f:

$$A_f = (\boldsymbol{d}_{f_1} - \boldsymbol{d}_{f_0},\ \boldsymbol{d}_{f_2} - \boldsymbol{d}_{f_0}) \, (\boldsymbol{x}_{f_1} - \boldsymbol{x}_{f_0},\ \boldsymbol{x}_{f_2} - \boldsymbol{x}_{f_0})^{-1}.$$

Note that A_f is antisymmetric if and only if $A_f + A_f^T = 0$, which is equivalent to $\|A_f + A_f^T\|_F^2 = 0$. Moreover, $\|X + X^T\|_F^2$ is a seminorm (i.e., a positive semidefinite quadratic form) on the general linear space $\mathrm{GL}_2(\mathbb{R}) = \mathbb{R}^{2\times 2}$, and it can be extended to a semi-inner product:

$$\langle X, Y \rangle = \operatorname{tr}(X + X^T)(Y + Y^T).$$

In practice the tangent vector field $\boldsymbol{d}$ is unknown, so one computes the Jacobian separately for each unit tangent vector field and evaluates the pairwise semi-inner products between the 6 Jacobian matrices of each triangle. Filling these, in order, into a 6×6 matrix $H_{AKVF,f}(\boldsymbol{x})$ and weighting by area yields a pseudo-Riemannian metric $H_{AKVF}(\boldsymbol{x})$ on the original Euclidean space. Using this pseudo-Riemannian metric as the preconditioner gives the so-called AKVF preconditioner. The search direction of each iteration can then be computed by

$$\boldsymbol{d} = -H_{AKVF}(\boldsymbol{x})^{-1} \nabla_{\boldsymbol{x}} E(M, \boldsymbol{x}).$$

A distinctive feature of this preconditioner is that it depends neither on the distortion energy nor on the geometry of the original mesh, but only on the connectivity of the mesh; it therefore works well when the original mesh is close to a developable surface.

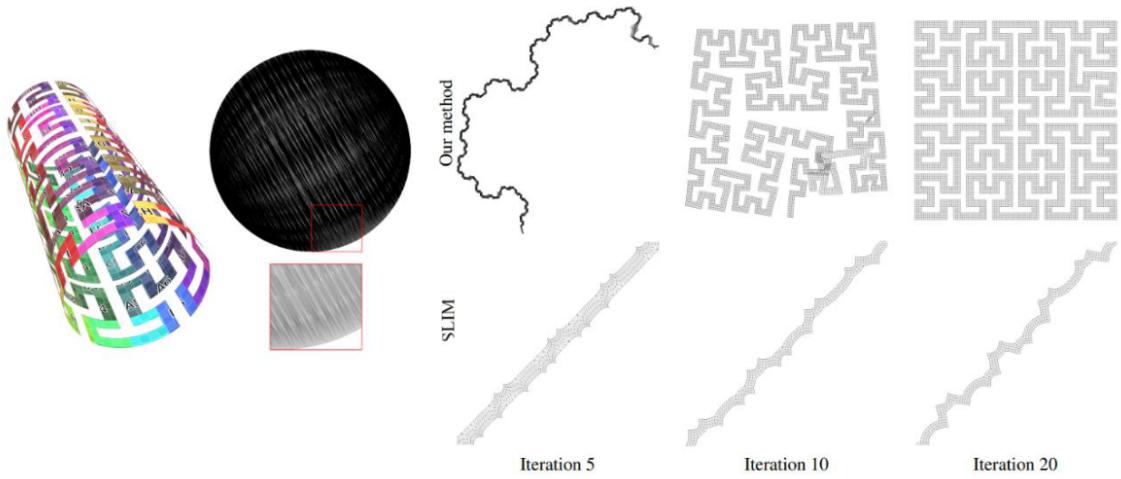


Figure 2.4: Comparison of the AKVF and SLIM preconditioners [5].

2.2.2 Second-order solvers

1. The CM (Composite Majorization) solver [41]

Since the objective function is nonconvex, a natural idea is to approximate it by a convex function. The preconditioners mentioned above essentially all approximate it by a positive semidefinite quadratic form and use only the first-order derivative information of the objective, so their final convergence rates are all linear. To achieve superlinear convergence, one must make full use of second-order derivative information. Shtengel et al. [41] pointed out that commonly used geometric energies can be split into the sum of a convex function and a concave function (i.e., one whose Hessian is negative definite), and the former can be used as an approximation of the objective. Here we describe in detail the convex approximation of the symmetric Dirichlet energy.

First, recall the singular-value representation of the symmetric Dirichlet energy:

$$E_{symDir}(M,\boldsymbol{x}) = \frac{1}{A(M)}\sum_{f\in F}\frac{1}{2}|f|\left(\|J_f\|_F^2 + \|J_f^{-1}\|_F^2\right) = \frac{1}{A(M)}\sum_{f\in F} h_f(\sigma_{1,f},\sigma_{2,f}),$$

where

$$J_f = (\boldsymbol{x}_{f_1}-\boldsymbol{x}_{f_0}, \boldsymbol{x}_{f_2}-\boldsymbol{x}_{f_0})\,(\boldsymbol{u}_{f_1},\boldsymbol{u}_{f_2})^{-1} = \begin{pmatrix} j_{00,f} & j_{01,f} \\ j_{10,f} & j_{11,f}\end{pmatrix} = U_f\,\mathrm{diag}(\sigma_{1,f},\sigma_{2,f})\,V_f^T,$$

$$h_f(u,v) = \frac{1}{2}|f|\left(u^2+v^2+\frac{1}{u^2}+\frac{1}{v^2}\right),$$

$$\sigma_{1,f} = \frac{1}{2}\left(\|\alpha_f\|+\|\beta_f\|\right), \qquad \sigma_{2,f} = \frac{1}{2}\left(\|\alpha_f\|-\|\beta_f\|\right),$$

$$\alpha_f = \begin{pmatrix} j_{00,f}+j_{11,f} \\ j_{10,f}-j_{01,f}\end{pmatrix}, \qquad \beta_f = \begin{pmatrix} j_{00,f}-j_{11,f} \\ j_{10,f}+j_{01,f}\end{pmatrix}.$$

Hence the Hessian of the symmetric Dirichlet energy is

$$\frac{\partial^2 E}{\partial \boldsymbol{x}^2} = \frac{1}{A(M)}\sum_{f\in F}\left(\left(\frac{\partial\sigma_{1,f}}{\partial\boldsymbol{x}},\frac{\partial\sigma_{2,f}}{\partial\boldsymbol{x}}\right)^T \Delta h_f(\sigma_{1,f},\sigma_{2,f})\left(\frac{\partial\sigma_{1,f}}{\partial\boldsymbol{x}},\frac{\partial\sigma_{2,f}}{\partial\boldsymbol{x}}\right) + \frac{\partial h_f}{\partial\sigma_{1,f}}\Delta\sigma_{1,f} + \frac{\partial h_f}{\partial\sigma_{2,f}}\Delta\sigma_{2,f}\right),$$

where

$$\Delta h_f(u,v) = |f| \operatorname{diag}\left(1 + \frac{3}{\sigma_{1,f}^4}, 1 + \frac{3}{\sigma_{2,f}^4}\right),$$
$$\frac{\partial h_f}{\partial \sigma_{1,f}} = |f|\left(\sigma_{1,f} - \frac{1}{\sigma_{1,f}^3}\right), \qquad \frac{\partial h_f}{\partial \sigma_{2,f}} = |f|\left(\sigma_{2,f} - \frac{1}{\sigma_{2,f}^3}\right),$$
$$\Delta\sigma_{1,f} = \frac{1}{2}\left(\Delta\|\alpha_f\| + \Delta\|\beta_f\|\right), \qquad \Delta\sigma_{2,f} = \frac{1}{2}\left(\Delta\|\alpha_f\| - \Delta\|\beta_f\|\right).$$

Since $h_f(u,v)$ is a convex function, the first term inside the sum of the Hessian is always positive semidefinite. Furthermore, $\|\alpha_f\|$ and $\|\beta_f\|$ are compositions of a linear map of $\boldsymbol{x}$ with a norm, and every norm is convex by the triangle inequality, so $\Delta\|\alpha_f\|$ and $\Delta\|\beta_f\|$ are also positive semidefinite. The last two terms inside the sum of the Hessian can be rewritten as

$$\frac{\partial h_f}{\partial \sigma_{1,f}}\Delta\sigma_{1,f} + \frac{\partial h_f}{\partial \sigma_{2,f}}\Delta\sigma_{2,f} = \frac{1}{2}\left(\frac{\partial h_f}{\partial \sigma_{1,f}} + \frac{\partial h_f}{\partial \sigma_{2,f}}\right)\Delta\|\alpha_f\| + \frac{1}{2}\left(\frac{\partial h_f}{\partial \sigma_{1,f}} - \frac{\partial h_f}{\partial \sigma_{2,f}}\right)\Delta\|\beta_f\|.$$

Since

$$\sigma_{1,f} > \sigma_{2,f} \;\Rightarrow\; \frac{\partial h_f}{\partial \sigma_{1,f}} = |f|\left(\sigma_{1,f} - \frac{1}{\sigma_{1,f}^3}\right) > |f|\left(\sigma_{2,f} - \frac{1}{\sigma_{2,f}^3}\right) = \frac{\partial h_f}{\partial \sigma_{2,f}},$$

the coefficient of $\Delta\|\beta_f\|$ is always positive. Therefore, if we take

$$H_{CM,symDir} = \frac{1}{A(M)}\sum_{f\in F}\left(\left(\frac{\partial\sigma_{1,f}}{\partial \boldsymbol{x}}, \frac{\partial\sigma_{2,f}}{\partial \boldsymbol{x}}\right)^T \Delta h_f(\sigma_{1,f},\sigma_{2,f})\left(\frac{\partial\sigma_{1,f}}{\partial \boldsymbol{x}}, \frac{\partial\sigma_{2,f}}{\partial \boldsymbol{x}}\right)\right.$$
$$\left. + \max\left(\frac{1}{2}\left(\frac{\partial h_f}{\partial \sigma_{1,f}} + \frac{\partial h_f}{\partial \sigma_{2,f}}\right), 0\right)\Delta\|\alpha_f\| + \frac{1}{2}\left(\frac{\partial h_f}{\partial \sigma_{1,f}} - \frac{\partial h_f}{\partial \sigma_{2,f}}\right)\Delta\|\beta_f\|\right),$$

then it is guaranteed to be positive semidefinite, and it is a good approximation of the Hessian of the original energy. Using it as the solver, i.e., computing the descent direction by

$$\boldsymbol{d} = -H_{CM,symDir}^{-1}\nabla_{\boldsymbol{x}} E_{symDir}(M,\boldsymbol{x}),$$

gives a second-order solver that can attain a superlinear convergence rate. The drawback of this method, however, is that it relies on the specific form of the target energy. Here we have only given a simple example with the symmetric Dirichlet energy; a general geometric energy may admit a convex–concave decomposition that is not obvious. Moreover, if the Hessian of the original energy is already positive semidefinite, this method does not necessarily preserve that matrix. In addition, it uses the analytic expression for the singular values of a 2×2 matrix, and such a simple expression does not exist for matrices of order three or higher, so the method is suited only to two-dimensional problems and not to higher-dimensional ones. The analytic projected Newton solver introduced next resolves these issues well.

2. The analytic projected Newton solver [42]

As mentioned in the introduction, a common second-order method for nonconvex optimization problems is projected Newton, which performs an eigendecomposition of the Hessian of the objective and simply sets its negative eigenvalues to 0. If no analytic expression for the eigendecomposition is known, it can only be computed numerically, which is almost a dead end for large-scale problems. Although all of our geometric energies can be written as weighted sums of local per-triangle energies—that is, one can apply projected Newton to each local energy separately and use the weighted sum of these positive semidefinite matrices as the solver—each iteration still takes a long time if numerical methods are used. Hence, if an analytic expression

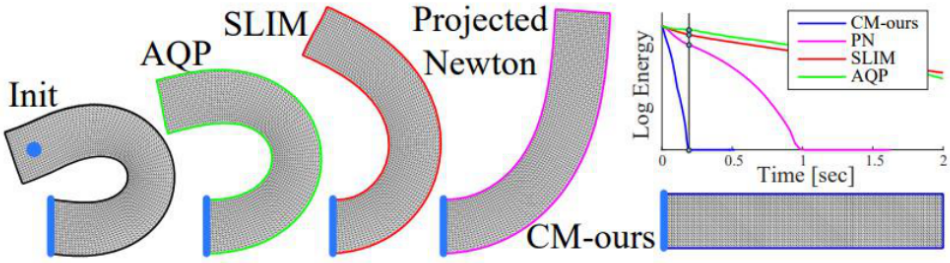


Figure 2.5: Comparison of CM with other solvers [41].

for the Hessian of the objective is available, the time per iteration can be greatly reduced. Smith et al. [42] pointed out that general isotropic geometric energies can locally be expressed as functions of the invariants of the stretch tensor of the deformation gradient, and these invariants are in turn functions of the position coordinates. Therefore, once the eigendecompositions of the Hessians of these invariants are understood, the eigendecomposition of each local energy is understood as well. More specifically, the polar decomposition of the deformation gradient of a triangle $f = (f_1, f_2, f_3) \in F$ of the mesh is

$$J_f = U_f \operatorname{diag}(\sigma_{1,f}, \sigma_{2,f}) V_f^T = \left(U_f V_f^T\right) \left(V_f \operatorname{diag}(\sigma_{1,f}, \sigma_{2,f}) V_f^T\right) = R_f S_f,$$

where S_f is called the stretch tensor, whose invariants are

$$I_f = \operatorname{tr}(S_f) = \sigma_{1,f} + \sigma_{2,f}, \qquad II_f = \det(S_f) = \sigma_{1,f}\sigma_{2,f}.$$

The local symmetric Dirichlet distortion of triangle f can be written as

$$D_{symDir}(J_f) = \frac{1}{2}\left(\sigma_{1,f}^2 + \sigma_{2,f}^2 + \frac{1}{\sigma_{1,f}^2} + \frac{1}{\sigma_{2,f}^2}\right) = \frac{1}{2}\left(I_f^2 - 2II_f\right)\left(1 + \frac{1}{II_f^2}\right),$$

and we can directly write down the eigendecomposition of its Hessian with respect to the deformation gradient [42]:

$$\Delta_{J_f} D_{symDir}(J_f) = \lambda_{1,f} \boldsymbol{d}_{1,f} \boldsymbol{d}_{1,f}^T + \lambda_2 \boldsymbol{d}_2 \boldsymbol{d}_2^T + \lambda_3 \boldsymbol{l}\boldsymbol{l}^T + \lambda_4 \boldsymbol{t}\boldsymbol{t}^T,$$

where

$$\begin{aligned}
\lambda_{1,f} &= 1 + \frac{3}{\sigma_{1,f}^4}, & \boldsymbol{d}_{1,f} &= \operatorname{vec}\left(U_f \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} V_f^T\right), \\
\lambda_{2,f} &= 1 + \frac{3}{\sigma_{2,f}^4}, & \boldsymbol{d}_{2,f} &= \operatorname{vec}\left(U_f \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} V_f^T\right), \\
\lambda_{3,f} &= 1 + \frac{1}{II_f^2} + \frac{I_f^2 - 2II_f}{II_f^3}, & \boldsymbol{l}_f &= \operatorname{vec}\left(\frac{1}{\sqrt{2}} U_f \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} V_f^T\right), \\
\lambda_{4,f} &= 1 + \frac{1}{II_f^2} - \frac{I_f^2 - 2II_f}{II_f^3}, & \boldsymbol{t}_f &= \operatorname{vec}\left(\frac{1}{\sqrt{2}} U_f \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} V_f^T\right),
\end{aligned}$$

where vec is any way of flattening a 2×2 matrix into a 4-dimensional vector, as long as the eigenvectors are flattened in the same way as the components of the deformation gradient. Note that the first three eigenvalues are always positive, and only the fourth eigenvalue can be negative. Hence the positive semidefinite part of the Hessian is

$$\left(\Delta_{J_f} D_{symDir}(J_f)\right)^+ = \lambda_{1,f} \boldsymbol{d}_{1,f} \boldsymbol{d}_{1,f}^T + \lambda_2 \boldsymbol{d}_2 \boldsymbol{d}_2^T + \lambda_3 \boldsymbol{l}\boldsymbol{l}^T + \max(\lambda_4, 0)\, \boldsymbol{t}\boldsymbol{t}^T.$$

Then, letting

$$H_{APN} = \frac{1}{A(M)} \sum_{f \in F} |f| \left(\Delta_{J_f} D_{symDir}(J_f)\right)^+,$$

we obtain the analytic projected Newton (APN) solver. Similarly, the local Hessians of many other geometric energies can also be eigendecomposed analytically in this way, and the weighted sum of their positive semidefinite parts can then be used as the solver. Note that this method relies on the invariants of the stretch tensor, so it generalizes easily to higher dimensions.

2.2.3 Line search [43]

With any of the solvers above we obtain a descent direction; the next step is to find a suitable step size along this descent direction. More specifically, we solve the following problem:

$$\alpha^* = \operatorname*{argmin}_{\alpha} E(\boldsymbol{x} + \alpha \boldsymbol{d}),$$
$$\text{s.t. } \det\left(\boldsymbol{x}_{f_1} - \boldsymbol{x}_{f_0} + \alpha\left(\boldsymbol{d}_{f_1} - \boldsymbol{d}_{f_0}\right), \boldsymbol{x}_{f_2} - \boldsymbol{x}_{f_0} + \alpha\left(\boldsymbol{d}_{f_2} - \boldsymbol{d}_{f_0}\right)\right) > 0.$$

The constraint ensures that no triangle is flipped after this iteration step; it is a system of quadratic inequalities in the step size α. Since the initialization is flip-free, we may assume that the parameterization is flip-free before every iteration. The set of all flip-free step sizes is an open subset of $\mathbb{R}$ and can therefore be written as a disjoint union of open intervals. What we care about is the upper bound of the open interval containing 0, which is called the global maximum step size of mesh M at position $\boldsymbol{x}$ along direction $\boldsymbol{d}$. This global maximum step size can be obtained as the minimum of the maximum step sizes of the individual triangles, and the maximum step size of each triangle is the smallest positive root of that triangle's degeneracy equation, i.e., the smallest positive root α_f [43] of

$$\det\left(\boldsymbol{x}_{f_1} - \boldsymbol{x}_{f_0} + \alpha\left(\boldsymbol{d}_{f_1} - \boldsymbol{d}_{f_0}\right), \boldsymbol{x}_{f_2} - \boldsymbol{x}_{f_0} + \alpha\left(\boldsymbol{d}_{f_2} - \boldsymbol{d}_{f_0}\right)\right) = 0.$$

Thus the global maximum step size is $\alpha_{max} = \min\{\alpha_f \mid f \in F\}$. As long as the actual step size lies within this maximum step size, no triangle can flip. Which step size to actually take is decided by a line search. We do not need the exact step size that decreases the energy the most, since that would require evaluating the energy repeatedly and is time-consuming; it suffices to find a step size that decreases the distortion energy sufficiently. Since the maximum step size is already available, we use a backtracking line search with the Armijo rule [43]: the step size is multiplied each time by a factor smaller than 1 until a suitable step size is found, with the Wolfe conditions used as the termination criterion. In this way we avoid repeated energy evaluations and quickly find a suitable step size within a few iterations. We can then update the parameter coordinates of the mesh by $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha \boldsymbol{d}$.

2.2.4 Termination criterion

Computing a descent direction with one of the solvers above, then finding a suitable step size along it and updating the parameter coordinates of the vertices, constitutes one complete iteration. Deciding when to terminate this iterative process, however, is not easy. We do not want too many iterations, but neither do we want the iteration to stop before it has converged. We would like a single termination criterion that works for different energy functions and meshes of different sizes, so that we need not tune parameters specifically for each case. Iterative optimization algorithms in general commonly use the following termination criteria:

1) based on the gradient of the objective: $\|\nabla_{\boldsymbol{x}} E(M, \boldsymbol{x})\| < \epsilon$;

2) based on the change of the iterate: $\|\Delta \boldsymbol{x}\| < \epsilon$ or $\|\Delta \boldsymbol{x}\| / \|\boldsymbol{x}\| < \epsilon$;

3) based on the change of the objective: $|\Delta E| < \epsilon$ or $|\Delta E|/|E| < \epsilon$.

Since our geometric energies are always positive and bounded below, we choose to terminate when the relative decrease of the energy is sufficiently small. This termination criterion is relatively fair across different energy functions and meshes of different sizes. Zhu et al. [51] studied termination criteria for geometric optimization methods in more detail.

Finally, we put all of the above steps together into a complete iterative algorithm:

Algorithm 1 Iterative geometric optimization algorithm for planar parameterization [41]

$\boldsymbol{x} = \text{Tutte}(\,)$
while $\Delta E/E > \epsilon$ **do**
  $\boldsymbol{d} = -H^{-1}\nabla_{\boldsymbol{x}} E(M, \boldsymbol{x})$
  $\alpha_{max} = \text{max_step}(\boldsymbol{x}, \boldsymbol{d})$
  $\alpha = \text{backtracking_line_search}(\boldsymbol{x}, \boldsymbol{d}, \alpha_{max})$
  $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha \boldsymbol{d}$
end while

This framework applies to all kinds of geometric energies and solvers, where H depends on the choice of solver.

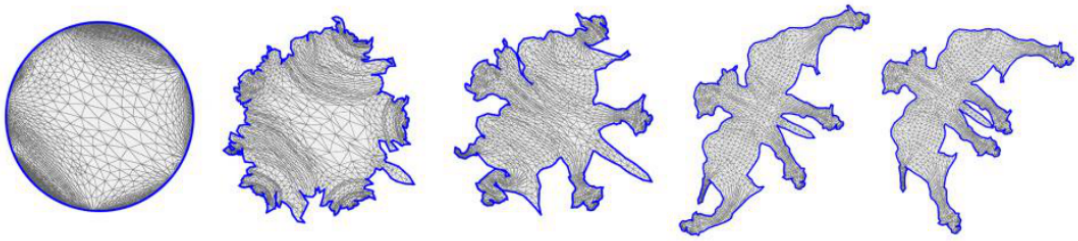


Figure 2.6: The geometric optimization process for planar parameterization [43].

2.3 A brief introduction to acceleration methods

The previous section introduced various iterative solvers and concluded with a unified iterative framework for geometric optimization. In this section we briefly introduce some special acceleration methods: changing the reference triangles, changing the search direction, Nesterov acceleration, and Anderson acceleration. The first two are geometric acceleration methods that exploit the geometric characteristics of the parameterization problem; the latter two are more general acceleration methods applicable to general nonlinear optimization problems.

2.3.1 Changing the reference triangles [23]

For meshes of complex shape, the initialization often contains triangles with extremely large local distortion. These triangles often make the maximum step size along the computed descent direction extremely small, so that the iteration cannot proceed smoothly. Liu et al. [23] observed that if the local distortion of every triangle is within a certain range, the iteration can proceed smoothly. They therefore proposed interpolating some intermediate triangles to replace the source triangles as reference triangles, so that the triangles whose distortion is large when the source triangles are used as reference have smaller distortion after the change of reference; the optimization of the distortion energy from the image triangles to the new reference triangles then becomes easier. Then, letting these reference triangles gradually converge to the source triangles

during the iteration, one can eventually still use the source triangles as reference and successfully optimize the distortion energy. In other words, this method shows the highly distorted triangles a path, so that triangles that could not reach their goal in one step can follow this path and reduce their distortion step by step, instead of getting stuck at the beginning because of an unsuitable direction. For details see [23].

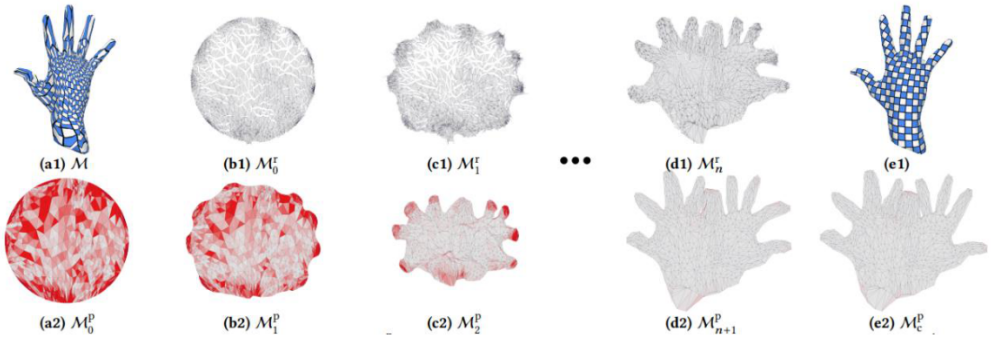


Figure 2.7: Reference triangles gradually converging to the original mesh [23].

2.3.2 Changing the search direction [51]

For search directions whose maximum step size is too small, Zhu et al. [51] proposed another remedy: find a search direction that is as close as possible to the computed descent direction while having as large a maximum step size as possible. They formulated this as a new optimization problem and used the damped projected Jacobi method to obtain an approximate solution within a fixed number of iterations, which serves as the adjusted search direction. They call this strategy Curing Line Search [51].

2.3.3 Nesterov acceleration [25][21]

From a physical point of view, the objective function of an optimization problem can be regarded as a potential energy, and its solution is then the steady state corresponding to the lowest energy. The essence of an iterative process is thus the motion of a point in the solution space under the force generated by the potential. If the resistance is sufficiently large, the point can only move a small step at a time in the direction of the force (i.e., the negative gradient of the objective); this is gradient descent. If the metric of the solution space is no longer Euclidean but some special Riemannian metric, the direction of the force becomes the negative gradient of the objective on the Riemannian manifold; this is the preconditioner approach. Now, if the resistance is reduced so that the velocity of the point can accumulate, its motion becomes smoother; this is the core idea of Nesterov acceleration [25]. This acceleration method introduces a momentum term that accumulates the gradient of the objective, and the direction of motion of the point is then determined by the momentum. Kovalsky et al. [21] used exactly this method to accelerate Sobolev gradient descent, obtaining their AQP (Accelerated Quadratic Proxy) method.

2.3.4 Anderson acceleration [2][28]

Anderson [2] viewed an iterative process as a function repeatedly applied to the iterate, i.e.,

$$\boldsymbol{x}^{(n+1)} = f\left(\boldsymbol{x}^{(n)}\right), \qquad n = 1, 2, \cdots.$$

The iterative process therefore eventually converges to a fixed point of this function. Hence one can use several consecutive iterates of the process to estimate this fixed point and take the

estimate as the next iterate, which is often more effective than iterating by applying the function directly. Since this method samples information about the function during the iteration, it is unstable when the iteration has just started (because the first few iterates differ too much for the function to be estimated well), and works better when the iterates are close to convergence. Peng et al. [28] modified Anderson acceleration and introduced it into geometric optimization, showing through experiments [28] that this acceleration method effectively speeds up the convergence of the SLIM preconditioner.

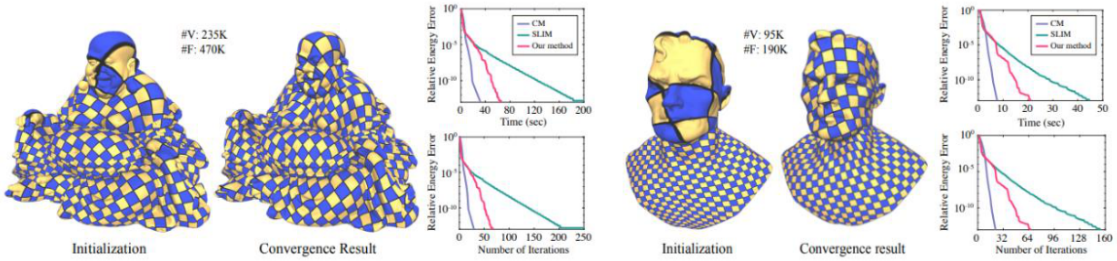


Figure 2.8: The SLIM method with Anderson acceleration [28].

Chapter 3

Low-Distortion Flip-Free Spherical Parameterization

3.1 Introduction to Spherical Parameterization

In practical applications, many meshes come from 3D scans of object surfaces and are therefore closed. Rather than being cut open and then flattened onto the plane, in many applications these meshes are better suited to being unfolded directly onto the sphere. However, because of the spherical constraint, computing a spherical parameterization map is considerably more complicated than computing a planar parameterization. The previous chapter described the complete pipeline of geometric optimization methods for planar parameterization, but generalizing these methods to the sphere is by no means trivial. First of all, a flip-free map from a closed mesh to the sphere is not easy to obtain. When it comes to maps from a surface to the sphere, the first thing that comes to mind is surely the Gauss map, which sends each point to its normal vector; for a general mesh, however, this is obviously not a bijection. Another easily conceived approach is to translate the mesh and then normalize each of its points, but this method yields a bijection only when the region enclosed by the mesh is star-shaped. Since a bijective parameterization in the plane can easily be obtained by linear methods, a natural idea is to first map the original mesh to the plane and then map it back onto the sphere by some plane-to-sphere map. This is spherical parameterization based on stereographic projection.

3.1.1 Spherical Parameterization Based on Stereographic Projection

Since stereographic projection is a bijection in the smooth setting and is moreover angle-preserving, it is natural to use this map to transfer planar parameterization coordinates to the sphere. Haker et al. [17] proposed removing one triangle from the original mesh, flattening the remainder onto the plane with a Tutte-like method, and then projecting the planar parameter coordinates back onto the sphere by stereographic projection, thereby obtaining a spherical map. Isenberg et al. [20] split the original mesh into two pieces, parameterized each into the unit disk with Tutte's method, and then used stereographic projection to map the two disks onto the upper and lower hemispheres of the unit sphere, respectively. However, Sheffer et al. [39] pointed out that these methods cannot guarantee a flip-free spherical parameterization, because stereographic projection in the discrete setting may introduce flips. They illustrated this with the following figure:

In the figure, the geodesic between the images on the sphere of two points A, B in the plane under stereographic projection is a great-circle arc on the sphere, and when this great-circle arc is mapped back to the plane by stereographic projection it becomes a circular arc passing through A and B. Consider a point C in the plane lying between the straight line connecting A, B and the above circular arc. When this point crosses the above arc, the planar triangle

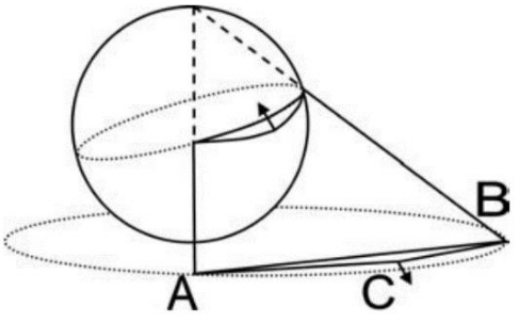


Figure 3.1: Discrete stereographic projection is no longer bijective [39]

ABC does not flip, whereas the corresponding spherical triangle does. In particular, when C lies exactly on the above arc, the planar triangle is non-degenerate, but the corresponding spherical triangle degenerates into a single great-circle arc. This example nicely illustrates that stereographic projection in the discrete setting can introduce flips.

This phenomenon occurs because stereographic projection cannot establish a correspondence between the geodesics of the plane and those of the sphere: if the geodesic between the images of the above points A, B on the sphere were mapped back to the plane precisely onto the straight line between A and B, the flip described above would not occur. Therefore, using stereographic projection to transfer planar parameterization coordinates to the sphere is inappropriate. Stereographic projection is bijective and has good properties in the smooth setting, but in the discrete setting it introduces flips. This is because in the smooth setting the map is a point-to-point map, whereas in the discrete setting the map is a triangle-to-triangle map, so besides the correspondence between points we must also take into account the correspondence between edges. Since the triangles we refer to are all geodesic triangles, i.e., triangles with straight-line-segment edges in the plane and triangles with great-circle-arc edges on the sphere, we need a map that establishes a correspondence between straight lines in the plane and great-circle arcs on the sphere in order to transfer parameterization coordinates from the plane to the sphere.

3.1.2 Spherical Parameterization Based on Gnomonic Projection

A very useful characterization of great-circle arcs on the sphere is as the intersection of the sphere with a plane in space passing through the origin. The geodesic between two points on the sphere is the intersection of the sphere with the plane formed by these two points and the origin. Hence, for a triangle in space not containing the origin, the intersection of the cone it generates with the sphere is a spherical triangle. This process is equivalent to normalizing every point of this spatial triangle, and is therefore called gnomonic (central) projection. From the above it is easy to see that, if we have a convex polyhedron containing the origin whose faces consist of triangles, then mapping its faces onto the sphere by gnomonic projection yields a flip-free map from its surface to the sphere. Thus the problem of finding a flip-free map from an arbitrary closed mesh to the sphere becomes that of first finding a flip-free map from the mesh to the surface of some convex polyhedron, and then gnomonically projecting the surface of this convex polyhedron onto the sphere.

And a flip-free map from a closed mesh to the surface of a convex polyhedron is easy to obtain: we can split the closed mesh into 4 pieces such that each piece is connected to each of the other 3 pieces, and any two pieces share exactly one continuous boundary. We can use **hierarchical clustering** [34, 36, 37] or the Voronoi diagram on the mesh [7] to achieve such a partition. In addition, we require that no triangle has all three of its vertices on the same

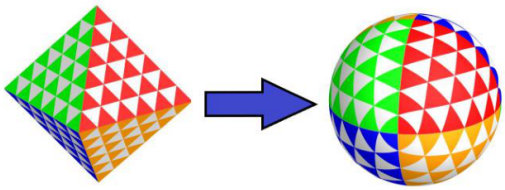


Figure 3.2: Gnomonic projection of a convex polyhedron [31]

boundary, which can be satisfied by straightening the boundaries with shortest paths [34]. Then, we use Tutte's method [46] to map the vertices of each piece onto one face of a tetrahedron, and afterwards use stereographic projection to map these four faces onto the sphere, obtaining a flip-free spherical parameterization. The parameterization obtained in this way is easy to compute and its bijectivity is theoretically guaranteed, but its drawback is that the resulting spherical map has extremely large distortion; moreover, for large meshes, because each of the pieces contains too many vertices, flips caused by numerical precision easily occur when applying Tutte's method.

3.1.3 Spherical Parameterization Based on Conformal Geometry

Since the theory of smooth surfaces is already very well developed, some early work on spherical parameterization [14, 15, 16] started from the theory of conformal geometry, turning the problem of computing a spherical conformal parameterization into a differential equation, which was then solved by Gauss–Seidel iteration initialized with the Gauss map. However, the bijectivity of maps in the discrete setting is completely different from that in the continuous setting. Therefore, even if the solution of the above differential equation is a bijective conformal parameterization map in the continuous setting, flips may still occur in the discrete setting. Furthermore, [33] later proved that this iterative process becomes unstable as it approaches a bijective solution, and then converges to a degenerate solution.

3.1.4 The Colin de Verdière Equations

An important reason why planar Tutte parameterization guarantees bijectivity is that the parameter coordinates of each vertex can be expressed as a convex combination of the parameter coordinates of its surrounding vertices. With deeper research into spherical parameterization, some results analogous to the planar case have also been proved. Gotsman et al. [12] introduced the Colin de Verdière theory from graph theory and proved that a discrete spherical parameterization is flip-free if and only if every vertex can be expressed as the gnomonic projection of a convex combination of the parameterizations of its surrounding vertices, as shown in the figure:

They also gave a general system of equations for spherical parameterization [12]:

$$\sum_{(i,j)\in E} \omega_{ij}\boldsymbol{x}_j = \alpha_i \boldsymbol{x}_i, \qquad \forall i \in V$$

$$\|\boldsymbol{x}_i\| = 1, \qquad \forall i \in V$$

This system of equations is analogous to the Tutte equations of planar parameterization and is the most fundamental characterization of discrete spherical parameterization; it is called the Colin de Verdière equations. However, since it is a nonlinear system, solving it directly is extremely difficult. They tried to solve these equations with some numerical tools, but even for fairly small meshes this took a very long time. Later, Saba et al. [33] attempted to propose a more efficient solution method, but they could not theoretically guarantee a flip-free parameterization result. Although the Colin de Verdière equations are the most fundamental

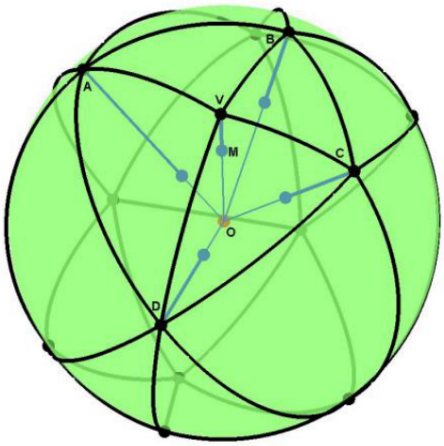


Figure 3.3: Vertex v is the gnomonic projection of a convex combination of its surrounding vertices A, B, C, D [12]

equations of spherical parameterization, to this day there is still no efficient and robust solver for them.

3.1.5 Spherical Parameterization Based on Multiresolution

Since it is very difficult to design a spherical parameterization with a theoretical flip-free guarantee by directly using the geometric information of the original mesh, one may as well start from a topological point of view. Topologically, if a spherical map of a mesh is such that the image of every vertex lies inside the spherical region enclosed by the images of its neighboring vertices, then this map is flip-free. Therefore, as long as we find an order in which to fill in the vertices of the original mesh and fill them onto the sphere in this order, we obtain a flip-free spherical map; this technique is called the multiresolution technique. Shapiro and Tal [38] were the first to apply this technique and gave a theoretical proof. Their method consists of two stages, mesh simplification and mesh refinement. In the mesh simplification stage, they remove one vertex from the original mesh at a time, record the information of this vertex, retriangulate its neighborhood, and repeat these steps until the original mesh is simplified to a tetrahedron. In the mesh refinement stage, they fill the vertices back onto the sphere one by one in the reverse order of their removal, until all vertices have been restored. Praun and Hoppe [31] also used this technique and attempted to optimize a distortion energy during mesh refinement. Later, [47, 19] further continued the study of this technique. The advantages of this technique are that it theoretically guarantees flip-freeness and that in practice it is more tolerant of numerical precision than the gnomonic projection method; its drawback is that it is time-consuming and unsuitable for especially large meshes.

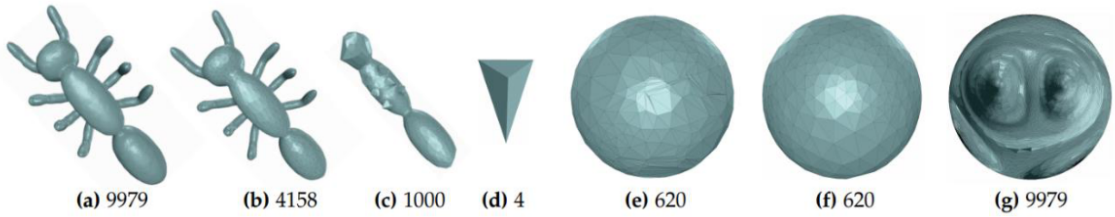


Figure 3.4: Pipeline of the multiresolution technique [19]

3.1.6 Geometric Optimization for Spherical Parameterization

Geometric optimization methods for the planar parameterization problem have been studied extensively, yet the literature on geometric optimization methods for the spherical parameterization problem is very scarce. The only works we know of are [49], [48] and [19]. Among them, [49] generalized the planar ARAP distortion energy to the sphere and optimized it. However, sharing the shortcoming of the planar ARAP method, the spherical ARAP energy is completely unable to prevent spherical triangles from flipping. [48] generalized the planar AMIPS energy to the sphere and, assuming a flip-free spherical map is already available as initialization, optimized the spherical AMIPS energy on this basis. The optimization strategy they used is **block coordinate descent**, a generalization of the single-vertex relaxation method, and the solver they used is ordinary gradient descent, since at the time there were not yet good solvers for geometric optimization. Although their method guarantees that no flips occur during the iterations, it easily gets stuck at places with high distortion. This is because their energy is defined using piecewise linear maps, so what it handles are still planar triangles rather than spherical triangles; and since the degeneration behavior of a spherical triangle is not consistent with that of the corresponding planar triangle, an energy defined in this way cannot effectively prevent spherical triangles that are close to flipping. In addition, they handled the spherical constraint only in a simple way, and they did not compute a maximal step size during the line search; instead, they first set a sufficiently large step size and then repeatedly shrank it by a given ratio until obtaining a step size with no spherical triangle flips and a sufficient decrease of the distortion energy. [19] used the multiresolution method, optimizing a distortion energy while filling the vertices onto the sphere one by one. Their geometric optimization is vertex-by-vertex rather than over the whole mesh, and is therefore also time-consuming. It can thus be seen that, to date, there is no spherical parameterization algorithm that makes full use of the geometric optimization methods commonly used in the plane. The main contribution of this thesis is precisely this work.

3.2 Spherical Triangle Meshes

3.2.1 The Flip-Free Condition for Spherical Triangles

Since a spherical triangle refers to a geodesic triangle on the sphere, a spherical triangle must pass through a degenerate stage when it flips, namely when the spherical triangle degenerates into a single great-circle arc on the sphere. Clearly, the original degeneracy criterion for planar triangles no longer applies, because at this moment the three spherical vertices lie on one great-circle arc, so the corresponding planar triangle is non-degenerate most of the time, unless two or more of its vertices happen to coincide. According to the characterization of great-circle arcs mentioned in the previous section, if three spherical vertices lie on one great-circle arc, then these three vertices must be coplanar with the origin. Conversely, if three points on the sphere are not coplanar with the origin, then these three points cannot lie on one great-circle arc, and hence the corresponding spherical triangle is also non-degenerate. Thus, we can use the signed volume of the tetrahedron formed by the three spherical vertices and the origin to determine whether a spherical triangle is flipped or degenerate: let the spherical parameter coordinates of the vertices of a triangle $f = (f_0, f_1, f_2) \in F$ in the mesh M be $\boldsymbol{x}_{f_0}, \boldsymbol{x}_{f_1}, \boldsymbol{x}_{f_2} \in S^2$, respectively; then this triangle is not flipped on the sphere if and only if

$$\det\left(\boldsymbol{x}_{f_0}, \boldsymbol{x}_{f_1}, \boldsymbol{x}_{f_2}\right) > 0.$$

Compared with the planar case, we find that the essence of a spherical triangle is no longer a triangle, but the cone formed by the three spherical vertices and the origin. The essence of a spherical triangle is three-dimensional rather than two-dimensional, so when defining the

related parameterization problems one should think more from a three-dimensional perspective, rather than simply treating it as a piecewise linear map between triangle meshes.

3.2.2 Rigorous Spherical Parameterization Maps

To define the spherical parameterization problem well, we must first clarify the nature of a map from a mesh to the sphere. The results of many spherical parameterization methods are not rigorous spherical maps, but piecewise linear maps from the mesh to a convex polyhedron inscribed in the sphere—they still map edges to straight edges and triangles to planar triangles, only with the vertices of these edges and triangles constrained to lie on the sphere. However, for a rigorous spherical map, not only must the vertices of the mesh be mapped onto the sphere, but the edges of the mesh must also be mapped to great-circle arcs on the sphere, and the triangles of the mesh must be mapped to spherical triangles. Given the images on the sphere of the three vertices of a triangle, a spherical triangle bounded by great-circle arcs is uniquely determined, but the map from the interior of the triangle of the original mesh to the interior of this spherical triangle is not unique. Any map from a planar triangle to the corresponding spherical triangle is called a **spherical triangle map**. Praun and Hoppe [31] studied such maps thoroughly, listing 8 spherical triangle maps and analyzing the properties of each:

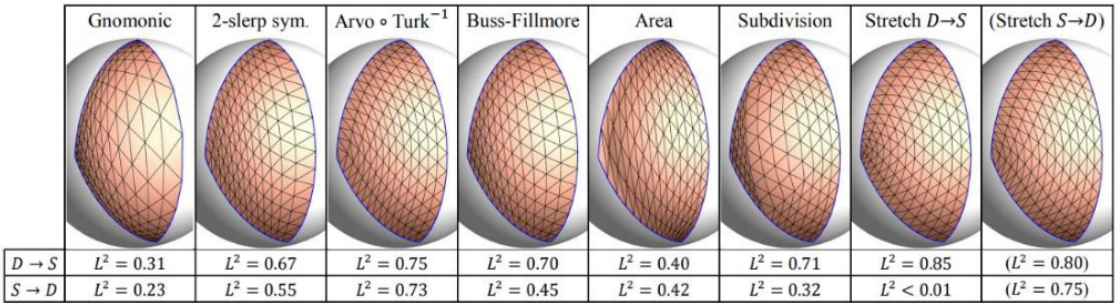


Figure 3.5: Comparison of the distortion of various spherical triangle maps [31]

Triangle map	Boundaries		Smooth	Analytic		Stretch	
	C^0	arc-len.	interior	$D \to S$	$S \to D$	$D \to S$	$S \to D$
Gnomonic	yes	**no**	yes	yes	yes	poor	poor
2-slerp-sym.	yes	yes	yes	yes	**no**	ok	ok
Arvo ∘ Turk^{-1}	yes	**no**	yes	yes	**no**	good	good
Buss-Fillmore	yes	yes	yes	**no**	yes	good	poor
Area	**no**	**no**	yes	yes	yes	poor	poor
Subdivision	yes	yes	**no**	**no**	**no**	good	poor
Stretch optim.	**no**	**no**	visually	**no**	**no**	best	best

Figure 3.6: Comparison of the properties of various spherical triangle maps [31]

The simplest of these is called the **gnomonic map**, which is essentially a linear map between planar triangles composed with gnomonic projection: given a triangle $f = (f_0, f_1, f_2) \in F$ in the mesh M, let the positions of the three vertices of its source triangle be $\boldsymbol{u}_{f_0}, \boldsymbol{u}_{f_1}, \boldsymbol{u}_{f_2} \in \mathbb{R}^2$, and let the images of these three vertices on the sphere be $\boldsymbol{x}_{f_0}, \boldsymbol{x}_{f_1}, \boldsymbol{x}_{f_2} \in S^2$; then the gnomonic map from the points in the interior of this triangle to the points in the interior of the corresponding spherical triangle is

$$\lambda_0 \boldsymbol{u}_{f_0} + \lambda_1 \boldsymbol{u}_{f_1} + \lambda_2 \boldsymbol{u}_{f_2} \mapsto \frac{\lambda_0 \boldsymbol{x}_{f_0} + \lambda_1 \boldsymbol{x}_{f_1} + \lambda_2 \boldsymbol{x}_{f_2}}{\|\lambda_0 \boldsymbol{x}_{f_0} + \lambda_1 \boldsymbol{x}_{f_1} + \lambda_2 \boldsymbol{x}_{f_2}\|},$$

where $\lambda_0, \lambda_1, \lambda_2 \geq 0$, $\lambda_0+\lambda_1+\lambda_2 = 1$ are barycentric coordinates in the triangle. The advantages of this map are that it is simple, its inverse can be expressed explicitly, and it also preserves the correspondence between geodesics in the interior of the triangle. Some other spherical triangle maps are better than the gnomonic map in terms of smoothness across boundaries, but these maps are generally rather complicated and cannot guarantee the correspondence between geodesics within the triangle. Therefore, we choose the gnomonic map as our spherical triangle map. Thus, given the parameter positions on the sphere of all vertices of the mesh, we define the corresponding spherical parameterization as the **piecewise gnomonic map** that maps each vertex to its spherical parameter coordinates and maps each triangle of the mesh, via the gnomonic map, to the corresponding spherical triangle. Only with this definition can a spherical map in the true sense be given rigorously. According to this definition, as long as the spherical parameter coordinates of all vertices are known, a spherical parameterization map is uniquely determined.

3.2.3 Distortion Energies of Spherical Parameterization Maps

Having a rigorous spherical parameterization map, we now define the distortion energy of this map. Although we could directly imitate the way planar parameterization distortion energies are defined and define it as the corresponding distortion energy of the piecewise linear map between triangle meshes, we find that a distortion energy defined this way is hard to optimize. This is because when a spherical triangle approaches degeneracy, the corresponding planar triangle does not degenerate; and when a spherical triangle flips, the corresponding planar triangle does not flip either. Therefore, an energy defined in this way cannot effectively prevent spherical triangles from approaching degeneracy.

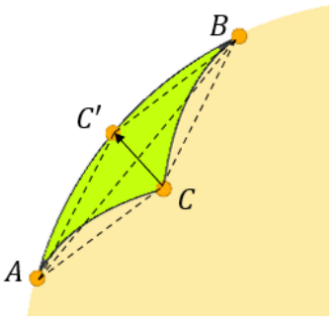


Figure 3.7: Degeneration of a spherical triangle

One alternative worth considering is to use the continuous form of the planar energy. Taking the symmetric Dirichlet energy as an example, its planar discrete form is

$$E_{symDir}(M, \boldsymbol{x}) = \frac{1}{A(M)} \sum_{f \in F} \frac{1}{2}|f| \left(\|J_f\|_F^2 + \left\|J_f^{-1}\right\|_F^2 \right).$$

The corresponding continuous form should be written locally as an integral:

$$E_{symDir}(M, \boldsymbol{x}) = \frac{1}{A(M)} \sum_{f \in F} \int_f \frac{1}{2} \left(\|J_f(\lambda_0, \lambda_1, \lambda_2)\|_F^2 + \left\|J_f^{-1}(\lambda_0, \lambda_1, \lambda_2)\right\|_F^2 \right) d|f|,$$

where $(\lambda_0, \lambda_1, \lambda_2)$ are the barycentric coordinates of a point in triangle f. These two forms are consistent, because a parameterization map in the plane is a piecewise linear map whose deformation gradient is constant on each triangle, so the corresponding integral form evaluates to exactly the discrete-sum form. However, as stated above, a spherical parameterization map

is no longer a piecewise linear map but a piecewise gnomonic map, so one should no longer use the first, discrete-sum form, but rather the second, sum-of-piecewise-integrals form. In this form, the deformation gradient is not constant within a triangle, so the local distortion function cannot be defined directly on the triangle; instead, it must be defined at every point within the triangle and then integrated over the triangle. This definition rigorously reflects the information of the spherical parameterization and is general: all geometric energies can be treated similarly. However, the integral in this definition cannot be evaluated explicitly for most energies, and in a practical algorithm it can only be computed accurately by numerical methods. This is very time-consuming; especially in geometric optimization, where besides the energy itself one must also compute the gradient, the Hessian, and so on, the time required for accurate computation would be prohibitive.

We can try to estimate this integral energy. Given the spherical parameter coordinates $\boldsymbol{x} = \{\boldsymbol{x}_v \in S^2 \subseteq \mathbb{R}^3\}_{v\in V}$ of the vertices of the mesh M, for a triangle $f = (f_0, f_1, f_2) \in F$, the gnomonic map from the corresponding source triangle to the spherical triangle is

$$\lambda_0 \boldsymbol{u}_{f_0} + \lambda_1 \boldsymbol{u}_{f_1} + \lambda_2 \boldsymbol{u}_{f_2} \mapsto \frac{\lambda_0 \boldsymbol{x}_{f_0} + \lambda_1 \boldsymbol{x}_{f_1} + \lambda_2 \boldsymbol{x}_{f_2}}{\|\lambda_0 \boldsymbol{x}_{f_0} + \lambda_1 \boldsymbol{x}_{f_1} + \lambda_2 \boldsymbol{x}_{f_2}\|},$$

where $\lambda_0, \lambda_1, \lambda_2 \geq 0$, $\lambda_0 + \lambda_1 + \lambda_2 = 1$ are barycentric coordinates in the triangle. We write

$$\boldsymbol{p}(\lambda_0, \lambda_1, \lambda_2) = \lambda_0 \boldsymbol{x}_{f_0} + \lambda_1 \boldsymbol{x}_{f_1} + \lambda_2 \boldsymbol{x}_{f_2},$$

so that the deformation gradient of the gnomonic map at $(\lambda_0, \lambda_1, \lambda_2)$ is

$$J_f(\lambda_0, \lambda_1, \lambda_2) = \frac{1}{\|\boldsymbol{p}\|}\left(I - \frac{\boldsymbol{p}\boldsymbol{p}^T}{\|\boldsymbol{p}\|^2}\right)(\boldsymbol{x}_{f_1} - \boldsymbol{x}_{f_0}, \boldsymbol{x}_{f_2} - \boldsymbol{x}_{f_0})\,(\boldsymbol{u}_{f_1} - \boldsymbol{u}_{f_0}, \boldsymbol{u}_{f_2} - \boldsymbol{u}_{f_0})^{-1}.$$

The deformation gradient here is no longer a 2×2 matrix but a 3×2 matrix, so its inverse is not well defined. Nevertheless, we can use its stretch tensor instead:

$$S_f(\lambda_0, \lambda_1, \lambda_2) = \left(J_f^T(\lambda_0, \lambda_1, \lambda_2) J_f(\lambda_0, \lambda_1, \lambda_2)\right)^{1/2}.$$

The local integral energy at triangle f can then be rewritten as

$$\int_f \frac{1}{2}\,\|S_f(\lambda_0, \lambda_1, \lambda_2)\|_F^2 \left(1 + \frac{1}{\det\left(S_f(\lambda_0, \lambda_1, \lambda_2)\right)^2}\right) d|f|.$$

Computing the determinant of the stretch tensor, we find

$$\det S_f(\lambda_0, \lambda_1, \lambda_2) = \frac{\det\left(\boldsymbol{x}_{f_0}, \boldsymbol{x}_{f_1}, \boldsymbol{x}_{f_2}\right)}{\|\boldsymbol{p}\|^3 \det\left(\boldsymbol{u}_{f_1} - \boldsymbol{u}_{f_0}, \boldsymbol{u}_{f_2} - \boldsymbol{u}_{f_0}\right)},$$

which explains precisely the fundamental reason why the integral-form distortion energy can effectively prevent spherical triangles from flipping: the integral energy contains a term reciprocal to the stretch tensor, and the determinant of the stretch tensor is proportional to the volume of the tetrahedron formed by the three vertices of the spherical triangle and the origin. Hence, as a spherical triangle approaches degeneracy, this volume tends to 0, and the integral energy therefore tends to $+\infty$. Thus, any energy containing a term reciprocal to this tetrahedron volume can effectively prevent spherical triangles from flipping.

Since the integral-form distortion energy is rather complicated to compute, we may as well approach the problem from another angle. Note that the degeneracy condition for spherical triangles is stated in terms of tetrahedron volume, so we may as well start from the perspective of tetrahedra. For a triangle in the original mesh, instead of flattening it onto the plane, we associate it with a tetrahedron formed by three vertices on the sphere and the origin [49], such

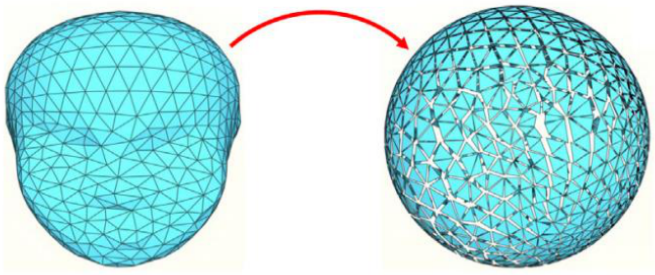


Figure 3.8: Associating triangles with source tetrahedra [49]

that the triangular face formed by the three points on the sphere is congruent to this triangle of the mesh and has the same orientation. We call such a tetrahedron the **source tetrahedron**. For a triangle $f = (f_0, f_1, f_2) \in F$, let the three spherical vertices of its source tetrahedron be $\boldsymbol{u}_{f_0}, \boldsymbol{u}_{f_1}, \boldsymbol{u}_{f_2}$; then the deformation gradient from it to the image tetrahedron is

$$J_{f,tet} = (\boldsymbol{x}_{f_0}, \boldsymbol{x}_{f_1}, \boldsymbol{x}_{f_2}) (\boldsymbol{u}_{f_0}, \boldsymbol{u}_{f_1}, \boldsymbol{u}_{f_2})^{-1},$$

which is a 3×3 matrix. We can then define the symmetric Dirichlet energy of a spherical parameterization as

$$E^{sphere}_{symDir}(M, \boldsymbol{x}) = \frac{1}{A(M)} \sum_{f \in F} \frac{1}{2} |f| \left(\|J_{f,tet}\|_F^2 + \left\| J_{f,tet}^{-1} \right\|_F^2 \right).$$

This energy can effectively prevent spherical triangles from flipping, because it contains a term reciprocal to the determinant of the deformation gradient, and the determinant of the deformation gradient is proportional to the volume of the tetrahedron formed by the spherical parameter coordinates and the origin. In the same way, we can also replace the triangle deformation gradient in other geometric distortion energies with the corresponding tetrahedron deformation gradient to obtain the corresponding spherical parameterization distortion energies. The advantage of this definition is that its form is much simpler than the integral form mentioned earlier, so computing its gradient and Hessian becomes easy, and the corresponding planar optimization methods can also be readily applied to spherical parameterization. Moreover, this tetrahedron-based definition is a very good approximation of the spherical triangles discussed earlier, because their degeneration behaviors are consistent, and when the mesh is large every triangle is very small, so that the cone formed by a spherical triangle and the origin can be approximated by the corresponding tetrahedron.

3.2.4 Formulation of the Geometric Optimization Problem for Spherical Parameterization

Finally, we formulate the low-distortion spherical parameterization problem as the problem of optimizing a spherical distortion energy subject to the spherical constraint and the constraint that no spherical triangle flips:

$$\begin{aligned} \boldsymbol{x}^* &= \operatorname*{argmin}_{\boldsymbol{x}} E^{sphere}(M, \boldsymbol{x}), \\ \text{s.t. } \|\boldsymbol{x}_v\| &= 1, \qquad \forall v \in V, \\ \det(\boldsymbol{x}_{f_0}, \boldsymbol{x}_{f_1}, \boldsymbol{x}_{f_2}) &> 0, \qquad \forall f \in F, \end{aligned}$$

where

$$E^{sphere}(M, \boldsymbol{x}) = \frac{1}{A(M)} \sum_{f \in F} |f| \, D^{sphere}(J_{f,tet}).$$

Note that, since the sphere has no boundary, a spherical parameterization map satisfying the local condition that no spherical triangle flips is a bijection, so there is no need to add a global bijectivity condition. The above problem is a nonconvex optimization problem under a manifold constraint and a nonlinear open-set constraint; therefore, besides using the optimization methods of planar parameterization, the spherical constraint must also be handled well in order to ensure that the distortion energy decreases smoothly. In the next section we will describe in detail the application of various planar geometric optimization solvers to this spherical parameterization problem, as well as how to handle the spherical constraint.

3.3 Geometric Optimization of Spherical Parameterizations

In this section, we extend the geometric optimization pipeline of flip-free planar parameterization to spherical parameterization. First, let us review the geometric optimization pipeline of planar parameterization [43]: we obtain a flip-free planar parameterization by Tutte's method as the initialization, and then use an iterative process to decrease the distortion energy until convergence. In each iteration, we first compute the gradient of the distortion energy, then obtain a descent direction of the distortion energy through a positive semidefinite solver, then compute along this descent direction the maximal step size that guarantees no triangle flips, then, starting from the maximal step size, perform a backtracking line search along this descent direction until a suitable step size is found, and finally update the parameter coordinates of all vertices. We now extend each of these steps to spherical parameterization.

3.3.1 Spherical Initialization

First, we need a flip-free spherical parameterization as the initialization. In Section 1 we introduced two methods that are guaranteed to produce a flip-free spherical parameterization: the gnomonic projection method and the multi-resolution method. The gnomonic projection method effectively transfers the parameter coordinates produced by Tutte's method [46] for planar parameterization onto the sphere, and is therefore very fast; its drawback is that it is prone to flips caused by limited numerical precision. The multi-resolution method inserts the vertices into the sphere one by one in a certain order by topological means, and is therefore more tolerant of numerical precision; its drawback is that it is time-consuming. Whichever method is chosen, as long as it yields a flip-free spherical parameterization, it can serve as the initialization, after which geometric optimization is used to decrease the distortion energy.

3.3.2 Optimization Solvers

In Chapter 2 we introduced various solvers for planar geometric optimization, which fall mainly into two classes: preconditioners and second-order solvers. Among them, the CM solver exploits the special form of the two-dimensional singular value decomposition and therefore cannot be extended to the sphere, while all the other solvers have three-dimensional forms and can therefore be used for the spherical parameterization problem. Here we present the spherical forms of these solvers.

1. Sobolev preconditioner [21]

The Sobolev preconditioner directly uses the Laplacian matrix of the mesh as the preconditioner; it depends neither on the target energy, nor on the specific parameter coordinates, nor even on the dimension of the problem. Therefore, this preconditioner can be applied without modification to the optimization of the distortion energy of spherical parameterizations. Its advantage is that it can be pre-factorized in advance, so each iteration takes very little time; its drawback is slow convergence.

2. SLIM preconditioner [32]

The SLIM preconditioner is also easy to extend to the sphere in form. Its basic idea is to apply a certain correction to the Sobolev preconditioner for different target energies; concretely, it locally approximates the target energy by a class of weighted quadratic functions. We use the spherical Dirichlet energy as an example to demonst this process. For a triangle $f = (f_0, f_1, f_2) \in F$ of the mesh M, its weighted approximation is

$$P_W^{sphere}(J_f) = \|W_f(J_f - R_f)\|_F^2, \qquad \text{s.t. } \nabla_{J_f} P_W^{sphere}(J_f) = \nabla_{J_f} D_{symDir}^{sphere}(J_f),$$

where we assume by default that J_f is the deformation gradient between tetrahedra. Computing the gradient of this approximating function, we obtain

$$\begin{aligned} \nabla_{J_f} P_W^{sphere}(J_f) = 2W_f^T W_f(J_f - R_f) = \nabla_{J_f} D_{symDir}^{sphere}(J_f), \\ \Rightarrow W_f = \left(\frac{1}{2}\nabla_{J_f} D_{symDir}^{sphere}(J_f)(J_f - R_f)^{-1}\right)^{1/2}. \end{aligned}$$

Writing each term in the form of a singular value decomposition, we have

$$\begin{aligned} J_f = U_f \operatorname{diag}(\sigma_{1,f}, \sigma_{2,f}, \sigma_{3,f})\, V_f^T, \qquad R_f = U_f V_f^T, \\ \nabla_{J_f} D_{symDir}^{sphere}(J_f) = U_f \operatorname{diag}\left(2\sigma_{1,f} - \frac{2}{\sigma_{1,f}^3}, 2\sigma_{2,f} - \frac{2}{\sigma_{2,f}^3}, 2\sigma_{3,f} - \frac{2}{\sigma_{3,f}^3}\right) V_f^T, \\ \Rightarrow W_f = U_f \operatorname{diag}\left(\sqrt{\frac{(\sigma_{1,f}+1)(\sigma_{1,f}^2+1)}{\sigma_{1,f}^3}}, \sqrt{\frac{(\sigma_{2,f}+1)(\sigma_{2,f}^2+1)}{\sigma_{2,f}^3}}, \sqrt{\frac{(\sigma_{3,f}+1)(\sigma_{3,f}^2+1)}{\sigma_{3,f}^3}}\right) U_f^T. \end{aligned}$$

Therefore, the spherical SLIM local preconditioner $H_{SLIM,f}^{sphere}$ is

$$H_{SLIM,f}^{sphere} = W_f^T W_f \otimes (\boldsymbol{u}_{f_0}, \boldsymbol{u}_{f_1}, \boldsymbol{u}_{f_2})^{-1} (\boldsymbol{u}_{f_0}, \boldsymbol{u}_{f_1}, \boldsymbol{u}_{f_2})^{-1^T}.$$

Weighting the local preconditioners by the triangle areas, we obtain the global SLIM preconditioner:

$$H_{SLIM}^{sphere} = \frac{1}{A(M)} \sum_{f \in F} |f|\, H_{SLIM,f}^{sphere}.$$

The spherical form of this preconditioner is relatively simple because we define the symmetric Dirichlet energy via tetrahedra, so that it can be written as a function of the three singular values of the deformation gradient, and moreover a separable one:

$$D_{symDir}^{sphere}(J_f) = \frac{1}{2} \sum_{k=1}^{3} \left(\sigma_{k,f}^2 + \frac{1}{\sigma_{k,f}^2}\right).$$

For other geometric energies, their tetrahedral-form SLIM preconditioners can be computed similarly. It is worth noting that if the distortion energy is defined in the integral form, the local energy does not have such a concise form, and the gradient-matching condition must then be used to compute the weighting matrix at every point. But even once the weighting matrices are computed, the deformation gradient is then not constant within a triangle, so the local preconditioners computed at each point would have to be integd over the triangle, and the amount of computation involved is unimaginable.

3. AKVF preconditioner [5]

The essence of the AKVF preconditioner is to regard parameterizations in the solution space that differ by a rigid transformation as equivalent, and to define a Riemannian metric on the manifold formed by these equivalence classes. For ease of computation, this Riemannian metric is written in the form of a positive semidefinite matrix on the original solution space and used as the preconditioner. For spherical parameterization, since the parameter coordinates of each vertex are written in three-dimensional form but are constrained to move on the sphere, the solution space is a $2|V|$-dimensional manifold in the $3|V|$-dimensional Euclidean space. Rigid transformations of the sphere have three dimensions—the position of the north pole accounts for two, and the rotation about the north pole accounts for one. After quotienting out these three dimensions, the actual solution space is a manifold of dimension $2|V|-3$, which agrees with the dimension of the actual solution space of planar parameterizations. Since a Riemannian metric on this manifold is inconvenient to define, we still choose to define a pseudo-Riemannian metric on the original Euclidean space such that the Killing vectors at each point are exactly all the vectors perpendicular to the tangent space of the manifold at that point. These vectors are, in essence, the vector fields that preserve the symmetry of the spherical parameterization at the current state, and they all have the form

$$\boldsymbol{d}_v = A\boldsymbol{x}_v, \qquad A + A^T = 0.$$

For a triangle $f = (f_0, f_1, f_2) \in F$ of the mesh M, from the system of equations

$$\boldsymbol{d}_{f_k} = A_f \boldsymbol{x}_{f_k}, \qquad k = 1, 2, 3$$

one can solve for a unique A_f; by measuring its antisymmetry we can tell how far it is from a rigid transformation, i.e., we use $\|X + X^T\|_F^2$ as a seminorm on the general linear space of order three, $GL_3(\mathbb{R}) = \mathbb{R}^{3\times 3}$. As in the planar case, this seminorm can be extended to a semi-inner product $\langle X, Y\rangle = \mathrm{tr}(X + X^T)(Y + Y^T)$, which is a positive semidefinite quadratic form on $GL_3(\mathbb{R})$. Since the vector field $\boldsymbol{d}$ is unknown during the iterations, the local AKVF preconditioner is taken to be the representation of this positive semidefinite quadratic form under the unit vector fields; that is, we solve for A_f sepaly for each unit tangent vector field, compute the pairwise semi-inner products among the 9 matrices A_f of each triangle, and fill them in order into a 9×9 matrix $H^{sphere}_{AKVF,f}(\boldsymbol{x})$. Finally, we assemble these matrices weighted by area to obtain the pseudo-Riemannian metric $H^{sphere}_{AKVF}(\boldsymbol{x})$ on the original Euclidean space, which is the global spherical AKVF preconditioner. It can be seen that the spherical AKVF preconditioner is computed in a way very similar to the planar case; only the form of the Killing vector fields differs slightly. The spherical AKVF preconditioner also depends only on the connectivity of the mesh; it is independent of the geometric information of the mesh and of the choice of the target energy.

4. Analytic projected Newton solver [42]

The three preconditioners introduced above either do not depend on the choice of the target energy at all or depend only on its first-order derivative information; they are therefore all first-order solvers and converge relatively slowly. We now introduce a second-order solver. As in the planar case, we hope to express the eigendecomposition of the Hessian of the spherical geometric energy analytically, so that a positive semidefinite solver can be obtained by setting its negative eigenvalues to 0. Fortunately, [42] gives the eigendecompositions of the Hessians of commonly used three-dimensional isotropic geometric energies. We take the spherical symmetric Dirichlet

energy as an example here:

$$E^{sphere}_{symDir}(M,\boldsymbol{x}) = \frac{1}{A(M)}\sum_{f\in F}|f|\, D^{sphere}_{symDir}(J_f),$$
$$D^{sphere}_{symDir}(J_f) = \frac{1}{2}II_f + \frac{1}{8}\left(\frac{I_f^2 - II_f}{III_f}\right)^2 - \frac{I_f}{III_f},$$

where

$$I_f = \sigma_{1,f} + \sigma_{2,f} + \sigma_{3,f}, \qquad II_f = \sigma_{1,f}^2 + \sigma_{2,f}^2 + \sigma_{3,f}^2, \qquad III_f = \sigma_{1,f}\sigma_{2,f}\sigma_{3,f}$$

are the three invariants of the stretch tensor of the tetrahedral deformation gradient of triangle f. The eigendecomposition of the Hessian of the local Dirichlet energy with respect to the deformation gradient can then be written as [42]

$$\nabla^2_{J_f} D^{sphere}_{symDir}(J_f) = \sum_{k=1}^{3}\left(\lambda_{\boldsymbol{d}_{k,f}}\boldsymbol{d}_{k,f}\boldsymbol{d}_{k,f}^T + \lambda_{\boldsymbol{l}_{k,f}}\boldsymbol{l}_{k,f}\boldsymbol{l}_{k,f}^T + \lambda_{\boldsymbol{t}_{k,f}}\boldsymbol{t}_{k,f}\boldsymbol{t}_{k,f}^T\right),$$

where

$$\lambda_{\boldsymbol{d}_{k,f}} = 1 + \frac{3}{\sigma^4_{k,f}}, \qquad \boldsymbol{d}_{k,f} = \mathrm{vec}\left(U_f E_{kk} V_f^T\right),$$
$$\lambda_{\boldsymbol{l}_{1,f}} = 1 + \frac{\sigma^2_{1,f}}{III_f^2} + \left(II_f - \sigma^2_{1,f}\right)\frac{\sigma^3_{1,f}}{III_f^3}, \qquad \boldsymbol{l}_{1,f} = \mathrm{vec}\left(\frac{1}{\sqrt{2}}U_f\begin{pmatrix}0&0&0\\0&0&1\\0&1&0\end{pmatrix}V_f^T\right),$$
$$\lambda_{\boldsymbol{t}_{1,f}} = 1 + \frac{\sigma^2_{1,f}}{III_f^2} - \left(II_f - \sigma^2_{1,f}\right)\frac{\sigma^3_{1,f}}{III_f^3}, \qquad \boldsymbol{t}_{1,f} = \mathrm{vec}\left(\frac{1}{\sqrt{2}}U_f\begin{pmatrix}0&0&0\\0&0&1\\0&-1&0\end{pmatrix}V_f^T\right).$$

For the second and third groups of eigenvalues and eigenvectors we have listed only one pair each, since the other two pairs can be obtained by cyclically permuting the indices. Note that the first two groups of eigenvalues are always positive; only the third group may contain negative eigenvalues. Therefore, our analytic projected Newton solver is

$$H^{sphere}_{APN} = \frac{1}{A(M)}\sum_{f\in F}|f|\sum_{k=1}^{3}\left(\lambda_{\boldsymbol{d}_{k,f}}\boldsymbol{d}_{k,f}\boldsymbol{d}_{k,f}^T + \lambda_{\boldsymbol{l}_{k,f}}\boldsymbol{l}_{k,f}\boldsymbol{l}_{k,f}^T + \max\left(\lambda_{\boldsymbol{t}_{k,f}},0\right)\boldsymbol{t}_{k,f}\boldsymbol{t}_{k,f}^T\right).$$

3.3.3 Handling the Spherical Constraint

If we directly use the above solvers to compute the descent direction, i.e., compute the descent direction by

$$\boldsymbol{d} = -H^{-1}\nabla_{\boldsymbol{x}}E(M,\boldsymbol{x})$$

and then search for a step size along this descent direction, the parameter coordinates that were originally constrained to the sphere will certainly leave the sphere. Fortunately, there is a very simple way to pull points that have left the sphere back onto it, namely the gnomonic map. More specifically, for a spatial parameterization $\boldsymbol{x}$ of the mesh M (not necessarily on the sphere), denote its gnomonic projection by $P(\boldsymbol{x})$; concretely, at a vertex $v \in V$,

$$P(\boldsymbol{x})_v = \frac{\boldsymbol{x}_v}{\|\boldsymbol{x}_v\|}.$$

Note that for a map that is already constrained to the sphere, $P(\boldsymbol{x}) = \boldsymbol{x}$. Thus, we can extend the definition of the energy of spherical parameterizations to general spatial parameterizations:

$$\tilde{E}(M, \boldsymbol{x}) := E(M, P(\boldsymbol{x})).$$

In this sense, we no longer need to worry about the spherical constraint, since we only need to take the gradient of this newly defined energy function of general spatial maps, and, after obtaining the descent direction $\boldsymbol{d}$ and the step size α, we only need to update the parameter coordinates by

$$\boldsymbol{x} \leftarrow P(\boldsymbol{x} + \alpha \boldsymbol{d}),$$

which guarantees that the parameter coordinates after each iteration's update always lie on the sphere. More specifically, when taking the gradient,

$$\nabla_{\boldsymbol{x}} \tilde{E}(M, \boldsymbol{x}) = dP\left(\nabla_{\boldsymbol{x}} E(M, \boldsymbol{x})\right).$$

For a vertex $v \in V$,

$$dP\left(\nabla_{\boldsymbol{x}} E(M, \boldsymbol{x})\right)_v = \left(I - \boldsymbol{x}_v \boldsymbol{x}_v^T\right) \nabla_{\boldsymbol{x}} E(M, \boldsymbol{x})_v.$$

This means that, before solving for the descent direction, the radial component of the original gradient at each vertex must be filtered out, so that the direction of motion at every vertex is tangent to the sphere (Figure 3.9); the descent direction is then computed by

$$\boldsymbol{d} = -H^{-1} dP\left(\nabla_{\boldsymbol{x}} E(M, \boldsymbol{x})\right).$$

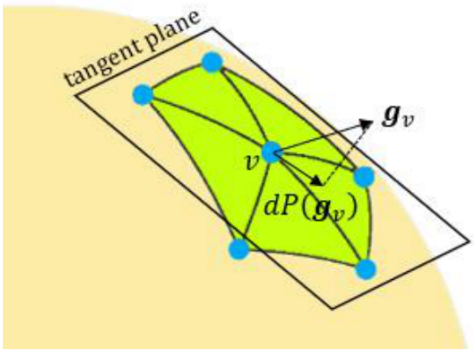


Figure 3.9: Tangential projection of the gradient.

Through experiments we found that only with this projection can a good descent direction be obtained; without this projection, the maximal step size along the resulting descent direction is often very small, causing the optimization to get stuck easily at the very beginning. This projection is therefore crucial. In fact, a similar projection is also needed for general manifold constraints [26, 27].

3.3.4 Line Search

With the descent direction in hand, we need to compute a suitable step size. Since we ultimately update the parameter coordinates by

$$\boldsymbol{x} \leftarrow P(\boldsymbol{x} + \alpha \boldsymbol{d}),$$

this is equivalent to searching along great-circle arcs on the sphere. In the step-size search, the problem we solve is

$$\begin{aligned} \alpha^* &= \operatorname*{argmin}_{\alpha} E\left(M, P(\boldsymbol{x} + \alpha \boldsymbol{d})\right), \\ \text{s.t. } & \det\left(\boldsymbol{x}_{f_0} + \alpha \boldsymbol{d}_{f_0}, \boldsymbol{x}_{f_1} + \alpha \boldsymbol{d}_{f_1}, \boldsymbol{x}_{f_2} + \alpha \boldsymbol{d}_{f_2}\right) > 0, \qquad \forall f \in F. \end{aligned}$$

We first need to compute a maximal step size. As in the planar case, this maximal step size is the minimum of the maximal step sizes of all triangles, and in the spherical case the maximal step size of each triangle is the step size at which the corresponding spherical triangle just degenes, i.e., the step size at which the tetrahedron formed by the three spherical vertices and the origin just degenes. It is the smallest positive root of the following cubic equation:

$$\det\left(\boldsymbol{x}_{f_0} + \alpha \boldsymbol{d}_{f_0}, \boldsymbol{x}_{f_1} + \alpha \boldsymbol{d}_{f_1}, \boldsymbol{x}_{f_2} + \alpha \boldsymbol{d}_{f_2}\right) = 0.$$

Since the analytic solution of a cubic equation is rather complicated, we can use a numerical library (such as gsl) to solve the above equation. Once the maximal step size is available, a suitable step size can be obtained by backtracking line search; this part is much the same as in the planar case, so we do not go into detail.

We have now described in detail the complete iterative process of geometric optimization for spherical parameterization; the termination criterion of the iterations is the same as for planar parameterization. As a summary, we write the whole pipeline above as a complete algorithm (Algorithm 2).

Algorithm 2 Iterative geometric optimization algorithm for spherical parameterization

$\boldsymbol{x} =$ Gnomonic_Tutte( )
while $\Delta E/E > \epsilon$ **do**
    $\boldsymbol{d} = -H^{-1}P\left(\nabla_{\boldsymbol{x}} E(M, \boldsymbol{x})\right)$
    $\alpha_{max} =$ max_step$(\boldsymbol{x}, \boldsymbol{d})$
    $\alpha =$ backtracking_line_search$(\boldsymbol{x}, \boldsymbol{d}, \alpha_{max})$
    $\boldsymbol{x} \leftarrow P(\boldsymbol{x} + \alpha \boldsymbol{d})$
end while

Here H depends on the choice of solver and E depends on the choice of energy. It can thus be seen that our geometric optimization framework for spherical parameterization applies to various geometric energies and various geometric optimization solvers. In addition, the various acceleration methods in planar geometric optimization can also be applied, with slight modifications, to the geometric optimization of spherical parameterizations; we do not elabo on them here.

3.4 Experimental Results

3.4.1 Implementation Details

Here we test the spherical geometric optimization pipeline described in the previous section. We implemented the above algorithm in C++ on a DELL computer with an 8-core i7 processor running Windows, and tested our algorithm with the mesh-visualization graphical user interface provided by [52]. For the initialization, we randomly select 4 seed triangles and use the method of [7] to obtain a partition of the mesh. This partition guarantees that each patch is connected, but it cannot guarantee that the adjacency between patches is consistent with that of a tetrahedron. If this is violated, we re-select the seed triangles until a partition satisfying the above condition is found. We straighten the partition boundaries by shortest paths and map them to the edges of a regular tetrahedron, then use Tutte's method [46] to map the vertices inside each patch into the faces of the regular tetrahedron, and then use the gnomonic projection to map the parameter coordinates on the tetrahedron onto the sphere, obtaining the spherical initialization. During the iterations, we use Eigen for singular value decompositions and matrix operations, use Pardiso [29, 30] to solve the corresponding quadratic programming problem when computing the descent direction, and use gsl to solve the corresponding cubic equation when computing the maximal step size.

3.4.2 Results

1. Comparison of the solvers

Model Name: Horse, $|V| = 2000$, $|E| = 5994$, $|F| = 3996$.

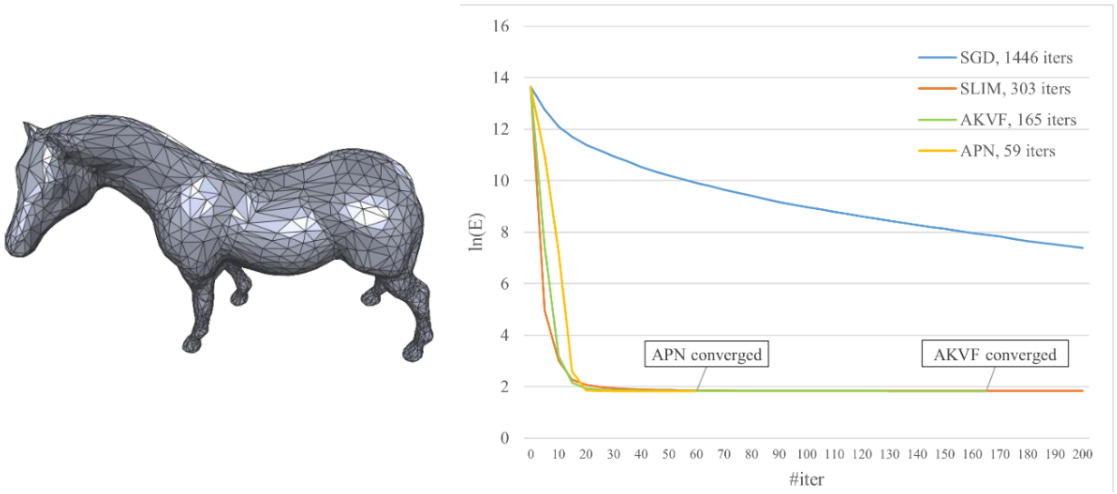


Figure 3.10: Comparison of the solvers on the Horse model.

Model Name: Spot, $|V| = 2930$, $|E| = 8784$, $|F| = 5856$.

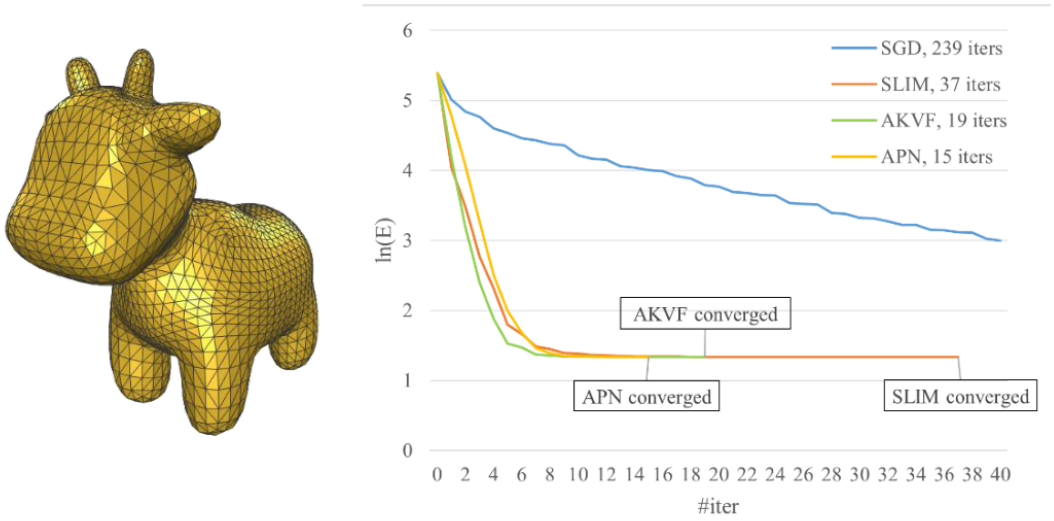


Figure 3.11: Comparison of the solvers on the Spot model.

2. Spherical parameterization results

3.4.3 Conclusion

From the above experiments, we conclude that our proposed geometric optimization pipeline solves the spherical parameterization problem well, producing low-distortion, flip-free spherical parameterizations. Among the solvers, the Sobolev preconditioner ites quickly, but each iteration reduces the distortion only a little, so that the number of iterations needed for convergence is far larger than for the other solvers; the SLIM preconditioner descends quickly at first, but converges very slowly; the AKVF preconditioner and the analytic Newton solver both perform

Model Name: Ant, $|V| = 9501$, $|E| = 28497$, $|F| = 18998$.

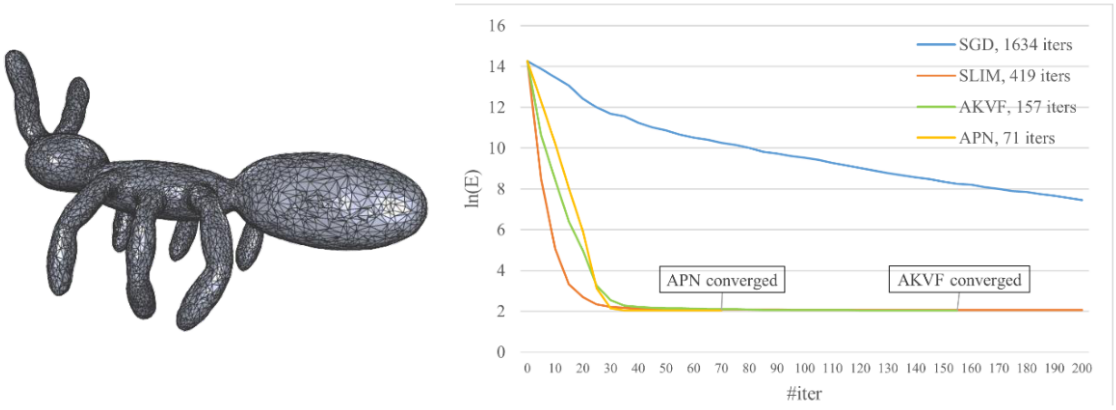


Figure 3.12: Comparison of the solvers on the Ant model.

Model Name: Hand, $|V| = 42068$, $|E| = 126198$, $|F| = 84132$.

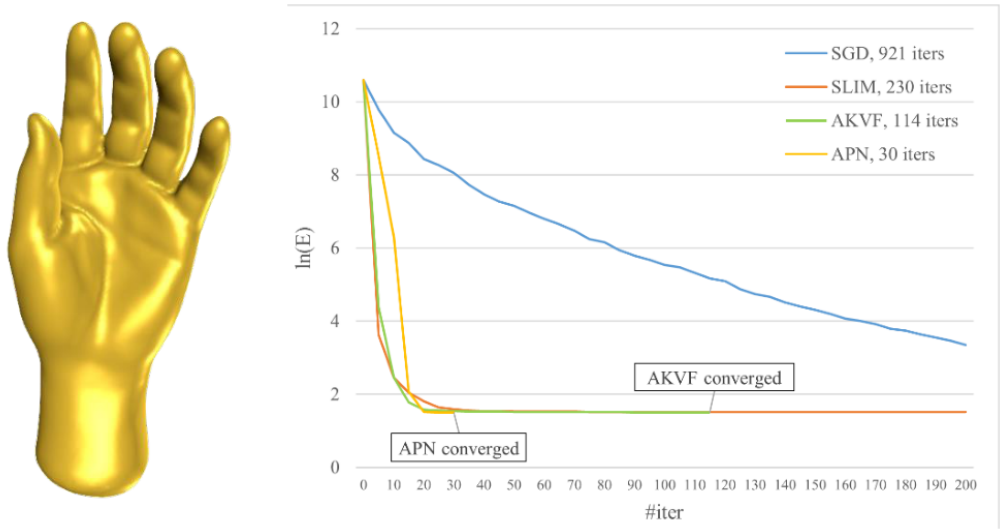


Figure 3.13: Comparison of the solvers on the Hand model.

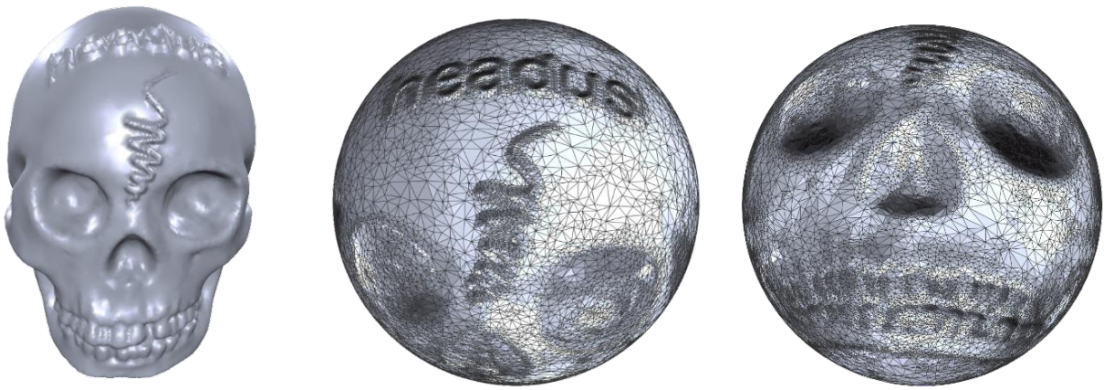


Figure 3.14: Spherical parameterization of the Skull model.



Figure 3.15: Spherical parameterization of the Buddha model.



Figure 3.16: Spherical parameterization of the Max-Planck model.



Figure 3.17: Spherical parameterization of the Bunnyih model.

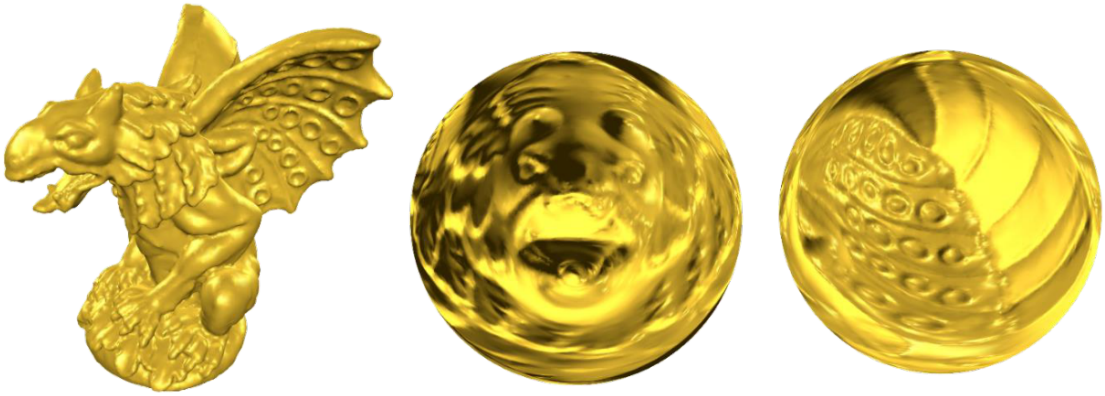


Figure 3.18: Spherical parameterization of the Gargoyle model.

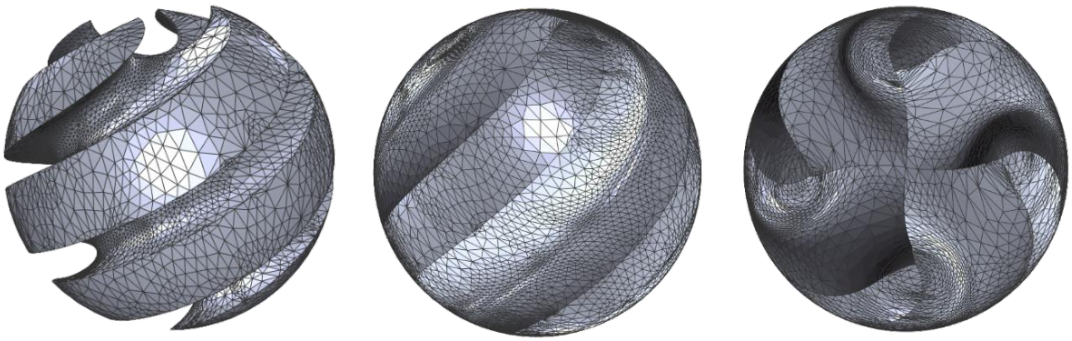


Figure 3.19: Spherical parameterization of the Sharp-sphere model.



Figure 3.20: Spherical parameterization of the Lucy model.



Figure 3.21: Spherical parameterization of the Armadillo model.

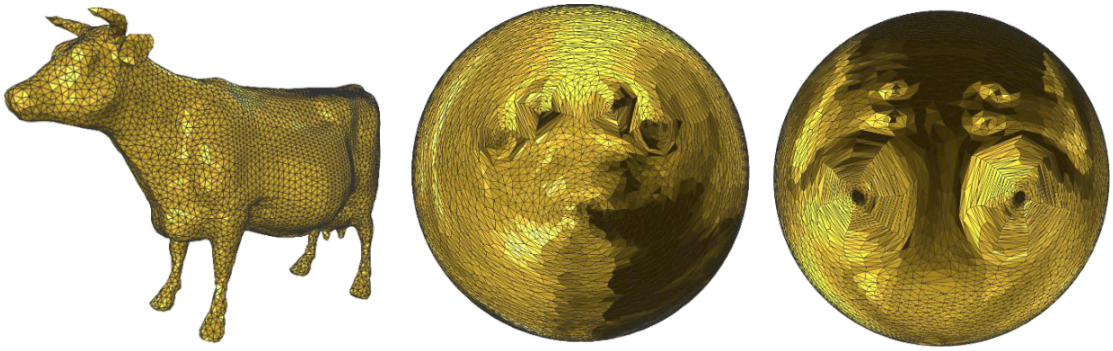


Figure 3.22: Spherical parameterization of the Cow model.

well, with the analytic Newton solver converging fastest. Compared with other methods, our method always obtains results with relatively low distortion, rather than getting stuck at high distortion. For large meshes, our spherical parameterizations clearly reflect the various details of the surface.

Chapter 4

Summary and Reflections

So far, we have introduced the mesh parameterization problem, presented in detail various geometric optimization methods for planar parameterization, surveyed existing spherical parameterization methods, then applied the geometric optimization methods of the plane to the solution of the spherical parameterization problem in our own way, and demonstd the results with experiments. In using geometric optimization to solve the spherical parameterization problem, we carefully analyzed the characteristics of spherical parameterizations and pointed out how they differ in essence from the planar parameterization problem. We also pointed out the importance of the gnomonic projection, and used this map repeatedly to build a bridge from planar parameterization to the sphere, thereby obtaining a general geometric optimization framework for spherical parameterization that applies to various geometric energies and various geometric optimization solvers. Our special treatment of the spherical parameterization problem enables us to optimize the distortion of spherical parameterizations efficiently, so that starting from an arbitrarily highly distorted initialization we can quickly obtain a low-distortion, flip-free spherical parameterization. No other method can currently achieve this.

Many problems concerning spherical parameterization remain to be studied, for example a robust solver for the Colin de Verdière equations, and how to deal with precision problems in the initialization caused by overly large meshes. For the former, we attempted to design a solver; although this solver is very fast, it cannot guarantee a flip-free result, so it is not discussed in this thesis. For the latter, [40] proposed a solution for the planar case, using precisely the multi-resolution technique. However, as mentioned earlier, this technique needs to remove the vertices of the mesh one by one and then insert them back in reverse order, which is very time-consuming for large-scale meshes. Here we propose the following idea: since Tutte's method solves for everything globally while the multi-resolution technique treats each vertex individually, we may design a compromise stgy, namely layer the mesh from coarse to fine, use the multi-resolution technique during the layering, and use Tutte's method when solving for the parameter coordinates of each layer. More specifically, for a large-scale mesh [50], we sample a number of seed points on it and then use the method of [7] to construct a base mesh as an approximation of the original mesh. We first parameterize this base mesh, and then parameterize the vertices of the original mesh covered by each triangle of the base mesh into the interior of the corresponding image triangle. For even larger meshes, we can compute a coarse-to-fine sequence of base meshes such that the number of triangles of the next base mesh covered by each triangle of a base mesh does not exceed a fixed threshold (e.g., 100), down to the finest mesh, i.e., the original mesh. This structure thus forms a tree: each level of the tree represents a base mesh, each node represents a triangle of a base mesh, and its children are all the triangles of the next base mesh that it covers. Building this tree requires the multi-resolution technique, while the parameterization of each node of the tree requires Tutte's method. During geometric optimization, we can partition the vertices of the original mesh into blocks according to the nodes of the tree and then perform block coordinate descent in depth-first or breadth-first

traversal order of the tree. This guarantees that all kinds of geometric optimization solvers can be used within each block while keeping the size of the subproblems mode. The above is only a rough idea; it applies to general processing of ultra-large-scale meshes, but many details deserve careful study, so we take it as the goal of our next work.

In addition, besides the plane and the sphere, there are mesh parameterization problems on other parameter domains. The spherical parameter domain is suitable only for the global parameterization of closed surface meshes of genus 0. For surface meshes of higher genus, the hyperbolic plane can be considered as the parameter domain; the related parameterization problems are also well worth studying, and we leave this part of the work to the future as well. A more long-term goal is to solve all mesh parameterization problems within a unified framework; this is clearly quite difficult at present, but worth attempting.

References

[1] Jiagui Peng, Qing Chen. *Differential Geometry* (in Chinese). Higher Education Press.

[2] Donald G. Anderson. Iterative Procedures for Nonlinear Integral Equations. J. ACM 12, 4 (1965), 547–560.

[3] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, B. Levy. Polygon Mesh Processing. A K Peters (2010).

[4] M. Botsch, S. Steinberg, S. Bischoff, L. Kobbelt. OpenMesh – a generic and efficient polygon mesh data structure.

[5] S. Claici, M. Bessmeltsev, S. Schaefer, and J. Solomon. Isometry-Aware Preconditioning for Mesh Parameterization. Comp. Graphics. Forum 36, 5 (2017), 37–47.

[6] Keenan Crane. Discrete Differential Geometry: An Applied Introduction.

[7] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, Multiresolution Analysis of Arbitrary Meshes. ACM SIGGRAPH (1995), 173–182.

[8] Michael S. Floater. Parameterization and smooth approximation of surface triangulations. Computer Aided Geometric Design 14 (1997), 231–250.

[9] Michael S. Floater. One-to-one Piecewise Linear Mappings over Triangulations. Math. Comput. 72 (2003), 685–696.

[10] Michael S. Floater and Kai Hormann. Surface parameterization: a tutorial and survey. In Advances in Multiresolution for Geometric Modelling. Springer, 157–186.

[11] Xiao-Ming Fu, Yang Liu, and Baining Guo. Computing locally injective mappings by advanced MIPS. ACM Trans. Graph. 34, 4 (2015), 71:1–71:12.

[12] Craig Gotsman, Xianfeng Gu, and Alla Sheffer. Fundamentals of Spherical Parameterization for 3D Meshes. ACM Trans. Graph. (2003), 358–364.

[13] Xianfeng Gu, Steven Gortler, and Hugues Hoppe. Geometry Images. ACM SIGGRAPH (2002), 356–361.

[14] 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 Transaction on Medical Imaging, 23, 7 (2004).

[15] Xianfeng Gu and Shing-Tung Yau. Computing Conformal Structures of Surfaces. Communications in Information and Systems. 2, 2 (2002), 121–146.

[16] Xianfeng Gu and Shing-Tung Yau. Global Conformal Surface Parameterization. Symposium on Geometry Processing. (2003), 127–137.

[17] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle. Conformal Surface Parameterization for Texture Mapping. IEEE TVCG 6, 2 (2000), 181–189.

[18] Kai Hormann and Gunther Greiner. MIPS: An Efficient Global Parameterization Method. Curve and Surface Design: Saint Malo 1999. Vanderbilt University Press (2000), 153–162.

[19] Xin Hu, Xiao-Ming Fu, and Ligang Liu. Advanced Hierarchical Spherical Parameterizations. IEEE T. Vis. Comput. Gr. 24, 6 (2018), 1930–1941.

[20] Martin Isenburg, Stefan Gumhold, and Craig Gotsman, Connectivity Shapes. Proceedings of IEEE Visualization (2001), 135–142.

[21] Shahar Z. Kovalsky, Meirav Galun, and Yaron Lipman. Acceled Quadratic Proxy for Geometric Optimization. ACM Trans. Graph. 35, 4 (2016), 134:1–134:11.

[22] H. Laga, Y. Guo, H. Tabia, R. B. Fisher, M. Bennamoun. 3D Shape Analysis: Fundamentals, Theory, and Applications. Wiley (2019).

[23] Ligang Liu, Chunyang Ye, Ruiqi Ni, and Xiao-Ming Fu. Progressive Parameterizations. ACM Trans. Graph. 37, 4 (2018), 41:1–41:12.

[24] Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. A Local/Global Approach to Mesh Parameterization. Comp. Graphics. Forum (SGP) 27, 5 (2008), 1495–1504.

[25] Yurii Nesterov. A method of solving a convex programming problem with convergence $(1/k^2)$. Soviet Mathematics Doklady, 27 (1983), 372–376.

[26] Yurii Nesterov. Lectures on Convex Optimization. Springer International Publishing (2018).

[27] J. Nocedal and S. Wright. Numerical Optimization. Springer Science & Business Media (2006).

[28] Y. Peng, B. Deng, J. Zhang, F. Geng, W. Qin, L. Liu. Anderson Acceleration for Geometry Optimization and Physics Simulation. ACM Trans. Graph. 37, 4 (2018), 42:1–42:14.

[29] C. G. Petra, O. Schenk, M. Anitescu. Real-time Stochastic Optimization of Complex Energy Systems on High-performance Computers. Computing in Science & Engineering 16, 5 (2014), 32–42.

[30] C. G. Petra, O. Schenk, M. Lubin, K. Gärtner. An augmented incomplete factorization approach for computing the Schur complement in stochastic optimization. SIAM J. Sci. Comp. 36, 2 (2014), C139–C162.

[31] Emil Praun and Hugues Hoppe. Spherical Parameterization and Remeshing. ACM SIGGRAPH (2003), 340–350.

[32] Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. Scalable Locally Injective Mappings. ACM Trans. Graph. 36, 2 (2017), 16:1–16:16.

[33] S. Saba, I. Yavneh, C. Gotsman, and A. Sheffer, Practical Spherical Embedding of Manifold Triangle Meshes. Proceedings of Shape Modeling International (2005).

[34] P. Sander, J. Snyder, S. Gortler, and H. Hoppe. Texture Mapping Progressive Meshes. ACM SIGGRAPH (2001) 409–416.

[35] P. Sander, Z. Wood, S. Gortler, J. Snyder, and H. Hoppe, Multi-chart Geometry Images. ACM Symposium on Geometry Processing (2003).

[36] Ariel Shamir. A Formulation of Boundary Mesh Segmentation. International Symposium on 3D Data Processing, Visualization and Transmission (2004), 82–89.

[37] Ariel Shamir. A Survey on Mesh Segmentation Techniques. Comp. Graphics. Forum 27, 6 (2008), 1539–1556.

[38] Avner Shapiro and Ayellet Tal. Polyhedron Realization for Shape Transformation. The Visual Computer 14, 8 (1998), 429–444.

[39] Alla Sheffer, Emil Praun, and Kenneth Rose. Mesh parameterization methods and their applications. Found. Trends. Comput. Graph. Vis. 2, 2 (2006), 105–171.

[40] Hanxiao Shen, Zhongshi Jiang, Denis Zorin, and Daniele Panozzo. Progressive Embedding. ACM Trans. Graph. 38, 4 (2019), 32:1–32:13.

[41] Anna Shtengel, Roi Poranne, Olga Sorkine-Hornung, Shahar Kovalsky, and Yaron Lipman. Geometric Optimization via Composite Majorization. ACM Trans. Graph. 36, 4 (2017), 38:1–38:11.

[42] Breannan Smith, Fernando De Goes, and Theodore Kim. Analytic Eigensystems for Isotropic Distortion Energies. ACM Trans. Graph. 38, 1 (2019), 3:1–3:15.

[43] Jason Smith and Scott Schaefer. Bijective Parameterization with Free Boundaries. ACM Trans. Graph. 34, 4 (2015), 70:1–70:9.

[44] Jian-Ping Su, Chunyang Ye, Ligang Liu, Xiao-Ming Fu. Efficient Bijective Parameterizations. ACM Trans. Graph. 39, 4 (2020).

[45] Loring W. Tu. An Introduction to Manifolds, Second Edition. Springer-Verlag New York (2011).

[46] W. T. Tutte. How to draw a graph. In Proceedings of the London Mathematical Society, Vol. 13, 747–767.

[47] S. Wan, T. Ye, M. Li, H. Zhang, X. Li. An Efficient Spherical Mapping Algorithm and its Application on Spherical Harmonics. Sci. China Inf. Sci. (2003), 56(9):1–10.

[48] Chunxue Wang, Xin Hu, Xiao-Ming Fu, Ligang Liu. Bijective Spherical Parameterization with low distortion. Computers & Graphics 58, (2016), 161–171.

[49] Chunxue Wang, Zheng Liu, Ligang Liu. As-Rigid-As-Possible Spherical Parameterization. Graph Models (2014), 76(5):457–67.

[50] Chunyang Ye, Jian-Ping Su, Ligang Liu, Xiao-Ming Fu. Memory-Efficient Bijective Parameterizations for Very-Large-Scale Models. Pacific Graphics 39, 7 (2020).

[51] Yufeng Zhu, Robert Bridson, and Danny M. Kaufman. Blended Cured Quasi-Newton for Distortion Optimization. ACM Trans. Graph. 37, 4 (2018), 40:1–40:14.

[52] Software: Surface Mesh Framework from Xiao-Ming Fu's group. Website: `http://staff.ustc.edu.cn/~fuxm/code/index.html#sec_surface_framework`