Surface Parametrization and Remeshing with Guarantees

by

Leyi Zhu

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

May, 2026

Professor Denis Zorin

Professor Daniele Panozzo

Dedication

To my parents, for their unwavering love, and to my advisors, for their invaluable guidance.

Acknowledgements

I would like to express my deepest gratitude to my advisors, Denis Zorin and Daniele Panozzo, for their guidance, support, and patience throughout my Ph.D. Their insight and encouragement shaped not only this thesis but also the way I think about research. I am especially grateful for their belief in me during the most difficult moments, and for helping me find the strength to keep going.

I am thankful to my family for their quiet and unwavering support over the years—it has been my foundation.

I would also like to thank my collaborators: Professor Marcel Campen, Dr. Michael Tao, Ryan Capouellez, Dr. Hanxiao Shen, Jiacheng Dai, and Dr. Yixin Hu. Exploring geometry together has been both intellectually rewarding and personally meaningful.

Finally, I want to thank my dear friends—Yi Yang, Yizhi Zhang, Tao Wang, Jinming Li, Yucong Shen, Esther Xie, and many others. You filled my years in the United States with laughter, even in the most bitter moments. Your friendship is one of the greatest treasures of my life.

Abstract

A core challenge in geometry processing is to construct surface and volume mappings that are robust, theoretically sound, and suitable for practical applications such as quadrangulation, texture mapping, and mesh adaptation. This thesis presents a unified framework for building such mappings with guarantees at three structural levels: geometric constraints, topological control, and bijective consistency across connectivity changes..

First, we develop a numerically robust algorithm for computing discrete conformal metrics that match prescribed curvature conditions at both interior vertices and surface boundaries. This work extends recent theoretical foundations to support surfaces with or without boundary, enabling controlled and guaranteed conformal parametrization under flexible geometric constraints.

Second, we address the global topological structure of seamless parametrizations by resolving a long-standing open question: which cross fields admit globally consistent parameterizations. We provide a theoretical characterization of admissible holonomy signatures and an accompanying algorithm that constructs seamless, locally injective maps with prescribed global holonomy, including nontrivial cycles. This result establishes precise conditions under which field-guided quadrangulation and globally consistent parameterization are possible.

Finally, we introduce a bijective remeshing framework that maintains an exact, continuous correspondence across sequences of connectivity-changing operations, including edge splits, collapses,

flips, and smoothing. Rather than relying on projection or interpolation, the global mapping is represented as a composition of local bijective atlases associated with individual mesh operations. This construction guarantees consistency of scalar fields, curves, and surfaces under remeshing, and naturally generalizes from surface meshes to volumetric tetrahedral meshes.

Together, these contributions form a coherent pipeline for surface parametrization and remeshing with guarantees on injectivity, curvature realization, holonomy, and mapping consistency—even in the presence of large deformations and nontrivial changes in mesh connectivity.

Contents

List of Figures

List of Tables

List of Appendices

1 Introduction

1.1 Motivation and Background

Mappings between geometric representations lie at the core of geometry processing, enabling tasks such as surface parametrization, quadrangulation, mesh adaptation, and simulation. In practice, however, these mappings are often constructed by numerical optimization and heuristic choices, and can fail in subtle but consequential ways: a map may become locally non-injective (fold-overs), violate boundary conditions, exhibit incorrect global "seam" behavior, or lose correspondence when the underlying discretization changes.

This thesis investigates how to build mapping pipelines with explicit guarantees. We focus on three recurring requirements: (1) *geometric validity* (e.g., curvature prescriptions and injectivity), (2) *topological consistency* (e.g., holonomy constraints for seamless maps), and (3) *stable correspondence* across remeshing operations. The subsequent chapters develop algorithms that make these guarantees enforceable in practice.

A key theme is that these requirements are coupled. Curvature prescriptions define an intrinsic metric that underlies many parametrization pipelines; holonomy constraints specify how locally valid charts must glue together globally; and remeshing changes the discretization on which both the metric and the map are represented, demanding correspondence mechanisms that preserve

(and therefore can transport) these structures.

1.2 Geometric Guarantees in Surface Parametrization

A first class of challenges arises at the geometric level. In surface parametrization and related deformation problems, one often seeks a map whose metric properties satisfy specific requirements: for instance, matching prescribed Gaussian curvature in the interior, controlling geodesic curvature along the boundary, and avoiding degenerate or flipped elements. These constraints are tightly coupled to discretization choices: the same surface geometry may admit or obstruct a desired solution depending on the intrinsic triangulation used, and numerical methods can become unstable when the mesh approaches Delaunay-critical configurations.

Recent advances in discrete metric uniformization [Gu et al. 2018b; Springborn 2020] provide a principled way to compute discrete conformal metrics with prescribed curvature, with guarantees derived from convex optimization. In practice, however, robust computation requires treating the intrinsic triangulation as part of the unknown: restricting to a fixed triangulation can obstruct curvature targets (e.g., due to violation of the triangle inequalities), and numerical methods must reliably handle Delaunay-critical events to avoid infinite flipping sequence.

Chapter 2 addresses these issues by formulating an efficient and robust construction of discrete conformal metrics with curvature prescriptions, explicitly accounting for boundaries and treating the intrinsic triangulation as part of the unknown.

1.3 Topological Guarantees and Global Structure

Beyond local geometric validity, many applications require control over global structure. A key example is *seamless* surface parametrization, which underlies seamless texture atlases and con-

forming surface quadrangulations. A seamless parametrization induces an intrinsically flat metric almost everywhere, with a small set of cone vertices whose angle deficit/excess is a multiple of $\frac{\pi}{2}$; more generally, the holonomy angle of any closed loop is a multiple of $\frac{\pi}{2}$ (cf. [Bright et al. 2017; Crane et al. 2010]). While there are infinitely many loops, their holonomy angles are determined by finitely many values on a basis, i.e., a *holonomy signature* that captures both local (around vertices) and global (non-contractible) aspects.

In many pipelines the target global structure is specified indirectly (e.g., by a cross field), yet existing methods may produce maps that are locally well-behaved while failing to realize the intended holonomy. This raises a fundamental question: for which holonomy signatures does a seamless parametrization exist, and how can one construct a parametrization that respects a prescribed signature?

Chapter 3 studies the relationship between cross-field topology and seamless parametrization topology and presents a method that enforces prescribed holonomy signatures while maintaining local injectivity, enabling reliable downstream applications such as quadrangulation.

1.4 Correspondence under Connectivity-Changing Remeshing

Many geometry processing pipelines involve remeshing operations that change mesh connectivity, such as edge collapses and splits, flips, and smoothing. While these operations can improve element quality or reduce complexity, they complicate data transfer: attributes (textures, simulation state, embedded curves, or intersection structures) must be mapped between changing discretizations.

Conventional solutions often rely on heuristics (e.g., barycentric interpolation or closest-point

projection [Kobbelt et al. 1998; Botsch et al. 2010]). While effective for small adjustments, these approaches provide no topological guarantees and can fail under substantial connectivity changes, producing non-bijective correspondences that manifest as drift, seam tearing, or loss of tracked features. This motivates representing the evolving correspondence explicitly as a map that remains continuous and bijective throughout a remeshing sequence.

Chapter 4 tackles this problem by constructing a continuous, bijective mapping through remeshing sequences via a composition of local bijective atlases, providing a rigorous foundation for tracking points, curves, and surfaces across both triangle and tetrahedral remeshing.

1.5 Contributions and Thesis Organization

This thesis develops algorithms for constructing mappings with (i) *geometric guarantees* (e.g., curvature control and local injectivity), (ii) *topological guarantees* (e.g., prescribed holonomy for seamless maps), and (iii) *correspondence guarantees* across connectivity-changing remeshing.

Chapter 2: Efficient and robust discrete conformal equivalence with boundary.. We present an efficient method to compute a discrete metric that is conformally equivalent to a given input metric while matching prescribed Gaussian curvature at interior vertices and prescribed geodesic curvature along the boundary. The construction builds on hyperbolic ideal Delaunay triangulations and treats the surface's intrinsic triangulation as a degree of freedom, enabling robust handling of both closed surfaces and surfaces with boundary as well as cut-aligned parametrizations.

Chapter 3: Global parametrization from prescribed holonomy signatures.. We study which cross fields admit quadrangulation from the viewpoint of global seamless parametrization. We propose a method that simultaneously guarantees local injectivity and offers full control over

holonomy by relating cross-field topology to parametrization topology and by modifying cut graphs to realize prescribed holonomy signatures.

Chapter 4: BijectiveRemesh—bijective correspondence through remeshing.. We introduce a framework for maintaining a continuous, bijective map through complex remeshing sequences in both 2D triangle meshes and 3D tetrahedral meshes. By chaining local bijective atlases for each primitive operation, the method produces a mathematically rigorous composite map that supports exact tracking of points, curves, and surfaces for applications such as texture transfer and simulation.

Chapter 5: Conclusion.. We summarize the contributions, discuss limitations, and outline promising directions for future research on guaranteed mappings in geometry processing.

2 | Efficient and Robust Discrete Conformal Equivalence with Boundary

This chapter is adapted from the publication on ACM Transactions on Graphics [Campen et al. 2021a], a joint work with Marcel Campen, Ryan Capouellez, Hanxiao Shen, Daniele Panozzo, and Denis Zorin.

Abstract

We describe an efficient algorithm to compute a discrete metric with prescribed Gaussian curvature at all interior vertices and prescribed geodesic curvature along the boundary of a mesh. The metric is (discretely) conformally equivalent to the input metric. Its construction is based on theory developed in [Gu et al. 2018b] and [Springborn 2020], relying on results on hyperbolic ideal Delaunay triangulations. Generality is achieved by considering the surface's intrinsic triangulation as a degree of freedom, and particular attention is paid to the proper treatment of surface boundaries. While via a double cover approach the case with boundary can be reduced to the case without boundary quite naturally, the implied symmetry of the setting causes addi-

tional challenges related to stable Delaunay-critical configurations that we address explicitly. We furthermore explore the numerical limits of the approach and derive continuous maps from the discrete metrics.

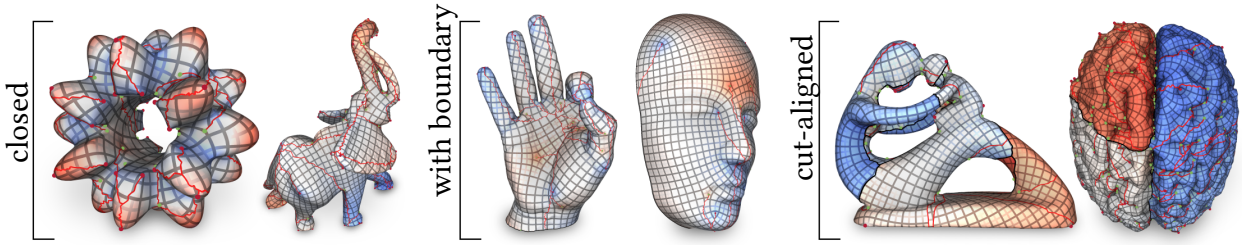


Figure 2.1: Given target discrete curvature per vertex on a triangle mesh, we describe a method to efficiently and robustly compute a discrete conformal deformation of the mesh's metric to satisfy this prescription. Its basis is the mathematical foundation described in Gu et al. [2018b]; Springborn [2020]. The figure illustrates several flattenings (parametrizations over the plane) obtained from computed metrics that are flat except at a few cone singularities (green and red points). The parametrization is visualized by means of an adaptive grid texture (see Section 2.7); red paths indicate transitions in the parametrization due to the prescribed cones. Conformal scale distortion is indicated by shading, from blue over white to red. The method supports closed surfaces as well as surfaces with boundary. By prescribing the geodesic curvature along the boundary, alignment of the parametrization with the boundary can be enforced. On the right, closed surfaces are cut (along the black curves), turning them into surfaces with boundary, which enables us to enforce parametrization alignment also along the cut curves.

Contributions

- We present an efficient and practical algorithm for computing discrete conformal metrics with prescribed Gaussian curvature at interior vertices and prescribed geodesic curvature along the boundary.
- We extend the closed-surface theory to surfaces with boundary via a double-cover construction and introduce algorithmic machinery to reliably maintain symmetric intrinsic Delaunay triangulations in the presence of Delaunay-critical configurations.
- We introduce practical improvements for convergence, accuracy, and robustness; analyze numerical limits (including arithmetic precision effects); and show how to derive continuous maps/flattenings from the computed discrete metrics.

Organization

We review background on discrete conformal metrics and hyperbolic ideal Delaunay triangulations (Section 2.3), then describe the main algorithm and implementation details (Section 2.4). We present the extension to surfaces with boundary (Section 2.5) and the construction of continuous maps from the discrete metric (Section 2.6), and conclude with an evaluation and numerical study (Section 2.7). Supporting lemmas and additional discussion appear at the end of the chapter.

2.1 Introduction

Computing discrete metrics with prescribed angles on meshes is a problem closely related to surface parametrization and quadrangulation, which is of interest in many geometric settings. Despite many years of efforts, only a few techniques for mesh parametrization provide theoretical guarantees, commonly derived from the same source: discrete harmonic mappings with convex boundary, based on Tutte's embedding theorem [Floater 1997].

Recent exciting advances concerning the theory of discrete metric uniformization [Gu et al. 2018b; Springborn 2020] provide a solid foundation for a much needed addition to this spectrum of methods. They enable the computation of discrete metrics with arbitrary prescribed discrete curvature at vertices, as long as the discrete Gauss-Bonnet theorem is respected. In particular, this allows to compute, with guarantees, flat metrics or almost-everywhere flat cone metrics with prescribed curvatures at cones—an essential component of global parametrization and quadrangulation algorithms. Guarantees follow from a reduction to an unconstrained convex optimization problem. However, compared to Tutte's method, the numerics involved are far more complex, in particular due to nonlinearity and large scale distortions inherent in conformal maps.

We present an efficient numerical algorithm based on these new theoretical ideas, extend it to

support surfaces with boundary, and explore its practical performance, focusing on robustness.

Problem Summary.. To define the problem more precisely, consider a manifold triangle mesh $\mathcal{T}$, possibly with boundary. For a given discrete metric on M, i.e., an assignment of lengths to its edges that satisfy the triangle inequality, we can compute inner angles of triangles.

Let Θ_i be the total angle (the sum of incident inner angles) at vertex v_i, and κ_i its angle deficit, defined as $2\pi - \Theta_i$ for interior vertices and $\pi - \Theta_i$ for boundary vertices. This quantity κ_i can be viewed as the discrete Gaussian curvature if v_i is an interior vertex and the geodesic curvature of the boundary if v_i is on the boundary. Given *target* curvatures $\hat{\kappa}_i$ (respecting the discrete Gauss-Bonnet theorem) our goal is to compute edge lengths that exhibit exactly these curvatures. Flattenings, i.e., mesh parametrizations over the plane, are a special case corresponding to prescribing $\kappa_i = 0$ in the interior [Ben-Chen et al. 2008]. Seamless maps for quadrilateral remeshing are obtained by prescribing $\hat{\kappa}_i = k_i \frac{\pi}{2}$ with $k_i \in \mathbb{Z}$ [Campen et al. 2019; Myles and Zorin 2012].

Approach. As shown in [Gu et al. 2018b; Springborn 2020], a discrete metric realizing target curvatures $\hat{\kappa}_i$ always exists, *if retriangulation of the surface is allowed.* When restricting to metrics *discretely conformally equivalent* to a given original metric, this metric is unique (up to scale) and can be computed by minimizing a convex function.

While the latter property has been exploited before for practical parametrization purposes [Springborn et al. 2008], the assumption of a fixed triangulation restricts the space of target curvatures that can be realized by a conformally equivalent metric. For example, a vertex v_i of valence k cannot, under any (Euclidean) metric, exhibit a discrete curvature $\kappa_i \leqslant (2-k)\pi$, because inner angles are bounded by π. As a consequence, the resulting discrete metric's edge lengths violate the triangle inequality in some places. This limitation can be remedied by allowing changes to the triangulation of the input surface.

More concretely, the main requirement for triangulation changes needed to enable this is that at all times the triangulation remains an intrinsic Delaunay triangulation. This leads to a natural algorithm [Sun et al. 2015] in the spirit of kinetic data structures [Basch et al. 1999], which, however, requires the determination of the exact sequence of all individual Delaunay-critical events during the metric computation process.

Contributions. In this paper we describe an efficient and practical algorithm, performing triangulation changes with greater flexibility, enabled by the theoretical connection to hyperbolic metrics established by [Gu et al. 2018b; Springborn 2020]. While this theory is developed for closed surfaces, in practice many, if not most, relevant applications involve surfaces with boundaries. These cases can be reduced to the closed surface case by creating a surface double, but a number of algorithmic issues need to be addressed to reliably maintain symmetric intrinsic Delaunay triangulations in such cases. We introduce a number of additional improvements to the basic algorithm, to speed up convergence and increase accuracy and robustness. We furthermore perform extensive evaluations, with a focus on numerical aspects such as the effect of varying arithmetic accuracy. Numerical behavior of the algorithm is of critical relevance as conformal metrics and maps can unavoidably exhibit very large ranges of scales.

We discuss the relevant background in Section 2.3. An implementation of the main ideas, with particular attention to practical aspects is described in Section 2.4. Generalization to surfaces with boundary is presented in Section 2.5, followed by the construction of a surface mapping from the discrete metric in Section 2.6, and concluded by the evaluation of the algorithm in Section 2.7.

2.2 Related Work

The problem of computing conformally equivalent metrics or, by implication, conformal maps of discrete surfaces, has been considered in a variety of works before. As there is no useful natural

notion of conformality in the discrete (non-smooth) setting, a range of discrete counterparts of the continuous concept of conformality have been proposed and used.

Static Triangulation. Prominent examples of works addressing the computation of conformal metrics or conformal maps on discrete surfaces, based on various definitions of discrete conformality, while considering the triangulation fixed are based on least-squares formulations [Lévy et al. 2002; Desbrun et al. 2002], vertex scaling formulations [Springborn et al. 2008; Ben-Chen et al. 2008; Sawhney and Crane 2017; Jin et al. 2007; Soliman et al. 2018a], circle patterns [Kharevych et al. 2006a], or formulations based on holomorphic one-forms [Gu and Yau 2003].

Dynamic Triangulation. A fixed triangulation restricts the space of metrics that can be achieved. By adjusting the triangulation depending on the prescribed target curvature, this limitation can be remedied. Two systematic approaches have been proposed to that end, both conceptually considering a continuous metric evolution from initial state to target state. [Luo 2004] proposes to adjust the triangulation by an intrinsic edge flip whenever an edge becomes triangle inequality critical (Figure 2.2 left). Implementation variants are described in [Campen and Zorin 2017b,a; Campen et al. 2019]. Differently, [Gu et al. 2018b,a; Springborn 2020] effectively consider the case of flipping an edge when it becomes Delaunay-critical, i.e., when four vertices become co-circular (Figure 2.2 right). Surfaces with boundary in this context are addressed in [Sun et al. 2015] using a double cover approach, reducing this case to the case without boundary. A correspondence map between the original triangulation and the modified triangulation can be kept track of by means of an overlay data structure [Fisher et al. 2007].

In concurrent work, [Gillespie et al. 2021] make use of the same theoretical results we use here and describe an algorithm that conceptually is very close to our core algorithm in Section 2.4. Main differences of our work are (i) a number of important details in the optimization procedure as described in Section 2.4, (ii) special combinatorial handling of symmetry in the double surface

used to support meshes with boundary, and (iii) extensive evaluation in particular of numerical limits and numerical precision effects. In comparison, [Gillespie et al. 2021] propose a more lightweight data structure (than [Fisher et al. 2007] that we use) to keep track of the mesh overlay, and additionally consider the case of spherical parametrization.

2.3 Background

We begin by considering the case of surfaces *without* boundary, i.e., we are given a closed manifold triangle mesh $M = (V, E, F)$. The case of surfaces with boundary can be reduced to the closed surface case with an additional symmetry structure, which we address in detail in Section 2.5.

The mesh M is equipped with an input metric defined by edge lengths $\boldsymbol{\ell} : E \to \mathbb{R}^{>0}$, satisfying the triangle inequality.

2.3.1 Conformal Equivalence

A *conformally equivalent* discrete metric, for a fixed triangulation M, is defined by means of *logarithmic scale factors* $\boldsymbol{u} : V \to \mathbb{R}$ associated with vertices $V = (v_1, \ldots, v_n)$, by defining new edge lengths as

$$\ell_{ij}(\boldsymbol{u}) = \ell_{ij}\, e^{\frac{u_i + u_j}{2}} \tag{2.1}$$

per edge e_{ij} [Luo 2004]. Given per-vertex target angles $\hat{\Theta}_i$ a conformally equivalent metric with these angles is characterized by, for all i:

$$g_i(\boldsymbol{u}) := \hat{\Theta}_i - \Theta_i(\boldsymbol{u}) = \hat{\Theta}_i - \sum_{T_{ijk}} \alpha^i_{jk}(\boldsymbol{u}) = 0, \tag{2.2}$$

where the inner angle $\alpha^i_{jk}(\boldsymbol{u})$ is computed under the metric defined by $\boldsymbol{u}$ via Equation (2.1), i.e., from edge lengths $\tilde{\boldsymbol{\ell}} = \boldsymbol{\ell}(\boldsymbol{u})$. We use shorthands $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha}(\boldsymbol{u})$ and $\tilde{\boldsymbol{\ell}} = \boldsymbol{\ell}(\boldsymbol{u})$ for these scale factor

dependent quantities in the following.

It is known that $\boldsymbol{g}(\boldsymbol{u}) = (g_1(\boldsymbol{u}), \ldots, g_n(\boldsymbol{u}))$ is the gradient of a twice-differentiable convex function [Springborn et al. 2008]. Hence, one may obtain factors $\boldsymbol{u}$ satisfying Equation (2.2) using (second-order) convex optimization methods, starting from arbitrary initializations (e.g. $\boldsymbol{u} \equiv \mathbf{0}$). This is true, however, only as long as there is a solution for which $\boldsymbol{u}$ stays in the feasible region $\Omega_M \subset \mathbb{R}^n$ where $\boldsymbol{\ell}(\boldsymbol{u})$ respects the triangle inequality for each triangle T_{ijk}; otherwise it does not define a Euclidean surface metric on M.

2.3.2 Dynamic Triangulation

The feasible region Ω_M can be altered by adjusting the triangulation dynamically *during* the evolution of $\boldsymbol{u}$ from $\mathbf{0}$ towards $\boldsymbol{u}^*$.

Note that a change of triangulation is possible without *intrinsically* changing the surface. M together with given edge lengths defines a surface S_V with a metric which is flat everywhere except at V. There are many triangulations (besides M) with vertices V and their own associated edge lengths, defining the same surface S_V (cf. [Sharp et al. 2019]); hence the differentiation between M and S_V. In particular, an edge flip replacing a pair of triangles (T_{ijk}, T_{jim}) sharing an edge e_{ij}, with triangles (T_{kim}, T_{mjk}) sharing edge e_{km} can be performed without intrinsically changing the surface S_V, by setting the length of the new edge e_{km} to the length of the diagonal of the planar quadrilateral obtained by unfolding T_{ijk}, T_{jim} [Fisher et al. 2007]. This is referred to as *intrinsic flip*.

Delaunay Flips. [Gu et al. 2018b; Springborn 2020] prove a remarkable fact: the convex energy can be extended to all of the space $\mathbb{R}^n$ of scale factors $\boldsymbol{u}$ defined at vertices, *if a particular change of triangulation is allowed.* Specifically, the triangulation is modified so that it stays (intrinsically) Delaunay at all times as $\boldsymbol{u}$ evolves. More specifically, whenever the Delaunay condition is violated

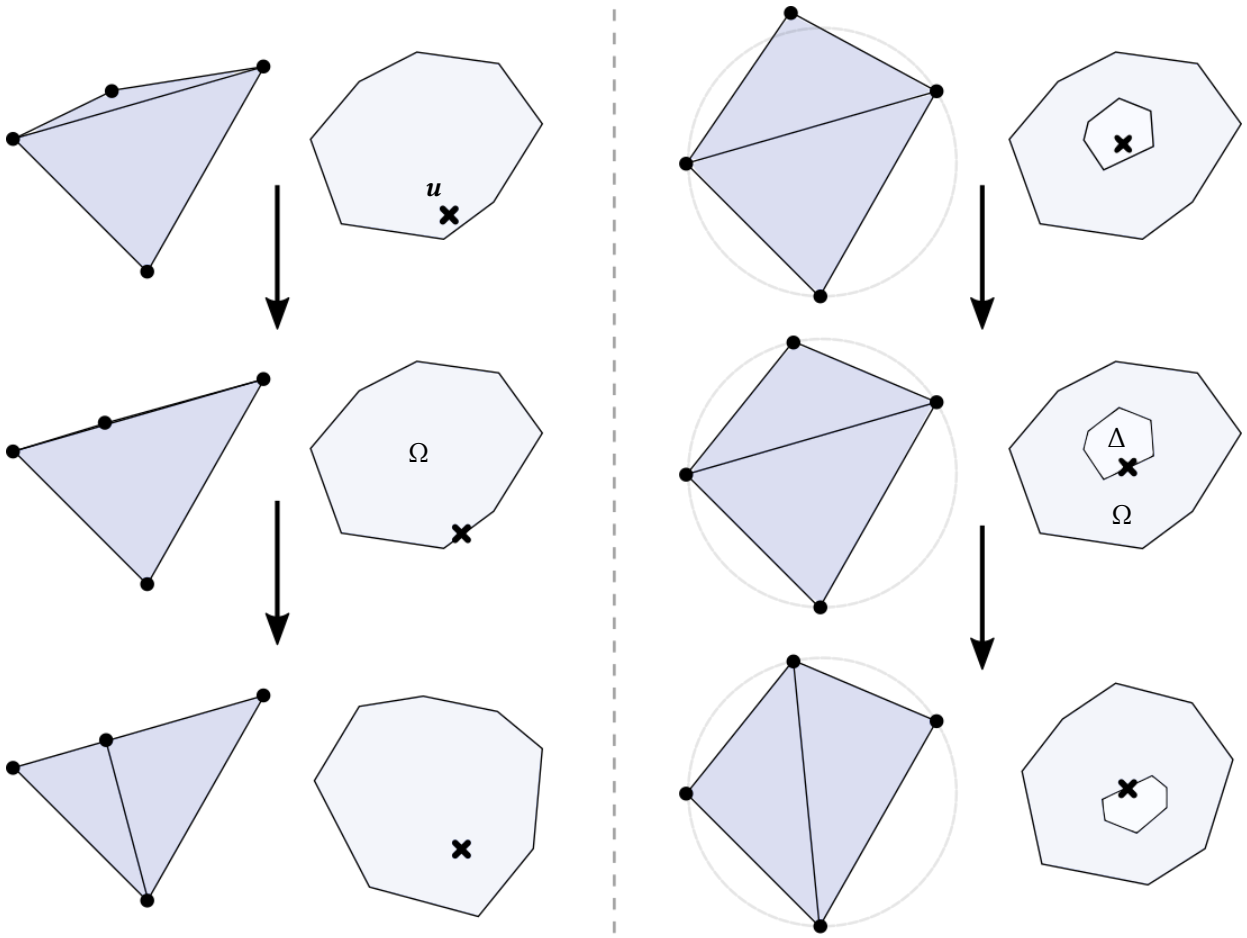


Figure 2.2: Left: flip-on-degeneration. Right: flip-on-Delaunay-violation. Alongside a conceptual illustration of the valid region Ω (light blue) and Delaunay region Δ (white) is shown (cf. Section 2.3.2), containing the current point $\boldsymbol{u}$ (cross mark) and changing due to the flip.

as a result of a change in $\boldsymbol{u}$, a flip is performed to maintain the Delaunay property. As the resulting energy is a globally convex function, it can be minimized by an unconstrained Newton method, and the resulting choice of $\boldsymbol{u}$ satisfies (2.2) with respect to the resulting triangulation.

Definition 2.1 (Intrinsic Delaunay)**.** A triangulation is intrinsically Delaunay if the angles of two triangles T_{ijk} and T_{jim} opposite a common edge e_{ij} satisfy the Delaunay condition:

$$\tilde{\alpha}_{ij}^{k} + \tilde{\alpha}_{ij}^{m} \leqslant \pi \tag{2.3}$$

or equivalently $\cos \tilde{\alpha}_{ij}^{k} + \cos \tilde{\alpha}_{ij}^{m} \geqslant 0$. Expressed directly in terms of edge lengths this condition is

equivalent to

$$\frac{\tilde{\ell}_{jk}^2 + \tilde{\ell}_{ki}^2 - \tilde{\ell}_{ij}^2}{\tilde{\ell}_{jk}\tilde{\ell}_{ki}} + \frac{\tilde{\ell}_{jm}^2 + \tilde{\ell}_{mi}^2 - \tilde{\ell}_{ij}^2}{\tilde{\ell}_{jm}\tilde{\ell}_{mi}} \geqslant 0. \tag{2.4}$$

This latter version of the Delaunay condition is particularly important for our construction.

Generically (iff these weak inequalities hold strictly), the intrinsic Delaunay triangulation is unique, but for special configurations (four or more intrinsically co-circular vertices resulting in equality in Equation (2.4)) it is not.

For a given triangulation M, the *Penner cell* $\Delta_M \subset \mathbb{R}^n$ denotes the set of scale factors $\boldsymbol{u}$ for which M, along with the modified metric defined by $\boldsymbol{u}$, is intrinsic Delaunay. Clearly, $\Delta_M \subset \Omega_M$, and when $\boldsymbol{u} \in \partial\Delta_M$ the Delaunay triangulation is not unique. Whenever $\boldsymbol{u}$ reaches the boundary of Δ_M, we can switch to another Delaunay triangulation M' by means of an intrinsic flip, thereby changing the region (from Δ_M to $\Delta_{M'}$), enabling $\boldsymbol{u}$ to evolve further, see Figure 2.2. The cells Δ_M form a partition of $\mathbb{R}^n$ [Gu et al. 2018b].

Such changes of scale factors together with intrinsic Delaunay flips lead to the following generalized notion of discrete conformal equivalence of two metrics [Gu et al. 2018b]:

Definition 2.2 (Discrete Conformal Equivalence)**.** Two metrics $(M_1, \boldsymbol{\ell}_1)$ and $(M_m, \boldsymbol{\ell}_m)$ are *discretely conformally equivalent*, if there is a sequence of meshes with the same vertex set, $(M_s, \boldsymbol{\ell}_s)$, $s = 1, \ldots, m$, such that, for each s, M_s is an intrinsic Delaunay triangulation for the metric $\boldsymbol{\ell}_s$ and either

- $(M_s, \boldsymbol{\ell}_s)$ and $(M_{s+1}, \boldsymbol{\ell}_{s+1})$ are different metrics with the same triangulation (i.e., $M_s = M_{s+1}$) and the edge lengths are related by Equation (2.1) for a choice of $\boldsymbol{u}_s : V \to \mathbb{R}$.
- $(M_s, \boldsymbol{\ell}_s)$ and $(M_{s+1}, \boldsymbol{\ell}_{s+1})$ are different Delaunay triangulations for the same metric.

Degeneration Flips. The alternative of performing a triangulation change only when $\boldsymbol{u}$ reaches

the boundary $\partial\Omega$ of the currently feasible region was considered by [Luo 2004]. This occurs when a triangle becomes a degenerate cap. An intrinsic flip of this triangle's longest edge yields a non-degenerate triangulation, effectively changing the valid region Ω such that $\boldsymbol{u}$ lies strictly in its interior. Figure 2.2 left illustrates this. An implementation of this approach is described and applied in [Campen and Zorin 2017b]. Open theoretical questions remain regarding the finiteness of the flip sequence and the sound handling of simultaneous adjacent degeneracies.

At first sight, the approach based on maintaining an intrinsic Delaunay triangulation may seem less efficient in comparison. Due to $\Delta \subset \Omega$, at least as many, but often many more cells Δ need to be traversed. Practically, this suggests a large number of small steps between flips in the process of optimizing $\boldsymbol{u}$, compared to, e.g., the use of (less frequent) degeneration flips, and much smaller steps compared to typical unconstrained optimization.

Remarkably, however, this Delaunay approach permits an implementation that is in general more efficient and more robust (see Section 2.7.2 for a comparison). As we will see, exploiting a relation to *hyperbolic* Delaunay triangulation, arbitrarily large steps can be made, beyond Δ and even beyond Ω (unconstrained by Euclidean triangle inequalities). Flips can be performed collectively *after the fact* and in *arbitrary order*. This is detailed in Section 2.3.4.

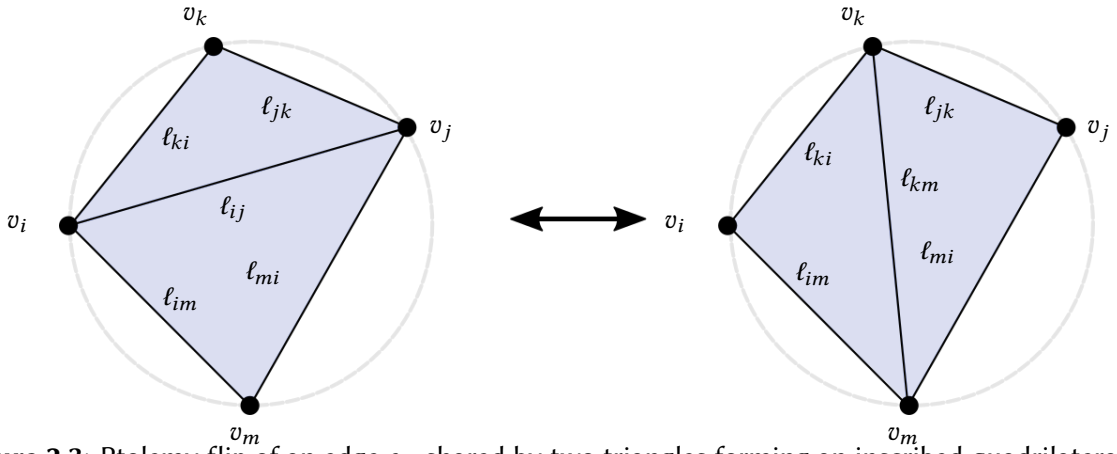


Figure 2.3: Ptolemy flip of an edge e_{ij} shared by two triangles forming an inscribed quadrilateral, i.e., a Delaunay-critical edge.

2.3.3 Evolution Step

Assume we are given a triangulation M that is intrinsic Delaunay under the metric defined by some $\boldsymbol{u}_\vdash$. Consider an evolution of $\boldsymbol{u}$ from $\boldsymbol{u}_\vdash$ to $\boldsymbol{u}_\dashv$, e.g., linear:

$$\boldsymbol{u}(t) = (1-t)\boldsymbol{u}_\vdash + t\boldsymbol{u}_\dashv, \;\; t \in [0,1].$$

As we move along the interval $[0,1]$, whenever four vertices forming triangles T_{ijk} and T_{jim} become co-circular under the metric defined by $\boldsymbol{\ell}(\boldsymbol{u}(t))$, an intrinsic flip of edge e_{ij} is performed. Due to the special configuration (the two triangles forming an *inscribed* quadrilateral, see Figure 2.3) the length that the new edge e_{km} needs to take can be computed following Ptolemy's theorem as

$$\tilde{\ell}_{km} = \frac{1}{\tilde{\ell}_{ij}}(\tilde{\ell}_{jk}\tilde{\ell}_{im} + \tilde{\ell}_{ki}\tilde{\ell}_{mj}), \tag{2.5}$$

where we use $\tilde{\boldsymbol{\ell}}$ as a shorthand for $\boldsymbol{\ell}(\boldsymbol{u}(t))$. For $\tilde{\ell}_{km} = \ell_{km}(u(t)) = \ell_{km}\, e^{\frac{u_k+u_m}{2}}$ to take this value for the current $\boldsymbol{u}(t)$, we need to set:

$$\ell_{km} := \frac{1}{\ell_{ij}}(\ell_{jk}\ell_{im} + \ell_{ki}\ell_{mj}). \tag{2.6}$$

Notice that this is Ptolemy's formula, Equation (2.5), applied to the *original* metric, as all scale factors cancel. In other words: applying the formula in the current ($\boldsymbol{u}(t)$-scaled) metric $\tilde{\boldsymbol{\ell}}$ is equivalent to applying it in the original metric $\boldsymbol{\ell}$, followed by scaling. Remarkably, this holds *even though the vertices are not co-circular under the original metric* in general. Moreover, the edge lengths $\boldsymbol{\ell}$ set in this way may *not even satisfy the triangle inequality*. This is no issue, though, as certainly the relevant scaled lengths $\tilde{\boldsymbol{\ell}} = \boldsymbol{\ell}(\boldsymbol{u}(t))$ do, by construction.

It was shown that the number of flip events along the path is finite [Wu 2014], which means that after a finite number of flips we will obtain the triangulation and edge length assignment needed

for the target $\boldsymbol{u}(1) = \boldsymbol{u}_\dashv$.

One practical downside of this procedure, in which the necessary flips along the evolution path are detected and performed one-by-one sequentially [Sun et al. 2015], is that it requires solving precisely for the sequence of flips. An alternative approach, whose correctness can be shown based on an interpretation of the involved edge lengths as defining hyperbolic metrics instead of Euclidean metrics, improves on this.

2.3.4 Hyperbolic Metric Approach

Instead of moving t along the interval $[0, 1]$, determining the sequence of flip events and executing them in order, let us directly consider $t = 1$. The initial triangulation M may not be Delaunay under $\boldsymbol{u}(1)$, and the edge lengths $\boldsymbol{\ell}(\boldsymbol{u}(1))$ may not even respect the triangle inequality. Nevertheless, we can test each edge for violation of the Delaunay criterion using Equation (2.4) applied to $\boldsymbol{\ell}(\boldsymbol{u}(1))$, and incrementally flip (using Equation (2.6)) all violating edges in arbitrary order following the classical flip algorithm until a Delaunay triangulation is reached [Bobenko and Springborn 2007]. While in case of triangle inequality violations this criterion lacks the geometric justification via Equation (2.3) (the involved quantities are no longer (cotangents of) Euclidean angles), this algorithm nevertheless succeeds.

Hyperbolic Delaunay.. The reasons for applicability of Equation (2.4) and use of Equation (2.6) are direct consequences of an elegant correspondence between hyperbolic and conformal metric structures used in the proofs of [Gu et al. 2018b; Springborn 2020] and introduced in [Rivin 1994], given by mapping edge lengths to Penner coordinates of a hyperbolic metric, and Euclidean triangulations to ideal triangulations. Detailed explanations can be found in these papers and an overview given in [Crane 2020, §5, §6]. We go into more detail in Section 2.6, as this relation is important when the conformal metric is used to establish a conformal map, for purposes of

evaluation of the map at arbitrary points. Here we just present a proposition summarizing the aspect of this theory relevant to our algorithm.

Proposition 1. *Suppose lengths $\tilde{\boldsymbol{\ell}}$ (possibly not satisfying triangle inequality) are assigned to edges in a triangle mesh M, conformally equivalent to a set of Euclidean metric lengths $\boldsymbol{\ell}$. If the flip algorithm is applied to $\tilde{\boldsymbol{\ell}}$, with the Delaunay criterion in algebraic form* (2.4) *used to determine which edges need to be flipped, and the Ptolemy formula* (2.6) *used for length updates, the algorithm produces a triangulation M' with lengths $\tilde{\boldsymbol{\ell}}'$ that satisfy the triangle inequality. This triangulation is intrinsic Delaunay. Moreover, the discrete metric defined by $(M', \tilde{\boldsymbol{\ell}}')$ is discrete conformally equivalent to $(M, \boldsymbol{\ell})$.*

In summary, instead of performing flips following an expensive-to-compute sequence required to maintain a valid Euclidean metric on triangles at all times, the algorithm performs the flips in arbitrary order, yielding edge lengths $\tilde{\boldsymbol{\ell}}$ satisfying the triangle inequality only in the end. This version of the flip algorithm is referred to as *Weeks algorithm* [Weeks 1993].

These observations ensure that whenever we modify scale factors $\boldsymbol{u}$ while computing the conformal metric, the flip algorithm can be used to recover a Delaunay triangulation, which can then be used to evaluate the value of the convex function we need to minimize, its gradient, and its Hessian.

2.4 Algorithm

Using this background, we can now formulate an efficient algorithm for the computation of a conformally equivalent metric, respecting prescribed target angles $\hat{\boldsymbol{\Theta}}$. The algorithm, spelled out in algorithm 1, is based on a standard Newton's method with line search, but incorporates several important details and modifications.

Algorithm 1: FindConformalMetric

Input : triangle mesh $M = (V, E, F)$, closed, manifold, edge lengths $\boldsymbol{\ell} > 0$ satisfying triangle inequality, target angles $\hat{\boldsymbol{\Theta}} > 0$ respecting Gauss-Bonnet

Output: triangle mesh $M' = (V, E', F')$, edge lengths $\tilde{\boldsymbol{\ell}} > 0$ satisfying triangle inequality, such that $\|\boldsymbol{\Theta}_{(M', \tilde{\boldsymbol{\ell}})} - \hat{\boldsymbol{\Theta}}\|_\infty \leqslant \varepsilon_{\text{tol}}$

Function *FindConformalMetric*$(M, \boldsymbol{\ell}, \hat{\boldsymbol{\Theta}})$**:**
- $\boldsymbol{u} \leftarrow \mathbf{0}$
- $(M, \boldsymbol{\ell}) \leftarrow$ MakeDelaunay$(M, \boldsymbol{\ell}, \boldsymbol{u})$
- **while not** *converged*$(M, \boldsymbol{\ell}, \boldsymbol{u})$ **do**
  - $\boldsymbol{g} \leftarrow g(M, \boldsymbol{\ell}, \boldsymbol{u})$ // gradient
  - $H \leftarrow H(M, \boldsymbol{\ell}, \boldsymbol{u})$ // Hessian
  - $\boldsymbol{d} \leftarrow solve(H\boldsymbol{d} = -\boldsymbol{g})$ // Newton direction
  - $(M, \boldsymbol{\ell}, \boldsymbol{u}) \leftarrow$ LineSearch$(M, \boldsymbol{\ell}, \boldsymbol{u}, \boldsymbol{d})$ // Newton step
- $\tilde{\boldsymbol{\ell}} \leftarrow$ScaleConformally$(M, \boldsymbol{\ell}, \boldsymbol{u})$
- **return** $(M, \tilde{\boldsymbol{\ell}})$

Function *LineSearch*$(M, \boldsymbol{\ell}, \boldsymbol{u}, \boldsymbol{d})$**:**
- $(M_1, \boldsymbol{\ell}_1) \leftarrow$ MakeDelaunayPtolemy$(M, \boldsymbol{\ell}, \boldsymbol{u} + \boldsymbol{d})$
- $(M_{1/2}, \boldsymbol{\ell}_{1/2}) \leftarrow$ MakeDelaunayPtolemy$(M, \boldsymbol{\ell}, \boldsymbol{u} + \frac{1}{2}\boldsymbol{d})$
- **if** $\frac{1}{2}\big(\boldsymbol{d}^\intercal g(M_1, \boldsymbol{\ell}_1, \boldsymbol{u}+\boldsymbol{d}) + \boldsymbol{d}^\intercal g(M_{1/2}, \boldsymbol{\ell}_{1/2}, \boldsymbol{u}+\frac{1}{2}\boldsymbol{d})\big) \leqslant \alpha \boldsymbol{d}^\intercal g(M, \boldsymbol{\ell}, \boldsymbol{u})$ **then** // Armijo-like condition
  - **return** $(M_1, \boldsymbol{\ell}_1, \boldsymbol{u} + \boldsymbol{d})$ // full step
- **while** *true* **do** // line search
  - $(M, \boldsymbol{\ell}) \leftarrow$ MakeDelaunayPtolemy$(M, \boldsymbol{\ell}, \boldsymbol{u} + \boldsymbol{d})$
  - **if** $\boldsymbol{d}^\intercal g(M, \boldsymbol{\ell}, \boldsymbol{u} + \boldsymbol{d}) \leqslant 0$ **then** // Equation (2.7)
    - **return** $(M, \boldsymbol{\ell}, \boldsymbol{u} + \boldsymbol{d})$
  - $\boldsymbol{d} \leftarrow \frac{1}{2}\boldsymbol{d}$ // backtracking

Function *converged*$(M, \boldsymbol{\ell}, \boldsymbol{u})$**:**
- **return** $\|\hat{\boldsymbol{\Theta}} - \Theta(M, \tilde{\boldsymbol{\ell}})\|_\infty \leqslant \varepsilon_{\text{tol}}$

Function g $(M, \boldsymbol{\ell}, \boldsymbol{u})$**:**
- **return** $\hat{\boldsymbol{\Theta}} - \Theta(M, \tilde{\boldsymbol{\ell}})$ // Equation (2.2)

Function H $(M, \boldsymbol{\ell}, \boldsymbol{u})$**:**
- **return** CotanLaplacian$(M, \tilde{\boldsymbol{\ell}})$

Function $\Theta(M, \boldsymbol{\ell}, \boldsymbol{u})$**:** // angle computation
- **for** $v_i \in V$ **do** // Equation (2.2)
  - $\Theta_i \leftarrow \sum_{T_{ijk} \in M'} \arccos\left((\tilde{\ell}_{ij}^2 + \tilde{\ell}_{ki}^2 - \tilde{\ell}_{jk}^2)/(2\tilde{\ell}_{ij}\tilde{\ell}_{ki})\right)$
- **return** $(\Theta_0, \ldots, \Theta_n)$

Function *MakeDelaunayPtolemy*$(M, \boldsymbol{\ell}, \boldsymbol{u})$**:**
- **while** *NonDelaunay*$(M, \boldsymbol{\ell}, \boldsymbol{u}, e_{ij})$ *for any edge* e_{ij} **do**
  - $(M, \boldsymbol{\ell}) \leftarrow$ PtolemyFlip$(M, \boldsymbol{\ell}, e_{ij})$
- **return** $(M, \boldsymbol{\ell})$

Function *NonDelaunay*$(M, \boldsymbol{\ell}, \boldsymbol{u}, e_{ij})$**:**
- **return** $(\tilde{\ell}_{jk}^2 + \tilde{\ell}_{ki}^2 - \tilde{\ell}_{ij}^2)/(\tilde{\ell}_{jk}\tilde{\ell}_{ki})$
  $+ (\tilde{\ell}_{jm}^2 + \tilde{\ell}_{mi}^2 - \tilde{\ell}_{ij}^2)/(\tilde{\ell}_{jm}\tilde{\ell}_{mi}) < 0$ // Equation (2.4)

Function *PtolemyFlip*$(M, \boldsymbol{\ell}, e_{ij})$**:**
- $M \leftarrow$ Flip(M, e_{ij})
- $\ell_{km} \leftarrow (\ell_{jk}\ell_{im} + \ell_{ki}\ell_{mj})/\ell_{ij}$ // Equation (2.6)
- **return** $(M, \boldsymbol{\ell})$

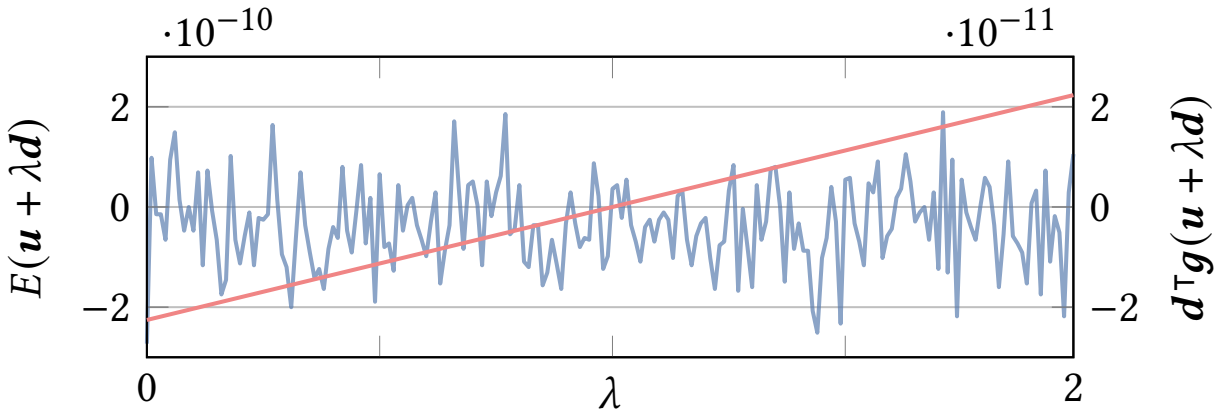


Figure 2.4: Energy (blue; mean (20202.12) subtracted) and projected gradient (red) along a descent direction $\boldsymbol{d}$. Notice that the numerical noise in the energy computation dominates the actual change in energy, making it less suitable to be a measure of progress in the line search. By contrast, the sign of the projected gradient (red) can be determined much more precisely.

Delaunay

Initially, if M is not already intrinsically Delaunay, it is turned into a Delaunay mesh using standard intrinsic edge flips. Then, whenever $\boldsymbol{u}$ is updated (during the line search), before the gradient and Hessian are evaluated the triangulation is turned into an intrinsic Delaunay triangulation with respect to the metric defined by $\boldsymbol{u}$ using Weeks flip algorithm—now using the Ptolemy length computation rule from Equation (2.6).

Energy-free Line Search

The function $E(\boldsymbol{u})$ that needs to be minimized is known explicitly [Springborn et al. 2008]:

$$E(\boldsymbol{u}) = \sum_{T_{ijk}} \Big(2f(\tilde{\lambda}_{ij}, \tilde{\lambda}_{jk}, \tilde{\lambda}_{ki}) - \pi(u_i + u_j + u_k) \Big) + \hat{\boldsymbol{\Theta}}^\mathsf{T}\boldsymbol{u},$$

where $\tilde{\lambda}_{ij} = 2\log \ell_{ij} + u_i + u_j$ and f is a per-triangle function involving Milnor's Lobachevsky function [Springborn et al. 2008, Eq. (8)]. The gradient of $E(\boldsymbol{u})$ is $\boldsymbol{g}(\boldsymbol{u}) = \hat{\boldsymbol{\Theta}} - \boldsymbol{\Theta}(\boldsymbol{u})$ (Equation (2.2)) and its Hessian $H(\boldsymbol{u})$ simply is the well-known positive semi-definite cotan-Laplacian in terms

of the scaled angles $\boldsymbol{\alpha}(\boldsymbol{u})$.

The obvious approach is to use the standard Newton's method with backtracking line search, using $E(\boldsymbol{u})$, $\boldsymbol{g}(\boldsymbol{u})$, $H(\boldsymbol{u})$ (cf. [Gillespie et al. 2021]). However, computing the energy $E(\boldsymbol{u})$, in particular evaluating the Lobachevsky function, presents numerical challenges, and efficient Chebyshev-polynomial approximations, like the one used in the implementation of [Springborn et al. 2008], may not yield sufficient accuracy, while incurring additional computational overhead. We observe that the energy can be very flat along the search direction, so using the decrease of energy evaluated this way as a criterion in the line search may lead to the algorithm stalling due to numerical noise (see Figure 2.4). This is particularly problematic in cases requiring high conformal distortion or if we want to compute the conformal metric with high precision, as needed for instance to derive an implied conformal map (cf. Section 2.7).

The evaluation of the gradient and Hessian, both of which are simple functions of angles (not involving the Lobachevsky function), by contrast, is more efficient and numerically robust than the energy itself (see Figure 2.4). Fortunately, we are able to formulate our algorithm such that it relies on $\boldsymbol{g}(\boldsymbol{u})$ and $H(\boldsymbol{u})$ only. This is possible for the following reason: As $E(\boldsymbol{u})$ is convex, it is also convex along the search direction $\boldsymbol{d}$, i.e., $E(\boldsymbol{u} + \lambda\boldsymbol{d})$ for fixed $\boldsymbol{u}$ and $\boldsymbol{d}$ is convex in the step size λ. Therefore its derivative

$$\frac{\partial}{\partial\lambda} E(\boldsymbol{u} + \lambda\boldsymbol{d}) = \boldsymbol{d}^\mathsf{T}\boldsymbol{g}(\boldsymbol{u} + \lambda\boldsymbol{d}), \tag{2.7}$$

i.e., the gradient's projection onto the search direction, has at most one zero. Hence, if we require that the step size λ is selected in the line search such that $\boldsymbol{d}^\mathsf{T}\boldsymbol{g}(\boldsymbol{u} + \lambda\boldsymbol{d}) \leqslant 0$, this guarantees that $E(\boldsymbol{u})$ decreases, without the need for checking the function value itself.

Note that avoiding energy evaluation precludes the use of standard *sufficient decrease conditions* (most commonly, Armijo condition) in the line search. However, a simple backtracking search,

starting with $\lambda = 1$, for a point along the search direction with negative projected gradient, ensures that the Newton step, when it is less than one, is always in the range $[\lambda_m/2, \lambda_m]$, where λ_m is the function's (unknown) minimum point along the search line. One can show that this is sufficient for convergence by following the standard analysis of Newton's method with inexact line search. However, this is, in general, not sufficient to guarantee that the algorithm converges *quadratically*. An additional Armijo-like condition (the first termination condition in the line search in algorithm 1; we use $\alpha = 0.1$, with a meaning similar to the Armijo condition constant) yields a more consistent quadratic behavior. The practical effect of this additional termination condition is small in most cases (most commonly, the reduction in the number of iterations on our test datasets is around 1-2). A detailed analysis of convergence of the proposed energy-free method can be found in [Zorin 2021].

Termination

The accuracy with which the target angles $\hat{\boldsymbol{\Theta}}$ can be matched of course depends (in a non-trivial manner) on the precision of the real number representation. If tolerance ε_{tol} is chosen too low relative to this, algorithm 1 may never terminate. For practical purposes therefore additional stopping criteria can be taken into account: an upper bound on the number of Newton steps and the number of line search halvings, a lower bound on the Newton decrement $\boldsymbol{d}^{\mathsf{T}}\boldsymbol{g}(M, \boldsymbol{\ell}, \boldsymbol{u})$. Information about the practically achievable accuracy can be found in Section 2.7.3.

Additional Performance Heuristic

In particularly challenging cases, the gradient direction and in particular its magnitude can be rapidly varying. The line search loop may then have to be executed many times before a valid step size is found, causing many redundant edge flips. One additional line search heuristic that proved beneficial in this regard is a *gradient norm decrease* condition. Specifically, as a stopping condition for the line search we require that, in addition to $\boldsymbol{d}^{\mathsf{T}}\boldsymbol{g} < 0$, the norm of the gradient

$\|\boldsymbol{g}\|$ decreases. Only if this additional condition forces the step size below a given threshold (we use 10^{-10}), the condition is lifted for one step, allowing the gradient to grow, so as to not hamper convergence.

Overlay Mesh. An embedding of the (by edge flips) modified mesh in the original mesh can be maintained by using a mesh overlay data structure. Towards the algorithm it behaves like a mesh, but internally it keeps track of the *overlay* of both meshes, updating it whenever an edge is flipped. [Fisher et al. 2007] propose to represent it explicitly by means of a polygon mesh data structure, [Gillespie et al. 2021] propose a more lightweight implicit representation by normal coordinates. We found the overhead of even the explicit structure to be benign (e.g., on average 11% added time cost on the example cases from Figure 2.12).

2.5 Boundaries

So far, we assumed that M is a closed surface. For a surface with boundary, the problem can be reduced to the case of closed surfaces by gluing a mirrored copy to the surface along the boundary, turning it into a closed surface with an obvious (reflectional) symmetry. A strategy of this kind is also used in [Sun et al. 2015] and [Gillespie et al. 2021].

However, the initial symmetry of the setting may be disturbed when applying algorithm 1. Due to numerical inaccuracies, the values $\boldsymbol{u}$ may diverge on the two copies; application of a standard Delaunay flip algorithm is further complicated by the presence of stably cocircular configurations, as we discuss below. Therefore we describe a version of this surface double cover approach that explicitly imposes and maintains symmetry, on the numerical as well as the combinatorial level, by construction.

2.5.1 Double Cover

Let the input surface be N. Its double cover is constructed as follows:

1. we attach a mirrored copy N' of the input mesh N along the boundary (merging boundary vertices and edges), as illustrated below, yielding a closed mesh M,

2. we transfer the edge lengths $\boldsymbol{\ell}$ and the target curvatures κ_i of interior vertices v_i from N to N',

3. we prescribe $\hat{\Theta}_i = 2\pi - 2\hat{\kappa}_i$ at each (former) boundary vertex v_i, where $\hat{\kappa}_i$ is the target discrete geodesic boundary curvature at vertex v_i.

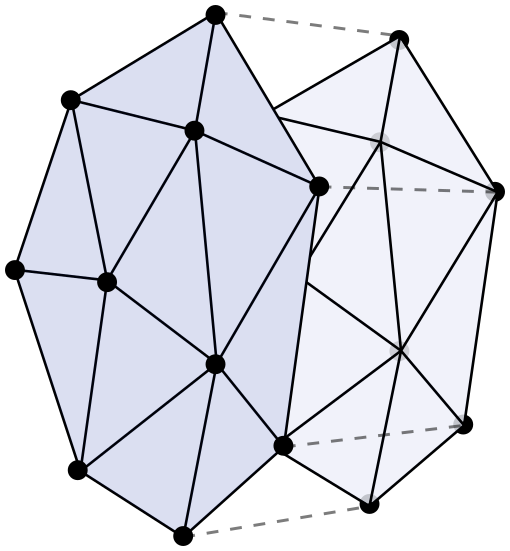


Figure 2.5: Double cover construction: a surface with boundary (left) is mirrored along its boundary to create a closed surface (right) with reflectional symmetry.

The double cover mesh M built this way exhibits an obvious reflectional symmetry, i.e., there is a map R with $R^2 = I$ that takes vertices to vertices, edges to edges, and faces to faces. It maps an element in the interior of N to its copy in N' and vice versa; on the merged (former) boundary vertices and edges, R is the identity.

Conformal Metric Symmetry. Due to symmetry (i.e., invariance with respect to R) of the mesh M, the metric $\boldsymbol{\ell}$, and the target angles $\hat{\boldsymbol{\Theta}}$, the symmetrically initialized factors $\boldsymbol{u}$ will (in theory, up to numerical round-off error) remain symmetric after each iteration of the optimization process. This can be seen by observing that the function $E(\boldsymbol{u})$ is the sum of per-triangle terms $E_T(\boldsymbol{u}_T)$, where $\boldsymbol{u}_T$ is the restriction of $\boldsymbol{u}$ to vertices of the triangle T. Given the above symmetry, its gradient $\boldsymbol{g}(\boldsymbol{u}) = \nabla_{\boldsymbol{u}} E$ therefore is invariant with respect to R. Consequently, if we cut the mesh along the symmetry line in the end, so as to discard one copy, a boundary vertex v_i will have exactly half the prescribed angle, $\frac{1}{2}\hat{\Theta}_i = \pi - \hat{\kappa}_i$, and therefore exhibit a discrete geodesic boundary curvature of $\hat{\kappa}_i$, just as intended.

Tufted Double Cover. The fact that $\boldsymbol{u}$ (and thus all vertex-associated attributes) are supposed to evolve symmetrically implies that we can use a *tufted* double cover as in [Sharp and Crane 2020], with the unknown scale factors $\boldsymbol{u}$ shared between the two symmetric halves of M, to reduce the number of variables (and to impose perfect symmetry on the numerical level). This does not mean, however, that computations could trivially be restricted entirely to one half of the double cover only: edge flips may, and commonly will, create edges and faces spanning both halves of the double cover, crossing the symmetry line.

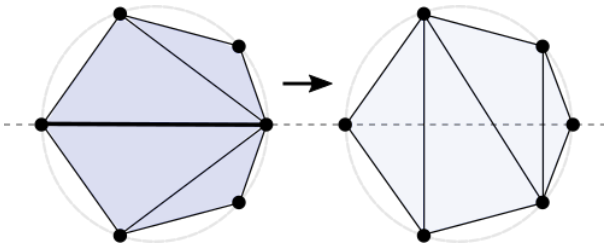


Figure 2.6: Edge flips across the symmetry line can lead to triangulations that are no longer combinatorially symmetric.

Edge flips across the symmetry line can lead to triangulations that are no longer combinatorially symmetric, as depicted in the inset. Unless special care is taken, this can increase the chance of numerical inaccuracies causing divergence from geometric symmetry. Furthermore, such cases

contain co-circular vertex configurations that are *stable*, i.e., for the given triangulation, due to the symmetry of $\boldsymbol{u}$, these remain co-circular *independent of the evolution of* $\boldsymbol{u}$. An example is the diagonal edge on the right in the inset. As in this case, numerical evaluation of the Delaunay condition results in an essentially random choice of the result, in order to avoid potentially infinite flip sequences of Delaunay-critical edges, we instead perform special flips at the symmetry line, maintaining perfect combinatorial symmetry by construction, as detailed in the next section. Our method explicitly identifies these stably cocircular configurations and ensures that Delaunay flips are never applied to these, even if they appear to be slightly non-Delaunay due to numerical inaccuracies. In addition, having a symmetric Delaunay mesh for the final configuration can simplify the extraction of the resulting metric or map for the original surface with boundary.

2.5.2 Symmetric Meshes

Our goal is to rigorously determine which edge flip cases can occur in a symmetric mesh, in particular at the symmetry line, so as to ensure all special cases are correctly handled in our method. To that end, we begin by making precise the general notion of *combinatorially symmetric* polygon mesh. In this, rather than using edges, we use *halfedges*, each associated with a unique face (or a boundary loop, which can be treated exactly like a face). Specifically, each edge corresponds to two halfedges.

Definition 2.3 (Combinatorial Mesh)**.** A combinatorial polygon mesh is a triple $(H, \mathcal{N}, O)$ of a set of halfedges H, a bijective function $\mathcal{N} : H \to N$ (*next-halfedge* function), and a bijective function O (*opposite-halfedge* function) with the property

$$O^2(h) = h;\ O(h) \neq h \tag{2.8}$$

i.e., all orbits of O have size 2.

This definition is quite general which is important for maintaining intrinsic Delaunay triangulations: e.g., it allows for vertices of valence 1, polygons glued to themselves, etc., all of which are possible configurations in these triangulations.

Definition 2.4 (Mesh Elements)**.** Define the bijective circulator function $C : H \to H$ to be $\mathcal{N}^{-1}(\mathcal{O}(h))$. Then the mesh has the following implied elements:

- *Faces* are the orbits of the next-halfedge function $\mathcal{N}$.
- *Vertices* are the orbits of the circulator function C.
- *Edges* are the orbits of the opposite-halfedge function $\mathcal{O}$.

Collectively we refer to them as (mesh) *elements*. A halfedge *belongs* to an element if it is part of the respective orbit.

A *mesh with boundary* is a mesh with a subset of its faces marked as boundary loops. The halfedges of these loops form the set H^{bnd} of *boundary halfedges*.

Definition 2.5 (Reflection Map)**.** A reflection map $R : H \to H$ for a mesh $(H, \mathcal{N}, \mathcal{O})$ is an involution ($R^2 = I$) defined on the set of halfedges: each halfedge is mapped either to itself, or forms a reflection pair with a distinct halfedge. It is required to satisfy the following conditions:

1. preservation of $\mathcal{O}$ relation: $\mathcal{O}(R(h)) = R(\mathcal{O}(h))$,
2. inversion of $\mathcal{N}$ relation: $\mathcal{N}(R(h)) = R(\mathcal{N}^{-1}(h))$,
3. preservation of boundary: $h \in H^{bnd} \iff R(h) \in H^{bnd}$.

Note that conditions (1) and (2) correspond to the properties of continuity and orientation-reversal of continuous reflection maps [Panozzo et al. 2012]. They imply that R maps orbits of $\mathcal{N}$, of $\mathcal{O}$, and, as a consequence, of C, to orbits of these functions, i.e., it is well-defined for faces, edges,

and vertices (via $R(x) = x' \iff R(h) \in x'$ for any $h \in x$). Furthermore, because $R^2 = I$, all orbits of R have length 1 or 2, whether it acts on halfedges, faces, edges, or vertices. This implies the following partitioning.

Proposition 2 (Halfedge Sets)**.** *H can be partitioned into disjoint sets H^1, H^2, H^s so that the following conditions are satisfied:*

- $h \in H^s \iff R(h) = h$;
- $h \in H^1 \iff R(h) \in H^2$;
- *for any face or edge x, either all belonging halfedges are in H^1, or all in H^2, or x is fixed by R (i.e. $R(x) = x$).*

This leads to the following partitioning of the sets of edges and faces, where $e = (h, h')$, $f = (h_0, \dots h_{m-1})$ denote the orbits of belonging halfedges:

- $e \in E_i \iff h, h' \in H^i$, $i = 1, 2$
- $e \in E^\perp \iff h, h' \in H^s$
- $e \in E^\parallel \iff h = R(h')$
- $f \in F_i \iff h_0 \in H^i$, $i = 1, 2$
- $f \in F^s \iff R(h_0) \in f$

The set $E^\perp$ is the set of edges (perpendicularly) crossing the symmetry line between two halves of a symmetric mesh mapped to each other (see Figure 2.7 right); the set $E^\parallel$ is the set of edges on the symmetry line; F^s is the set of faces that cross the symmetry line, and are mapped by the symmetry map to themselves. For additional details, see Section 2.8.

Using this terminology, our double cover construction from Section 2.5.1 can be described formally in terms of combinatorial structure of the mesh (see Section 2.9). Initially we have $E^{\perp} = \varnothing$ and $F^s = \varnothing$, i.e., no element crosses the symmetry line (the former boundary). $E^{\parallel}$ contains the edges lying *on* the symmetry line, i.e., those for whose halfedges the O relation was adjusted to glue the two copies. This initially simple situation can change, however, when edge flips are performed on the double cover mesh.

2.5.3 Symmetric Flips

When an edge e in a symmetric mesh $M = (H, N, O, R)$ shall be flipped, the edge $R(e)$ needs to be flipped as well (unless $R(e) = e$), so as to be able to maintain a symmetric mesh. The simultaneous flip of e and $R(e)$ (as well as the single flip of e if $R(e) = e$) is referred to as *symmetric flip*. As discussed in Section 2.5.1, in the algorithm from Section 2.4 the metric evolves symmetrically. This implies that whenever the algorithm intends to flip an edge e, it simultaneously intends to flip $R(e)$ as well. The algorithm is therefore compatible with the restriction to symmetric flips.

While for an edge $e \in E_i$ with incident faces $f, g \in F_i$ the process is obvious, special care needs to be taken when elements from $E^{\parallel}$, $E^{\perp}$, or F^s are involved. We will exhaustively distinguish different types of symmetric flips based on the membership of the involved edges and faces in these sets.

Flip Types. For a triple (f_a, e, f_b) of an edge e with incident faces f_a, f_b, the triple of labels denoting their set memberships, e.g., $(1, \parallel, 2)$, is called flip *type* of the edge e.

Consistent Flip Types. We say that an edge has a *consistent* flip type, if this particular triple may occur in a symmetric mesh. For instance, $(1, \perp, 1)$ is not a consistent type, as edges from $E^{\perp}$ necessarily have incident faces from F^s by definition.

Proposition 3 (Section 2.8) helps to reduce the possible set to the following six possibilities, up to a $1 \leftrightarrow 2$ exchange. It is easy to construct examples proving that all of them are consistent, i.e., may occur in a symmetric mesh:

- Edge in E^1: Set 1a: $(1, 1, 1)$, $(1, 1, s)$ Set 1b: $(s, 1, s)$
- Edge in $E^{\|}$: Set 2a: $(1, \|, 2)$ Set 2b: $(s, \|, s)$
- Edge in $E^{\perp}$: Set 3: $(s, \perp, s)$

Relevant Flip Types. Among these types, only four are also *relevant*; Following Proposition 4 (Section 2.8), types of the form $(s, \|, s)$ and $(s, 1, s)$ in the sets 1b and 2b are necessarily associated with edges that satisfy the Delaunay condition Equation (2.4) irrespective of the choice of lengths of edges involved. These are not relevant for the purpose of the algorithm from Section 2.4, which exclusively flips non-Delaunay edges. This leaves only sets 1a, 2a, and 3 for further consideration.

Triangles and Quadrilaterals. A flip of type $(1, 1, s)$ leads to a pair of triangles in F^s that together form a quadrilateral which is inscribed, i.e., the four vertices are intrinsically co-circular (Figure 2.7 center). Remarkably, this statement holds regardless of metric, as long as it is symmetric, i.e., invariant with respect to R. Instead of randomly choosing a diagonal splitting this quadrilateral into two triangles, we explicitly represent it as a quadrilateral face. This avoids violating the symmetry by the diagonal, which, e.g., would complicate recovering the surface with boundary after the conformal metric is computed, and avoids potential issues such as sequences of flips caused by numerically nearly co-circular points.

Faces in F^s can therefore be triangular or quadrilateral. We accordingly partition $F^s = F^t \cup F^q$, and based on this distinguish t-versions and q-versions of flip types involving the label s. This yields a total of seven types that are consistent and relevant.

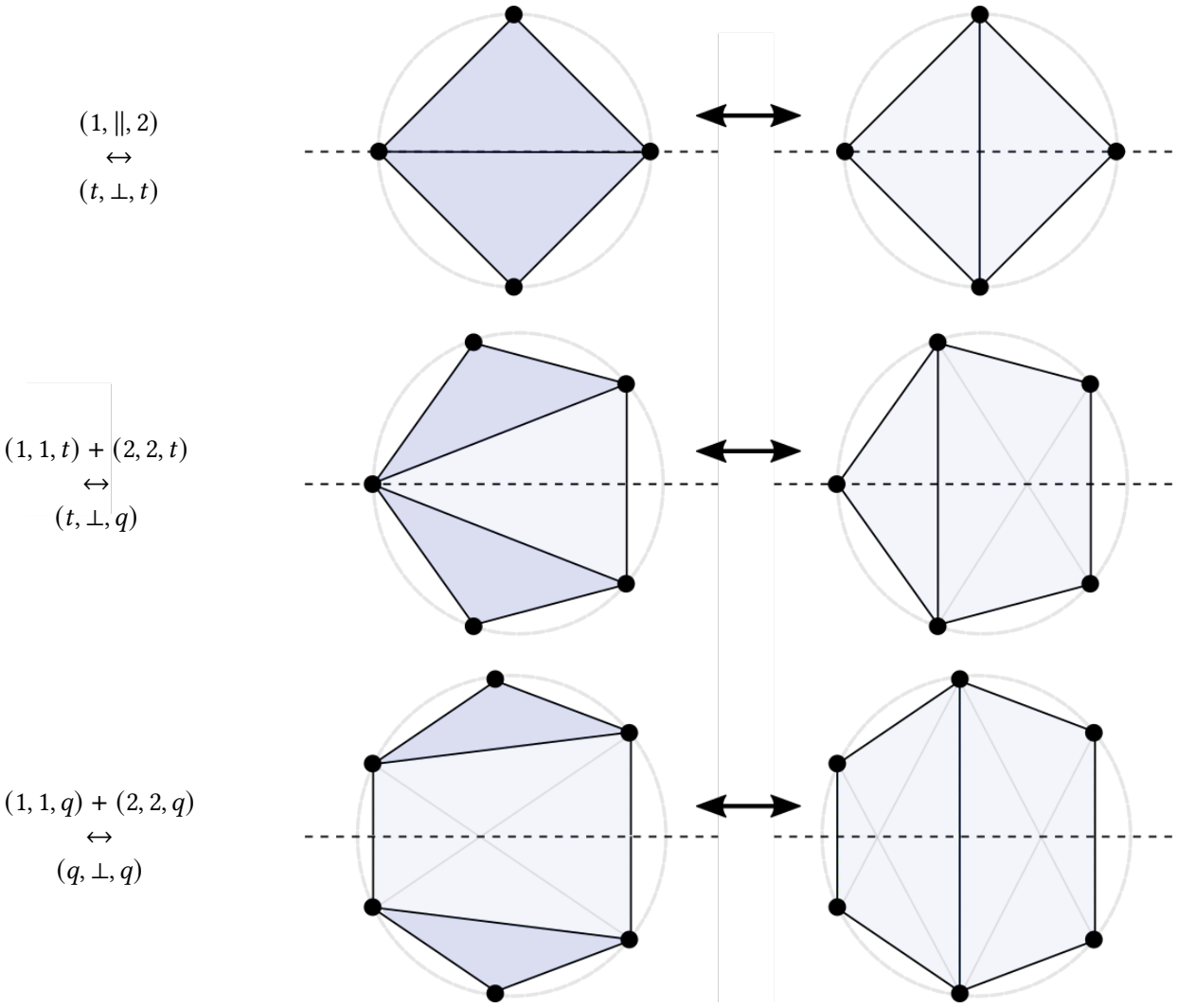


Figure 2.7: Symmetric edge flips involving faces from F^s (light blue), crossing the symmetry line (dashed). Faces from F^1 and F^2 are colored dark blue. The configurations are shown with co-circular vertices, though combinatorially flips can be performed in any state. Note that the light blue quads' vertices, however, are necessarily co-circular by symmetry, regardless of metric.

Six of these seven flip types form three pairs of mutually inverse flips, while one is self-inverse.

We can thus succinctly summarize :

1. $(1, 1, 1) + (2, 2, 2) \leftrightarrow (1, 1, 1) + (2, 2, 2)$;
2. $(1, \|, 2) \leftrightarrow (t, \perp, t)$;
3. $(1, 1, t) + (2, 2, t) \leftrightarrow (t, \perp, q)$;

4. $(1, 1, q) + (2, 2, q) \leftrightarrow (q, \perp, q)$.

Case (1) is the standard case of flipping a configuration not involving the symmetry line. (2), (3), and (4) are the special cases crossing the symmetry line; they are illustrated in Figure 2.7. Table 2.1 details the combinatorial changes to be performed on the symmetric mesh so as to execute these symmetric flips. In terms of implementation, it thus simply comes down to initially labeling the edges and faces of the double cover, updating the labels when flipping edges, and using one of these special case rules whenever a label other than 1 or 2 is involved in a flip.

2.5.4 Symmetric Metric

To be able to apply algorithm 1 to such symmetric meshes to compute a symmetric conformal metric, what is left to clarify is how to deal with quadrilateral faces.

Delaunay Criterion. For edges with two incident triangles, the Delaunay check needed for the algorithm is standard, via Equation (2.4). If one of the incident faces is a quad, due to symmetry it, regardless of the metric, is an inscribed trapezoid. As a consequence, whichever way we (virtually) split it into triangles we get the same angles opposite any of its edges. Hence, we may perform the Delaunay check assuming arbitrary virtual diagonals in the quads.

Gradient and Hessian. For the same reason, the computation of gradient $g(\boldsymbol{u})$ and Hessian $H(\boldsymbol{u})$ can be performed based on arbitrary diagonals; the choice does not affect the result [Springborn 2020].

Ptolemy Formula. Note that each of the edges created by symmetric flips involving quads (Figure 2.7) can also be obtained by a sequence of edge flips involving triangles (and split quads) only. In this way the length of such edges can be computed using (multiple instances of) the standard Ptolemy formula Equation (2.6). As there are only four types of flips involving quads, one

can conveniently derive closed form expressions for these cases in advance, rather than actually performing these sequences for each flip. Note that each quad needs to store its diagonal length to enable these computations.

2.5.5 Restriction to Single Cover

Once algorithm 1 has terminated and the desired conformal metric has been computed, we finally need to discard half of the double cover: we need to cut the symmetric surface along the line of symmetry. While initially the entire symmetry line is formed by a sequence of mesh edges, this may no longer be the case due to flips (unless an overlay is used), namely whenever F^s and $E^\perp$ are not empty in the end. One simply needs to split all edges from $E^\perp$ at their midpoint, and split the triangles and quads from F^s by connecting these inserted split vertices.

2.6 Continuous maps from discrete metrics

The algorithms described in previous sections deal exclusively with discrete metric definitions, i.e., assignments of edge lengths to edges of a mesh. If mesh connectivity does not change, an affine map from the initial mesh triangles T to the final mesh triangles $\tilde{T}$ can be easily inferred from the lengths. However, as pointed out in [Springborn et al. 2008], a natural map is actually a projective map between triangles, which, in addition to mapping the original lengths to conformally deformed ones, also maps the circumcircle of T to the circumcircle of $\tilde{T}$. While for fixed connectivity this yields only a moderate improvement in, e.g., texture quality, for changing connectivity the map definition is more relevant.

While for the discrete algorithm itself we only needed a simple-to-formulate (although surprising) fact that Weeks flip algorithm can be used to obtain an intrinsically Delaunay mesh even if the triangle inequality is violated at intermediate steps, defining maps between the original mesh

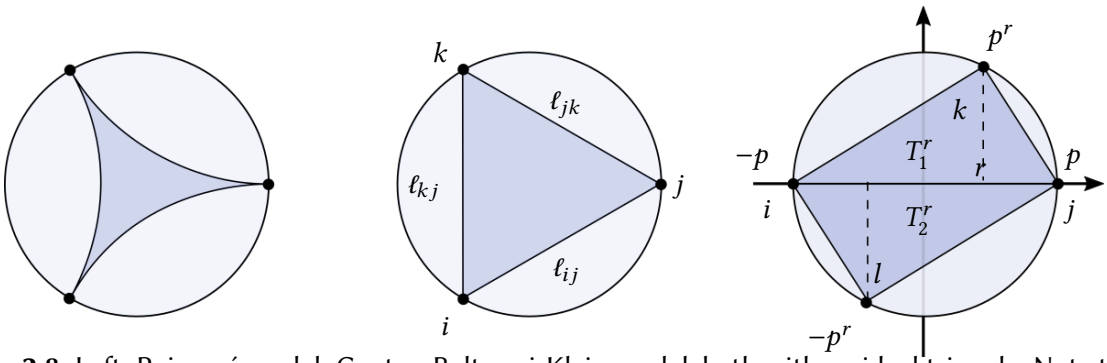


Figure 2.8: Left: Poincaré model. Center: Beltrami-Klein model, both with an ideal triangle. Note that in the Beltrami-Klein model it forms a Euclidean triangle. Right: Two-triangle chart.

points and the final (e.g. flat) mesh points requires a more in-depth exposition of the underlying theory.

Our goal in this section is to define a map $f : |M| \to |M'|$, from the original to the final (e.g. conformally flattened) mesh, more specifically, mapping formulas of the form $(w'_l, w'_m, w'_m; T') = f(w_i, w_j, w_k; T)$, where (w_i, w_j, w_k) are barycentric coordinates of a point on the input triangle T_{ijk} in M and (w'_l, w'_m, w'_n) is the corresponding point on a triangle T'_{lmn} in mesh M'.

2.6.1 Cusped Hyperbolic Metric on Meshes

The central idea of the theory in [Gu et al. 2018b] and several other papers dealing with related problems is a construction of a hyperbolic metric corresponding to a Euclidean metric $\boldsymbol{\ell}$ which is invariant to conformal scale factors $\boldsymbol{u}$; in this context the lengths $\boldsymbol{\ell}$ are referred to as *Penner coordinates* of the hyperbolic metric.

Conformal deformations of $\boldsymbol{\ell}$ do not change this hyperbolic metric, and flips define just different triangulations of a fixed surface. The update of Penner coordinates for an edge flip using the Ptolemy formula Equation (2.6) happens to produce a mesh that is isometric in the hyperbolic metric to the mesh before the flip. Next, we discuss the hyperbolic metric definition and isometric retriangulation in this metric in more detail.

Beltrami-Klein Model.. We use the *Beltrami-Klein* hyperbolic plane model. The model represents the hyperbolic plane H^2 as the interior of a unit disk, with points of the boundary of the disk being points at infinity in the hyperbolic metric. These points (which are not a part of the hyperbolic plane, but play an important role in the model) are called *ideal points*. The model has the following properties.

- Lines are segments connecting points on the boundary.
- Given two distinct points p and q in the disk, the unique Euclidean straight line connecting them intersects the disk's boundary at two ideal points, a and b; label them so that the points are, in order, a, p, q, b along the line. The hyperbolic distance between p and q then is:
$$d_H(p,q) = \frac{1}{2}\log\frac{|aq|\,|pb|}{|ap|\,|qb|}$$
- *Isometries of the hyperbolic plane correspond to projective transformations preserving the unit disk.*
- An isometry is defined uniquely by specifying images of three points on the boundary of the disk (ideal points). There is an isometry mapping any three ideal points to any other three ideal points. We denote such projective maps $P[T \to T']$ where T and T' are triples of points on the unit disk (Figure 2.8 center).
- While angles are not preserved, if a line is a diameter, perpendicular lines are also perpendicular to it in the model.

Defining the Hyperbolic Metric.. For a mesh M with vertices excluded, the hyperbolic metric is defined by mapping each mesh triangle, with edge lengths given by $\boldsymbol{\ell}$, to a similar Euclidean triangle inscribed in a unit disk, and using the Beltrami-Klein model to define the hyperbolic

distances inside the triangle. Under this hyperbolic metric the triangles are ideal, with vertices at infinity (referred to as *cusped*, for reasons more obvious in the Poincaré model, see Figure 2.8 left). Furthermore they are all congruent, because there is a hyperbolic isometry, a projective circumcircle-preserving map, mapping one triangle to the other.

Note however, that unlike the case of finite triangles, the identification of sides of ideal triangles that are adjacent in M is not unique: because the sides are infinitely long, one can slide them along each other isometrically. The natural gluing defined by identifying points that correspond in the disk model picks one such isometric identification. One can show that if Penner coordinates $\boldsymbol{\ell}$ and $\tilde{\boldsymbol{\ell}}$ are related by a set of conformal scale factors $\boldsymbol{u}$, the resulting gluing between adjacent ideal triangles is the same, i.e., they define the same metric.

This allows a convenient definition of *two-triangle isometric charts* (Figure 2.8, right) for this metric, which provide most of what we need for defining our maps f across edge flips.

Two-Triangle Charts.. Consider two adjacent triangles T_{ijk} and T_{jil} sharing edge e_{ij}, and five Penner coordinates ℓ_{ij}, ℓ_{jk}, ℓ_{ki}, ℓ_{il}, ℓ_{lj}. For a single triangle, Penner coordinates can be changed arbitrarily using conformal deformations. Note however, that there are only four conformal scale factors u_i, u_j, u_k, u_l involved when mapping two adjacent triangles, so the five lengths cannot be chosen completely arbitrarily. The invariant that is preserved under these remappings is the *cross-ratio* $c_{ij} = (\ell_{jk}\ell_{il})/(\ell_{jl}\ell_{ki})$. Cross-ratio assignments to edges (*shear coordinates*) actually are in one-to-one correspondence with choices of cusped hyperbolic metrics on a fixed mesh.

We are thus free to choose the conformal scale factors u_i, u_j, u_k, u_l so that the following conditions are satisfied: (1) edge e_{ij} is mapped to the diameter $(-p, p)$, with $p = (1, 0)$, on the horizontal coordinate axis; (2) vertices k and l are mapped to antipodal points $p^r = (r, \sqrt{1-r^2})$ and $-p^r$ on the circle. It is easy to check that the four scale factors are uniquely defined by these conditions, with r equal to $(1 - c_{ij})/(1 + c_{ij})$. Notice that $c_{ij} > 0$, thus $r \in (-1, 1)$, regardless of any triangle

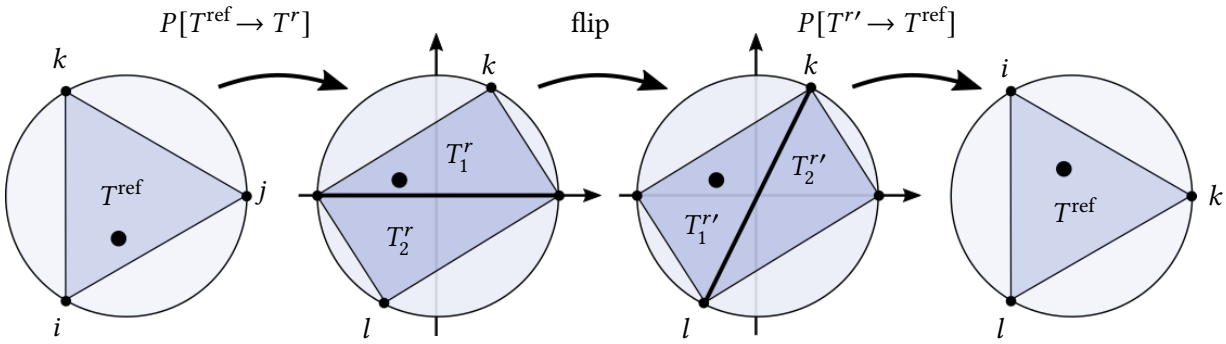


Figure 2.9: Mapping a point through a single flip via a two-triangle chart.

inequality condition. We denote these two chart triangles T_1^r and T_2^r.

Thus, an isometric atlas can be constructed for the whole mesh, by mapping each triangle pair to a chart as described above. This gives us the necessary tools to define the map f.

Mapping Across a Flip.. Let M_k be a mesh obtained after applying a sequence of k flips to M, and M_{k+1} a mesh obtained by flipping a single further edge e_{ij} shared by triangles $T_1 = T_{ijk}$ and $T_2 = T_{jil}$ as above. Each mesh has length assignments $\boldsymbol{\ell}_k$, but as these are guaranteed to satisfy triangle inequalities only at certain steps k where the Delaunay condition is satisfied, these are best viewed as Penner coordinates for a hyperbolic metric.

As barycentric coordinates are not invariant with respect to projective maps, we need to choose a reference triangle for barycentric representation (w_i, w_j, w_k). We use an equilateral reference triangle T^{ref}, with vertices q_0, q_1, q_2, with $q_s = (\cos 2s\pi/3, \sin 2s\pi/3)$ for any triangle T_1 of M_k, see Figure 2.9 left.

In the two-triangle chart, T_1 and T_2 are mapped to T_1^r and T_2^r. After the flip in the chart, the new chart triangles, corresponding to triangles $T_1' = T_{jkl}$ and $T_2' = T_{ilk}$ are $T_1^{r\prime} = (p, p^r, -p^r)$ and $T_2^{r\prime} = (-p, -p^r, p^r)$, see Figure 2.9 center. If the image of the point (w_i, w_j, w_k) in the chart belongs to triangle T_1' then the map $(w_i, w_j, w_k) \to (w_i', w_j', w_k')$ is obtained as the composition of

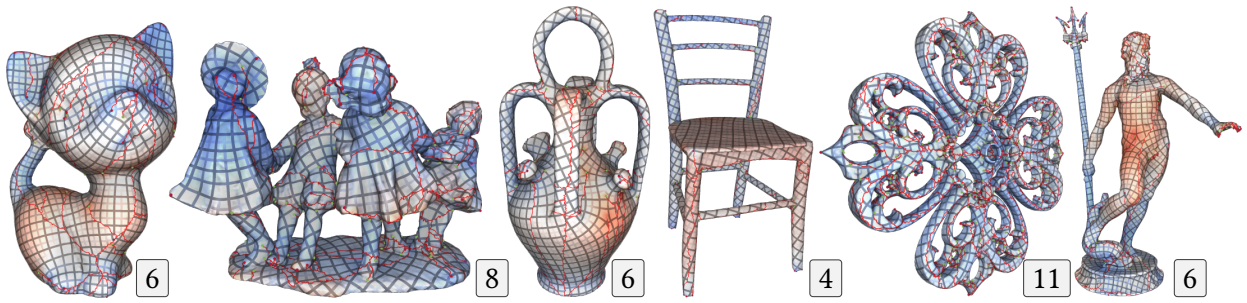


Figure 2.10: Visualization of conformal maps, implied by conformal cone metrics, on some of the closed models with angle prescriptions from the dataset of [Myles et al. 2014]. The numbers indicate the scale range (difference of maximal and minimal conformal (natural) logarithmic scale factor $\boldsymbol{u}$) for each model. Cones are marked by red and green dots; texture jumps due to cones are marked red. The textured map and scale visualization follow the description from Section 2.7.

circumcircle-preserving projective maps:

$$\begin{aligned}(w'_j, w'_k, w'_l) &= f(w_i, w_j, w_k) = \\ &\Big(P[T_1^{r\prime} \to T^{\mathrm{ref}}] \circ B \circ P[T^{\mathrm{ref}} \to T_1^r]\Big)(w_i, w_j, w_k)\end{aligned} \tag{2.9}$$

where B is the matrix converting barycentric coordinates on T_1^r to barycentric coordinates on $T_2^{r\prime}$. The expression is similar in the case when the image of the point in the chart lands in $T_2^{r\prime}$. The circumcircle-preserving projective maps P can be computed in barycentric coordinates using the following formula:

$$P(w_i, w_j, w_k) = (w_i S_i, w_j S_j, w_k S_k)/(w_i S_i + w_j S_j + w_k S_k)$$

with $S_i = \frac{\ell_{ij}\ell_{ki}\tilde{\ell}_{jk}}{\tilde{\ell}_{ij}\tilde{\ell}_{ki}\ell_{jk}}$, where $\boldsymbol{\ell}$ are lengths of the source, and $\tilde{\boldsymbol{\ell}}$ are lengths of the target triangle.

2.7 Evaluation

We have implemented algorithm 1 (with support for boundaries following Section 2.5) in C++. Our goal is to assess how well this theoretically sound method performs practically. While by default we use standard double precision floating point numbers, the optional use of extended

precision arithmetics in our implementation allows us to assess to what extent potential convergence issues are related to finite precision or other problems, as detailed in Sections 2.7.3 and 2.7.4. We find that, as conformal maps can easily involve a very large range of scales across a mesh, for certain challenging settings the use of extended precision arithmetics can be essential to yield results of adequate quality.

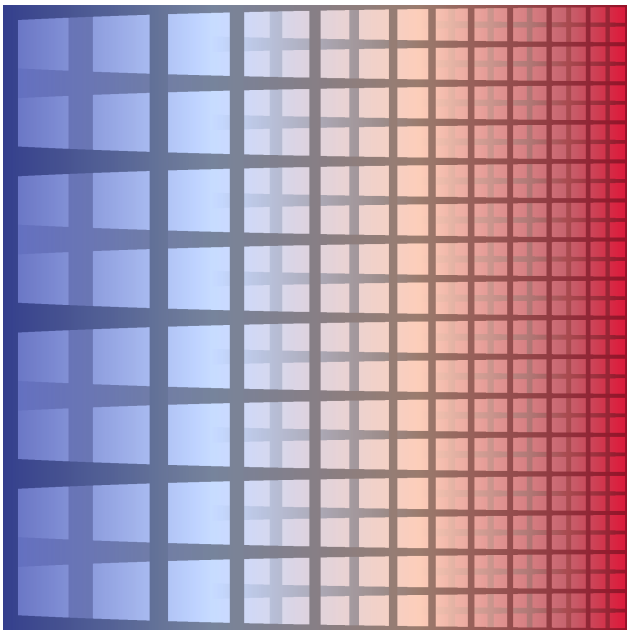


Figure 2.11: Hierarchical grid texture used for visualizing conformal metric distortion. The grid density adapts to the local scale factor of the conformal map.

2.7.1 Validation

Closed Surfaces

A dataset of mesh models together with angle prescriptions $\hat{\Theta} > 0$ (based on cones of cross fields) has been released with [Myles et al. 2014]. We applied our implementation to the closed models from this dataset. The angle error decay in the course of the algorithm on these cases is visualized in Figure 2.12. Some of the models with the resulting conformal map are visualized in Figure 2.10. We observe that the models reach angle accuracy of 10^{-10} in less than 15 Newton iterations. The final achievable accuracy varies and is correlated with the range of scale factors in the final mesh

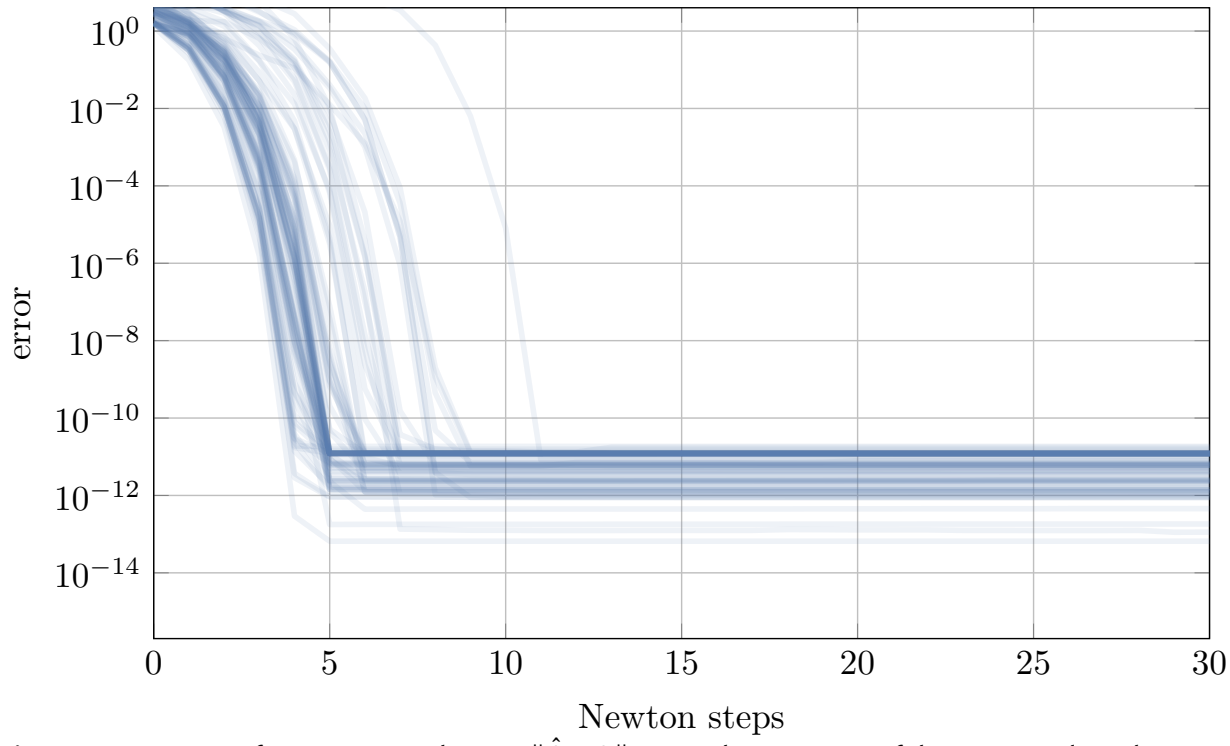


Figure 2.12: Decay of maximum angle error $\|\hat{\Theta} - \Theta\|_\infty$ over the iterations of the Newton algorithm. Each graph represents one of the closed-surface instances from the dataset of [Myles et al. 2014].

(cf. Figure 2.23), as a large variation of scale factors leads to a moderate loss of precision in the gradient computation.

As further test instances, we use 1000 different random target angle prescriptions $\hat{\Theta}$ (with $\hat{\Theta}_i \in (\pi, 3\pi)$ for all vertices v_i) on a sphere mesh (1K vertices). The error decay is visualized in Figure 2.13. Note that the overall behavior is very similar, whether the prescribed angles are random or geometrically meaningful (as in Figure 2.12).

We consider the extreme scenario of concentrating the target metric's entire curvature in one point (i.e., prescribing a single cone of angle $2\pi(2g-1)$ in an otherwise flat metric). Errors for surfaces of increasing genus g (procedurally generated g-tori) are shown in Figure 2.14. A blow-up of the configuration around the single prescribed cone vertex on a genus 6 example is shown in Figure 2.16.

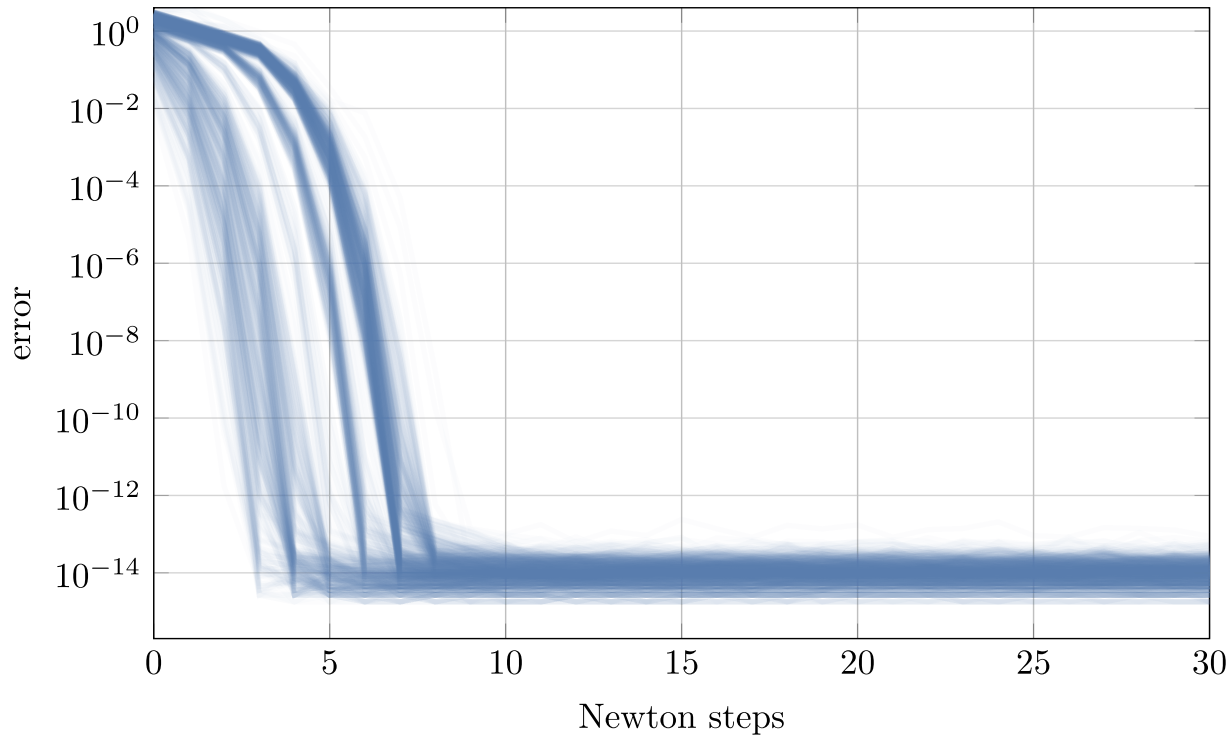


Figure 2.13: Like Figure 2.12, but each graph represents one of 1000 random test instances (again without boundary)

Surfaces with Boundary

The above mentioned dataset from [Myles et al. 2014] also contains meshes with one or more boundary loops, together with angle prescriptions $\hat{\Theta} > 0$ for interior and boundary vertices. The error decay on these cases is shown in Figure 2.17. Some of the models are visualized in Figure 2.15. The behavior is overall similar to the closed surface case.

As a synthetic test, we generate 1000 different random target angle prescriptions $\hat{\Theta}$ for a surface with boundary (a disk with 5K vertices). In the interior we prescribe a flat metric, at the boundary we prescribe a geodesic curvature, maximally in the range $\pm\pi$, i.e., $\hat{\Theta}_i \in (0, 2\pi)$ for all boundary vertices v_i. Figure 2.18 shows the number of the different types of symmetric flips that are performed in the course of the algorithm on these cases. As expected, the number of flips is larger for cases with a prescribed curvature spanning a larger range.

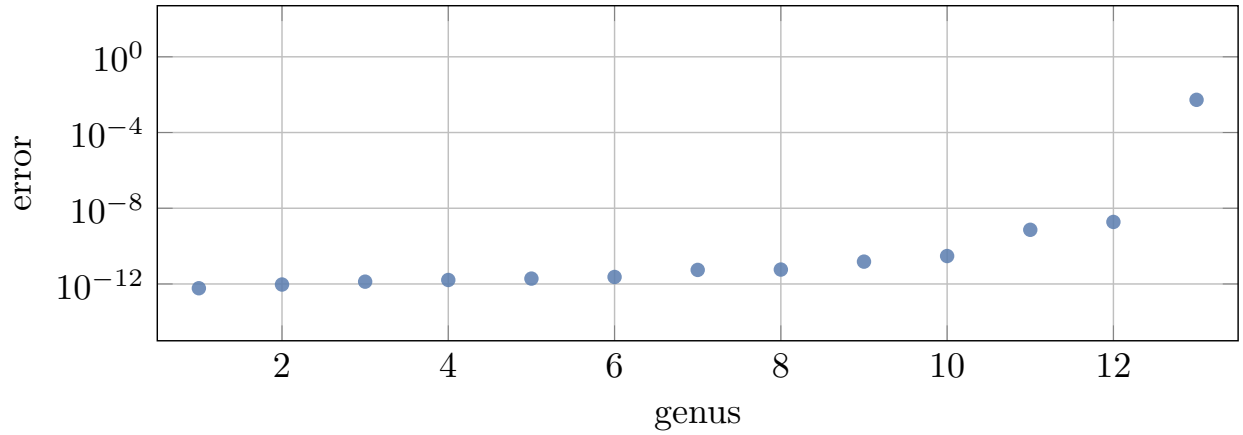


Figure 2.14: Final residual angle error for the extreme case of concentrating all curvature in a single cone on an g-torus surface (genus g). For the genus 12 case, where the residual error is still benign, the conformal scale factor spans 232 orders of magnitude. For the problematic genus 13 case it surpasses 262. By increasing numerical precision (Section 2.7.3), this can be remedied; for instance, with 200-bit precision, the $g = 150$ case converges to below 10^{-29}, with 400-bit precision, the $g = 400$ case to below 10^{-65} (with the scale factors spanning 611 orders of magnitude). (To reduce numerical issues in this extreme experiment, the initial step size λ was halved until the range of the coefficients of $\lambda \boldsymbol{d}$ was less than 10.)

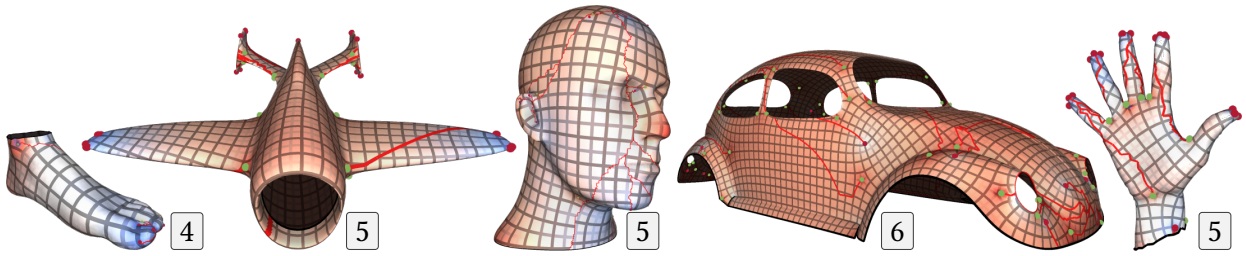


Figure 2.15: Visualization of conformal maps, analogous to Figure 2.10, on some of the models *with boundary* from the dataset of [Myles et al. 2014]. The boundary geodesic curvature is prescribed to be zero, therefore the angle between texture grid lines and the boundary is constant per boundary loop.

Another relevant scenario is that of prescribed geodesic curvature along a cut graph. We take the closed models of the dataset from [Myles et al. 2014] and mimic the setting employed by [Campen et al. 2019]: we compute a cut graph on each of these surfaces, and prescribe $\hat{\Theta}_i = \pi$ along this cut graph's segments' from both sides (effectively asking them to be straight under the conformal metric). The cut graph is composed of g short handle loops computed as in [Diaz-Gutierrez et al. 2009], connected by additional shortest paths. The resulting cut forms a graph on the surface with nodes of valence 3; at each node, an angle of π is prescribed for the largest sector, and angles $\pi/2$ for the remaining two. Some of the models are visualized in Figure 2.19,

with the cut graph marked in black. Depending on the shape of the cut graph, this scenario turns out to be the most challenging numerically: As can be seen in Figure 2.20 left, in a few cases the final maximum error is over 10^{-10}, i.e., higher than in the previously discussed scenarios. This is related to the scale distortion of the implied conformal metric spanning a range of up to 73 orders of magnitude in these cases. With higher-precision arithmetic, these residuals can be reduced, as discussed in Section 2.7.3.

2.7.2 Comparison

We demonstrate the advantages of the Delaunay flip approach over the degeneration flip approach (Section 2.3.2) in terms of efficiency as well as numerical robustness. To this end, we apply an implementation of the described method and an implementation of the algorithm described by [Campen and Zorin 2017b] (both using standard double precision floating point numbers) to the same set of inputs.

Efficiency

The main differences between the two methods lie in the number of linear system solves (to compute the descent direction $\boldsymbol{d}$) and the number of intrinsic flips. In the proposed method, the number of flips is often significantly higher (see the discussion in Section 2.3.2), while the number of system solves is lower. As a flip is a cheap local operation, while a system solve is an expensive global operation, a run time benefit can be conjectured.

The scatter plot in Figure 2.21 shows that this is the case on average. As test instances we use 1000 different random target angle prescriptions $\hat{\boldsymbol{\Theta}}$ (with $\hat{\Theta}_i \in (\pi, 3\pi)$ for all vertices v_i) on a sphere mesh (10K vertices). Only for relatively simple cases, where the target curvature can be matched without degeneration flips, the number of system solves may be similar. On average, run time is 73× lower with the Delaunay-based method on these examples.

Robustness

Differences in robustness can best be observed by considering extreme cases. In Figure 2.22 we show the residual error of the two methods when prescribing one very small or very large target angle (while distributing the remaining curvature). For small angles it becomes apparent that the degeneration flip algorithm is numerically more fragile.

2.7.3 Accuracy

While the method is theoretically guaranteed to yield the desired result, in practice numerical inaccuracies limit how closely the target curvature will be matched. As the method involves exponential and trigonometric functions (Equations (2.1) and (2.2)), it cannot be implemented in a numerically exact manner using adaptive precision rational or integer number types. Using extended precision floating point number types (such as MPFR), the method's accuracy can, however, be increased arbitrarily. We evaluate the effect of this choice on result accuracy in Figure 2.23. As test instances we use 1000 different random target angle prescriptions $\hat{\boldsymbol{\Theta}}$ (with $\hat{\Theta}_i \in (\pi, 3\pi)$ for all vertices v_i) on a sphere mesh (1K vertices).

As can be observed, the remaining error decreases consistently as the number of bits used for the floating point computations is increased. Due to dependence on many factors (input mesh and edge lengths, target angles, choice of linear system solver for the Newton direction) a simple bound on the error cannot be given, but Figure 2.23 gives an empirical idea of the behavior. Note that some correlation can be observed to the conformal scale distortion (the range $[e^{\min \boldsymbol{u}}, e^{\max \boldsymbol{u}}]$) that is required to match the target curvature.

In Figure 2.20 right the effect of increased precision on test cases from Section 2.7.1 can be observed. In particular, for the models that have maximum error over 10^{-10} when using standard double precision arithmetic, the error is reduced to below 10^{-16} when using a 100 bits mantissa

instead.

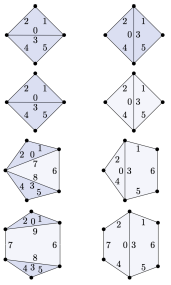
2.7.4 Failure Modes

We can distinguish two types of potential issues (detailed below): high residual error or a high number of optimization steps. The former can be caused by limited arithmetic precision (and can therefore be remedied by using extended precision, as demonstrated above). The latter can be caused by an unfavorable energy landscape, and is therefore more fundamental, regardless of numerics.

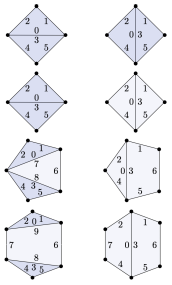
Precision-related failures.. Depending on the target curvature, a high amount of metric distortion may be required, with negative numerical effects on result accuracy. It can be observed that this is correlated with local concentrations of positive or negative target (Gaussian or geodesic) curvature. Figure 2.24 left shows an experiment in which an increasingly large cluster of vertices have a target angle below 2π and the rest above 2π. When using standard double precision, a large fraction of these synthetic test cases essentially fails to reach a reasonably accurate state. Performing these computations with higher precision (Figure 2.24 right) resolves these problems. Analogously, we notice that cut graphs with more complex shape than the ones used in Figure 2.20 (e.g., the more constrained "hole-chain" in [Campen et al. 2019]) cause a similar behavior.

Near-degeneracy failures.. This second issue is more fundamental. While the method may, in principle, converge eventually, the step size can decrease to the point that the number of iterations needed becomes impractical. When using the above mentioned hole-chain choice of cuts on the dataset of [Myles et al. 2014], we can identify four high-genus models with complex singularities which fail to converge in a reasonable number of steps even with high-precision arithmetic. The underlying reason is illustrated in Figure 2.25, showing the plot of the projected gradient $\boldsymbol{d}^\intercal \boldsymbol{g}(\boldsymbol{u} + \lambda \boldsymbol{d})$ for a line search direction. One can see that while theoretically the gradient is C^1,

Table 2.1: Combinatorial updates required to perform symmetric flips of all relevant consistent types. The change to $\mathcal{N}$ is given by listing the orbits (halfedge cycles forming faces) of $\mathcal{N}$ created by the flip. The employed indexing is depicted in the figures left and right. Similarly, we define changes to R viewing it as a permutation with orbits of length 1 or 2, and listing the sets of orbits being replaced. Finally, rather than deleting and adding new halfedges on demand, for implementational efficiency we can associate a superfluous pair of halfedges, eliminated by a quad-creating flip, with the quad (listed behind the bar)



$(1,1,1)+(2,2,2) \leftrightarrow (1,1,1)+(2,2,2)$	
$\mathcal{N}: (h_0^i, h_1^i, h_2^i), (h_3^i, h_4^i, h_5^i),\ i=1,2$ R : unchanged	$\mathcal{N}: (h_0^i, h_2^i, h_4^i), (h_1^i, h_3^i, h_5^i),\ i=1,2$ R : unchanged
$(1,\|\|,2) \leftrightarrow (t,\perp,t)$	
$\mathcal{N}: (h_0,h_1,h_2), (h_3,h_4,h_5)$ $R: (h_0,h_3)$	$\mathcal{N}: (h_0,h_2,h_4), (h_1,h_3,h_5)$ $R: (h_0), (h_3)$
$(1,1,t)+(2,2,t) \leftrightarrow (t,\perp,q)$	
$\mathcal{N}: (h_0,h_1,h_2), (h_3,h_4,h_5), (h_6,h_7,h_8)$ $R: (h_0,h_3), (h_7,h_8)$	$\mathcal{N}: (h_0,h_2,h_4), (h_1,h_3,h_5,h_6) \mid h_7,h_8$ $R: (h_0), (h_3)$
$(1,1,q)+(2,2,q) \leftrightarrow (q,\perp,q)$	
$\mathcal{N}: (h_0,h_1,h_2), (h_3,h_4,h_5), (h_6,h_9,h_7,h_8)$ $R: (h_0,h_3), (h_8,h_9)$	$\mathcal{N}: (h_0,h_2,h_7,h_4), (h_1,h_3,h_5,h_6) \mid h_8,h_9$ $R: (h_0), (h_3)$



it may experience very significant jumps, when a large number of triangle flips happen nearly simultaneously as λ changes (in this particular case 58). We observe that this occurs in particular in the presence of highly distorted near-degenerate triangles.

2.8 Proofs and additional Lemmas

Proof of Proposition 2

Proof. If x is not fixed, by the well-definedness of R on mesh elements, for each $h \in x$ we have $R(h) \notin x$. Therefore for a non-fixed individual face or edge x all its halfedges can be assigned to H^1 (or to H^2) without contradicting the conditions. It needs to be shown that this can be done for all such elements consistently.

Let H^e the set of halfedges whose edges are not fixed and H^f the set of halfedges whose faces are not fixed. Let Q the relation that is the union of $O|_{H^e}$ and $\mathcal{N}|_{H^f}$ on $H \setminus H^s$. Consider the connected components H_i of Q (intuitively: the mesh's connected components separated by fixed edges and

fixed faces). Due to the properties of R (preserving/inverting O and N) it is well-defined on these connected components via $R(H_i) = H_j \Leftrightarrow R(h) \in H_j$ for any $h \in H_i$. Using arguments analogous to [Panozzo et al. 2012, Prop. 2] one verifies that the set of fixed elements necessarily forms a cycle; therefore there are at least two such connected components.

As R on $H \setminus H^s$ has orbits of length 2 only, it allows a bipartition of the connected components, i.e., they can be assigned to two sets H^1 and H^2 in accordance with the above conditions. □

Label Compatibility

Proposition 3.

(a) $e \in E^{\perp} \Rightarrow f_a, f_b \in F^s$.

(b) $e \in E^{\parallel} \Rightarrow f_a \in F^1, f_b \in F^2$ *or* $f_a = f_b \in F^s$.

(c) $e \in E^1 \Rightarrow f \notin F^2,\ e \in E^2 \Rightarrow f \notin F^1$.

(d) $e \in E^i, f_a, f_b \in F^s \Rightarrow R(e) \in f_a, f_b$.

Proof. Part (a) follows immediately from the definition of F^i, as faces from F^i cannot have edges from $E^{\perp}$.

Suppose a face f_a is incident at an edge e from $E^{\parallel}$. For these edges $R(e) = e$. Suppose $f_a \in F^1$, then $R(f_a)$ is incident to $R(e) = e$, therefore $f_b = R(f_a)$. As $R(f_a) \in F^2$ by definition of F^2, this proves the first part of (b). Suppose $f_a \in F^s$, and let h a halfedge $h \in e$, $h \in f_a$. Then $R(h) \in f_a$ by the definition of F^s; but, by definition of $E^{\parallel}$, $R(h) \in e$, so $f_a = f_b$, i.e., a face is adjacent to itself along e.

Part (c) directly follows from the definitions of E^i and F^i.

In part (d), suppose f_a and f_b are incident at $e \in E^1$, $f_a, f_b \in F^s$, and $e = (h_a, h_b)$. Then $R(h_a) \in R(f_a) = f_a$, $R(h_b) \in f_b$, and $O(R(h_a)) = R(h_b)$ by the properties of R, i.e., $(R(h_a), R(h_b))$ is an edge. By definition of E^i, it has to be in E^2, i.e., faces f_a and f_b share a second edge, and this edge is from E^2. □

Irrelevance of Flip Types $(s, \|, s)$ and $(s, 1, s)$

Proposition 4. *Types $(t, \|, t)$, $(q, \|, q)$, $(t, 1, t)$, $(t, 1, q)$, and $(q, 1, q)$ are associated with edges that are Delaunay regardless of metric.*

Proof. Consider $(t, \|, t)$. By Proposition 3(b), it corresponds to a configuration with a single face: $(f^t, e^{\|}, f^t)$. As the triangle f^t is isosceles, and both side edges of the triangle coincide with $e^{\|}$, angles opposite $e^{\|}$ are $\pi/2 - \alpha/2$ if the apex angle is α, i.e., their sum is guaranteed to be less than π and the edge is Delaunay. For $(q, \|, q)$, to evaluate the Delaunay criterion, we split f^q into triangles. As f^q is inscribed the choice of diagonal does not affect the angles; we can choose the diagonal that connects a vertex of $e^{\|}$ with a vertex with trapezoid angles $\leqslant \pi/2$ (see inset), from which we can see that both angles opposite $e^{\|}$ are less than $\pi/2$. For cases $(t, 1, t)$, $(t, 1, q)$, and $(q, 1, q)$ the same logic applies to each face incident at the shared edge e^1. □

2.9 Double Cover: Formal Definition

Given a mesh $N = (H^0, \mathcal{N}^0, O^0)$, with boundary and interior halfedges $H^{bnd} \cup H^{int} = H^0$, we discard H^{bnd} and set $H = H^1 \cup H^2$ where $H^1 = H^{int}$ and $H^2 = \bar{H}^{int}$, where $\bar{\cdot}$ denotes a copy. The reflection map R is defined via $R(h) := h'$ if $h' \in H^2$ is the copy of $h \in H^1$. O^0 is adopted on both copies to define O, except that $O(h) := R(h)$ if $O^0(h) \in H^{bnd}$; this latter adjustment constitutes the *gluing*

of the two copies along their boundaries. Finally

$$\mathcal{N}(h) := \begin{cases} \mathcal{N}^0(h) & \text{if } h \in H^1, \\ R((\mathcal{N}^0)^{-1}(R(h))) & \text{if } h \in H^2. \end{cases}$$

This forms the symmetric double cover mesh $M = (H, \mathcal{N}, \mathcal{O}, R)$ with triangle faces and map R. Note that R is a reflection map: it satisfies the conditions of Theorem 2.5 (where condition (3) is void as M has no boundary). It is easy to see that this construction implies $E^{\perp} = \emptyset$ and $F^s = \emptyset$, i.e., no element crosses the symmetry line (the former boundary). $E^{\parallel}$ contains the edges lying *on* the symmetry line, i.e., those for whose halfedges the $\mathcal{O}$ relation was adjusted to glue the two copies.

2.10 Conclusions and Future Work

We presented a practical realization of the method for computing discrete conformal maps based on the ideas of [Gu et al. 2018b; Springborn 2020], elaborating how it can be applied safely to meshes with boundary, the most practically relevant scenario for conformal mapping. Our improvements include a straightforward to implement algorithm for maintaining symmetric Delaunay triangulations and several improvements increasing the robustness of Newton's optimization method in the context of our application. We explored its behavior on a standard dataset, and for a number of challenging synthetic examples, demonstrating its robustness for a broad range of cases involving high distortion. We also observe that common failure cases can be addressed by using extended precision arithmetic, albeit at a significant cost in run time.

However, in our extensive tests we did identify a small number of cases for which the method does not produce a conformal map in reasonable time, which indicates potential for further algorithmic improvements. It would also be desirable to find ways to minimize the use of extended

precision arithmetic to the minimum necessary in a *filtered* approach, so as to increase accuracy while maintaining performance. Finally, extension of the method from Euclidean to spherical and hyperbolic discrete metrics would be not only of theoretical interest [Schmidt et al. 2020].

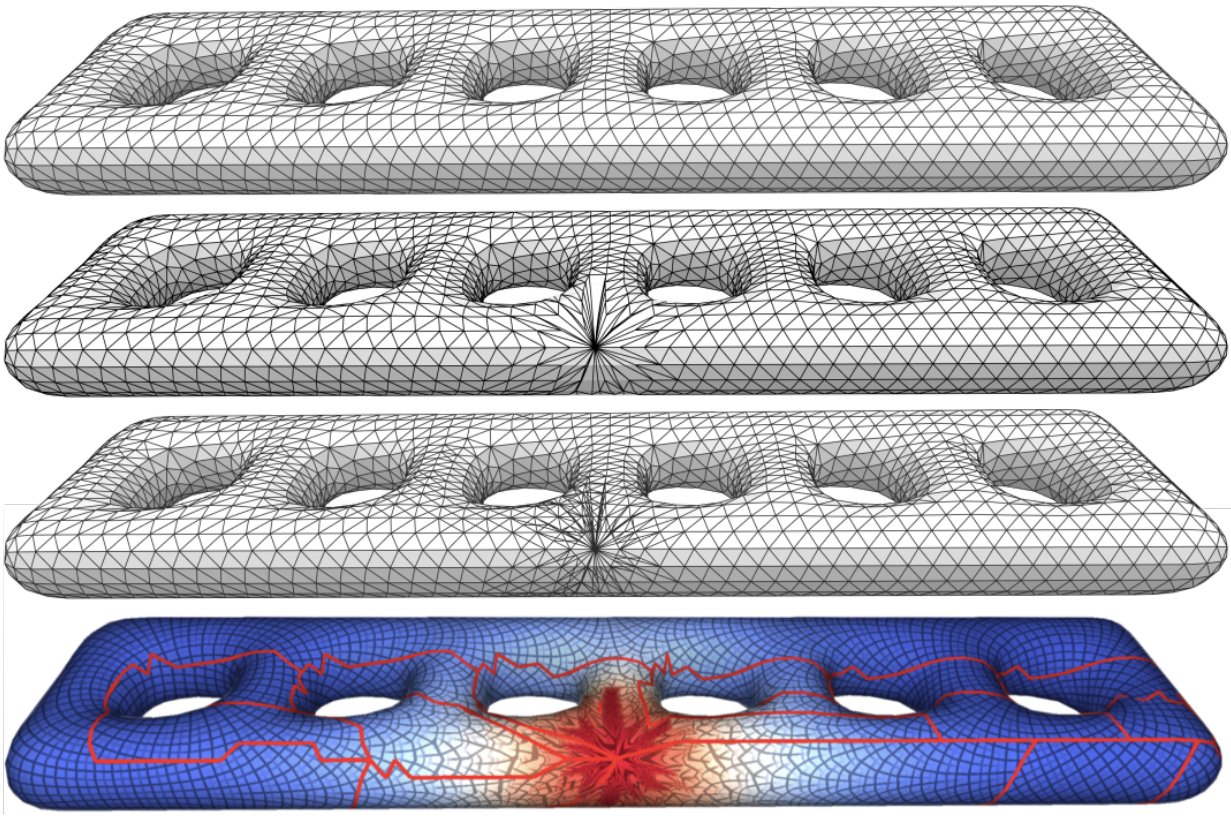


Figure 2.16: Top: Input triangulation. Second row: Resulting intrinsic retriangulation, when concentrating all curvature on a single vertex ($\Theta = 22\pi$); it is Delaunay under the computed conformal metric (with curvature -20π at the central vertex). Third row: overlay triangulation [Fisher et al. 2007], allowing for a simple representation of the implied conformal map, linear or projective per triangle. Bottom: Visualization of implied conformal map using a hierarchical grid texture (spanning 25 levels in this extreme case).

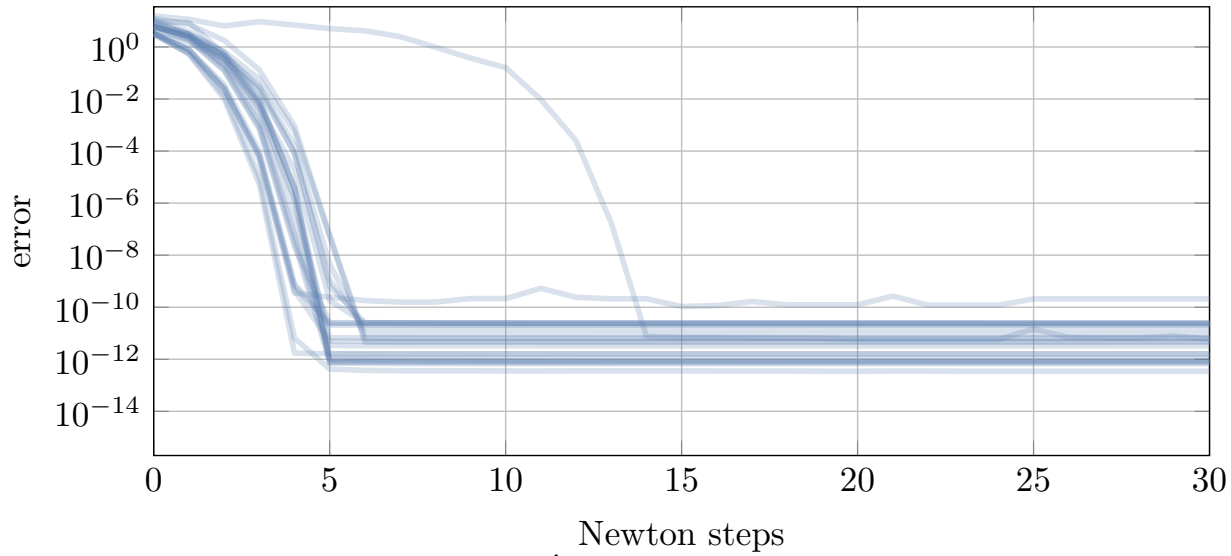


Figure 2.17: Decay of maximum angle error $\|\hat{\Theta} - \Theta\|_\infty$ over the iterations of the Newton algorithm. Each graph represents one of the instances *with boundary* from the dataset of [Myles et al. 2014].

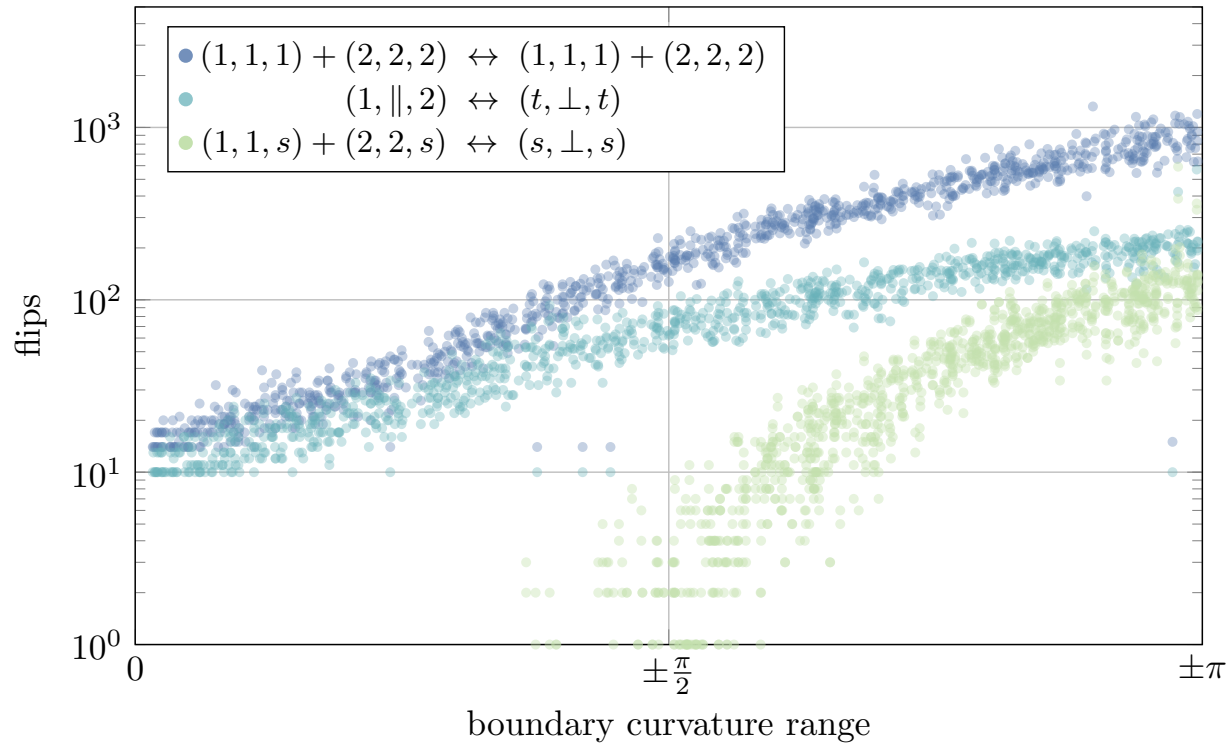


Figure 2.18: Scatter plot showing the numbers of different types of symmetric flips during the algorithm relative to the range of prescribed random boundary curvatures. Each dot represents one type of flips for one of 1000 test instances.

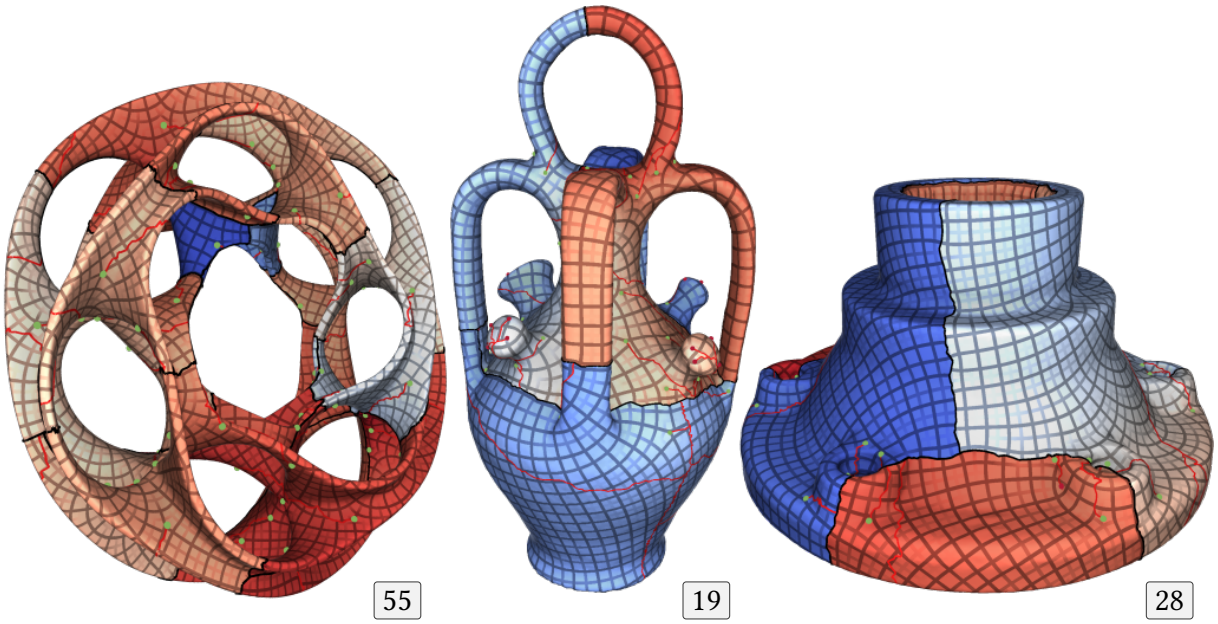


Figure 2.19: Visualization of conformal maps with cones, analogous to Figure 2.10, on models cut to disk topology using a cut graph (black). Due to the prescribed geodesic curvature along the cut boundary, the cut is axis-aligned under the map. Notice that such enforced alignment can easily imply a broad range of scales, which is challenging numerically.

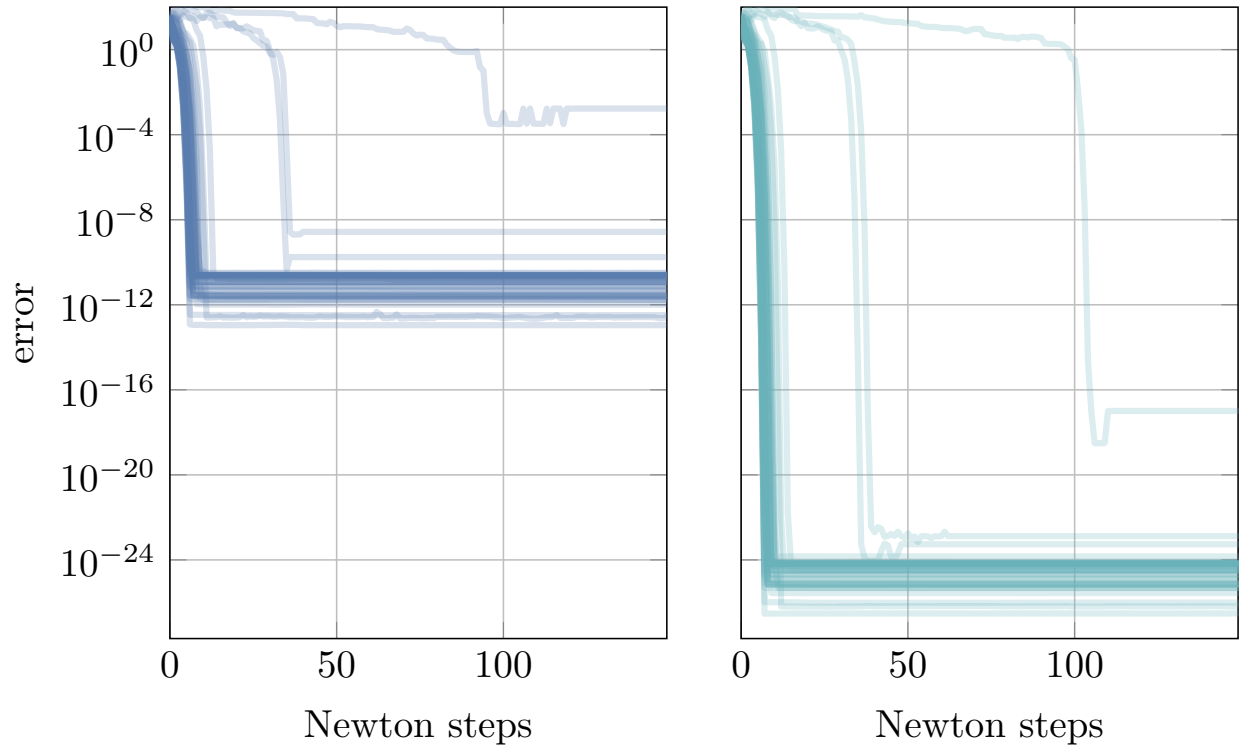


Figure 2.20: Decay of maximum angle error $\|\hat{\Theta} - \Theta\|_\infty$ over the iterations of the Newton algorithm. Each graph represents one of the closed instances from the dataset of [Myles et al. 2014], with *prescribed curvature along a cut graph.* Left: `double` precision. Right: extended precision (100 bits mantissa).

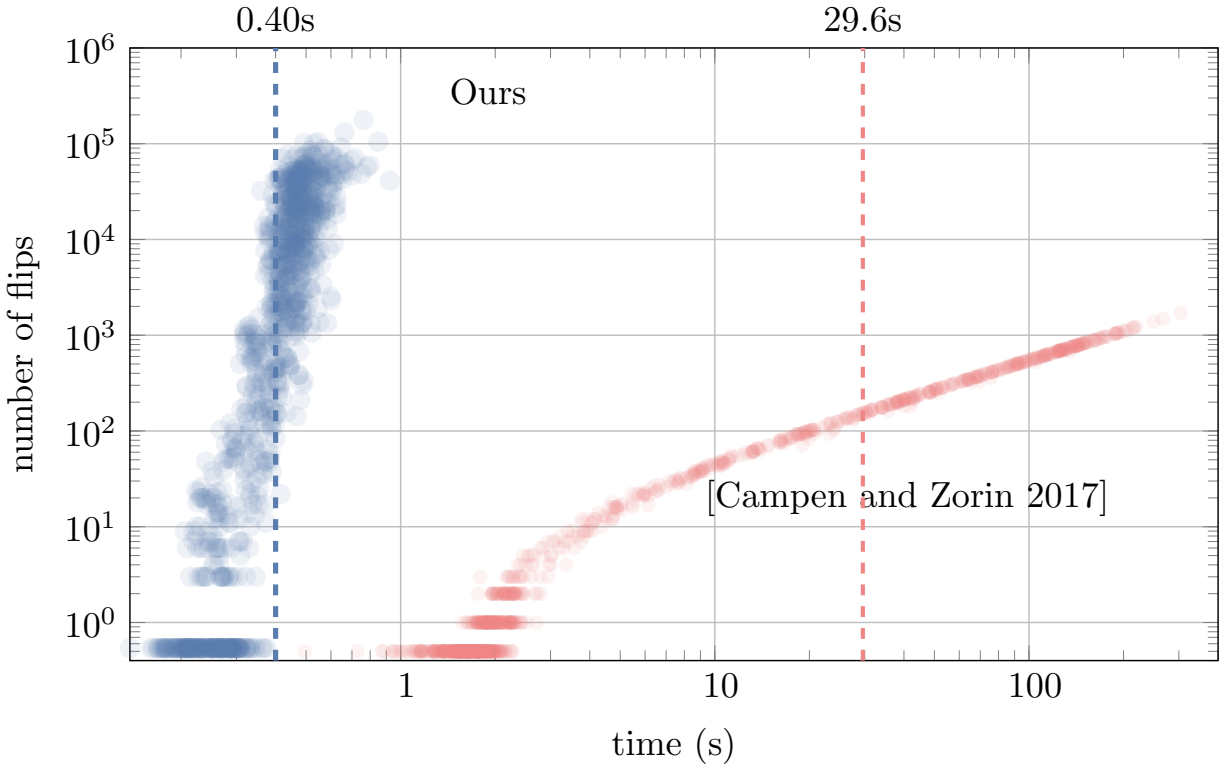


Figure 2.21: Scatter plot showing the number of flips and the run time (to reach $\varepsilon_{\text{tol}} = 10^{-10}$), for the described Delaunay-flip method (blue) and the degeneration flip method (red). Each dot represents one of 1000 test instances. Dashed lines mark the average run time, 0.4s and 29.6s, respectively.

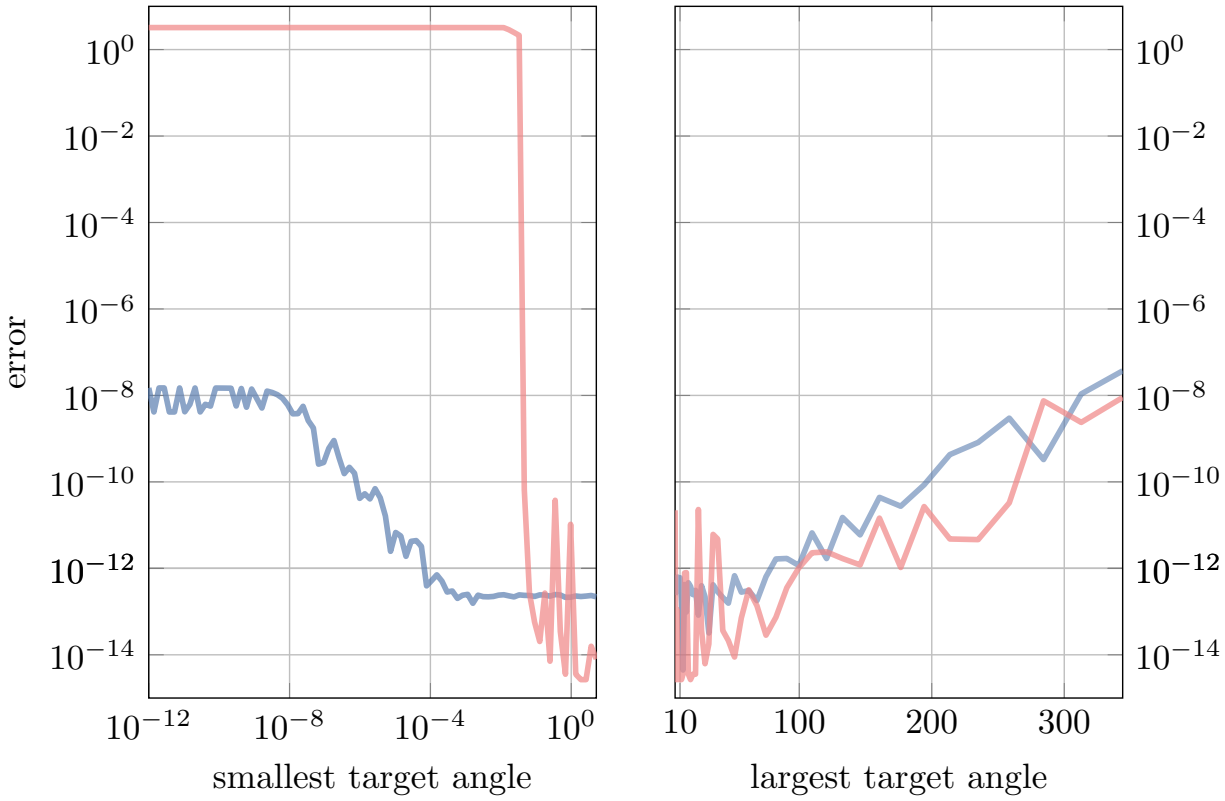


Figure 2.22: Final residual angle error $\|\hat{\Theta} - \Theta\|_\infty$ for extreme cases (one very small or very large target angle, on a sphere with 1K vertices), comparing the Delaunay-based algorithm (blue) and the degeneration flip algorithm. [Campen and Zorin 2017b] (red).

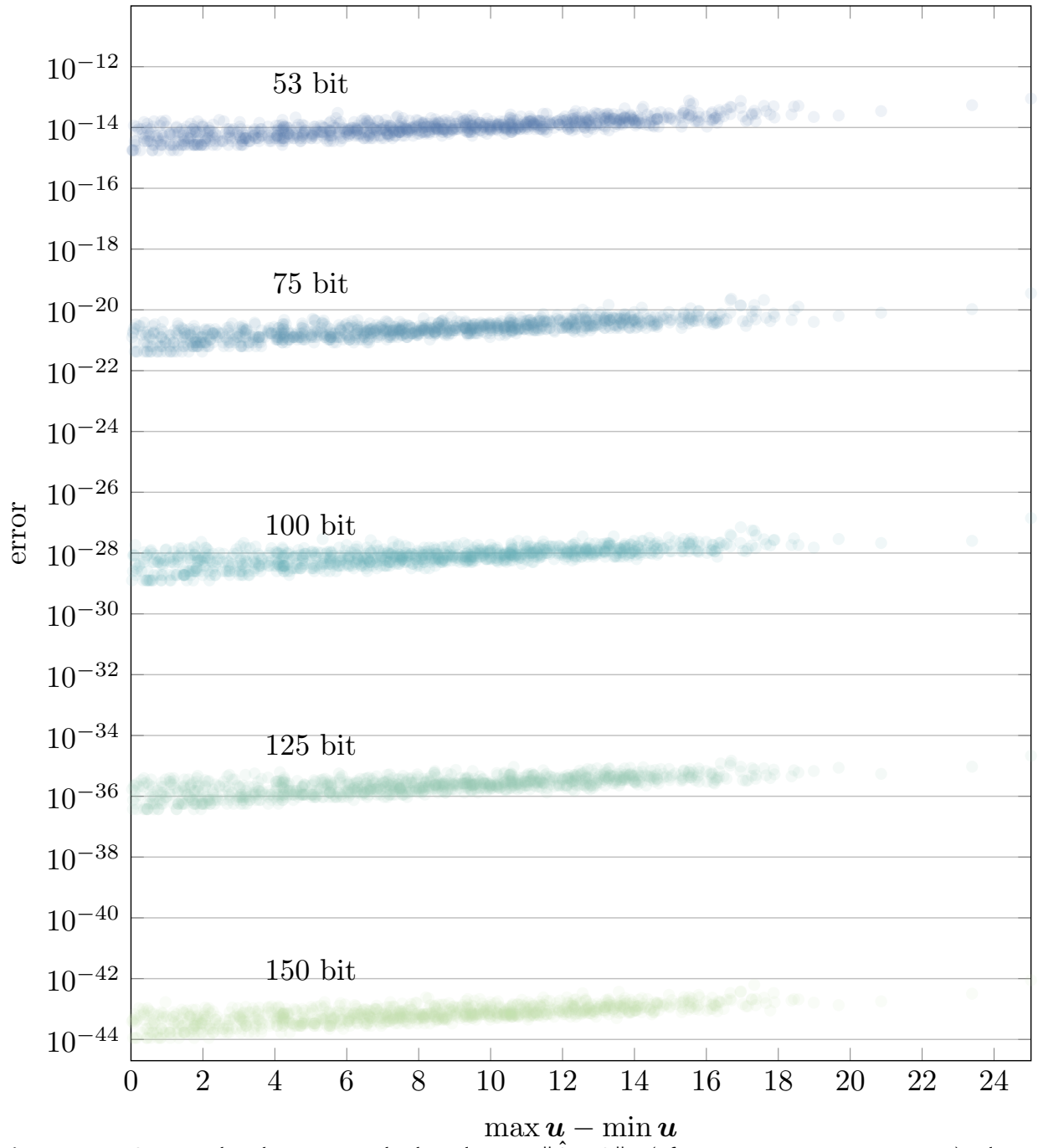


Figure 2.23: Scatter plot showing residual angle error $\|\hat{\boldsymbol{\Theta}} - \boldsymbol{\Theta}\|_\infty$ (after at most 50 Newton steps) relative to the range of logarithmic conformal scale factors $\boldsymbol{u}$. Each dot represents one test instance, run using floating point numbers with a mantissa of 53 bits (`double`), 75 bits, 100 bits, 125 bits, 150 bits (`MPFR`).

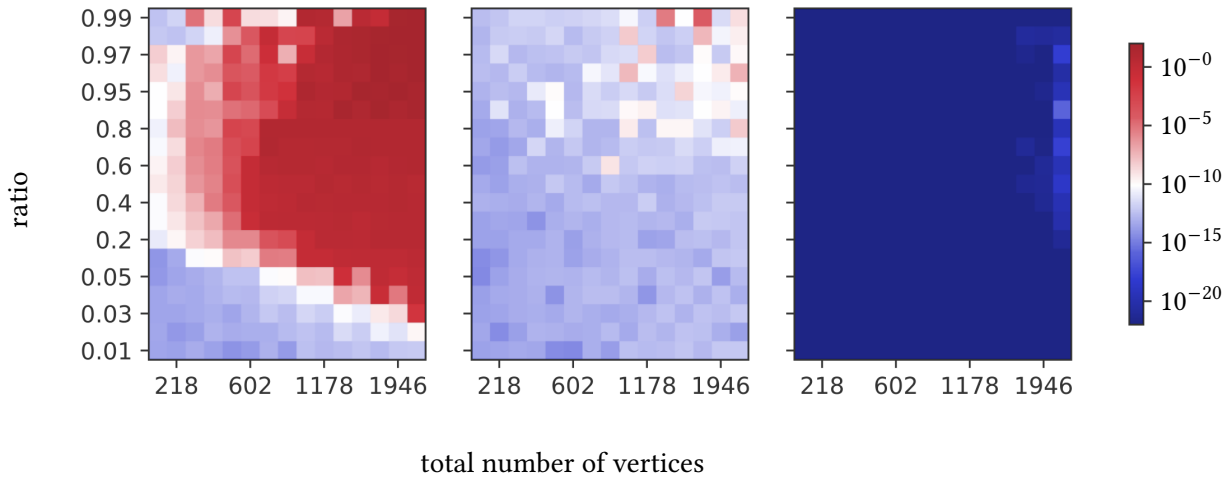


Figure 2.24: Heatmap showing the final error $\|\hat{\Theta} - \Theta\|_\infty$ for spheres of varying resolution (x-axis) with some ratio (y-axis) of the vertices set to target angle 3 and the rest to a constant target angle $< 2\pi$ such that the Gauss-Bonnet theorem is satisfied. Left: double precision results when the two angle values are distributed in two clusters. Center: double precision results when the two angle values are distributed randomly over the sphere. Right: extended precision (150 bits mantissa) results with the same distribution as left. (For this experiment, the threshold for the gradient norm decrease was set to 0 and, to reduce the run time in this particular case, λ was chosen adaptively, initially halved until the range of coefficients of $\lambda \boldsymbol{d}$ was less than 10.)

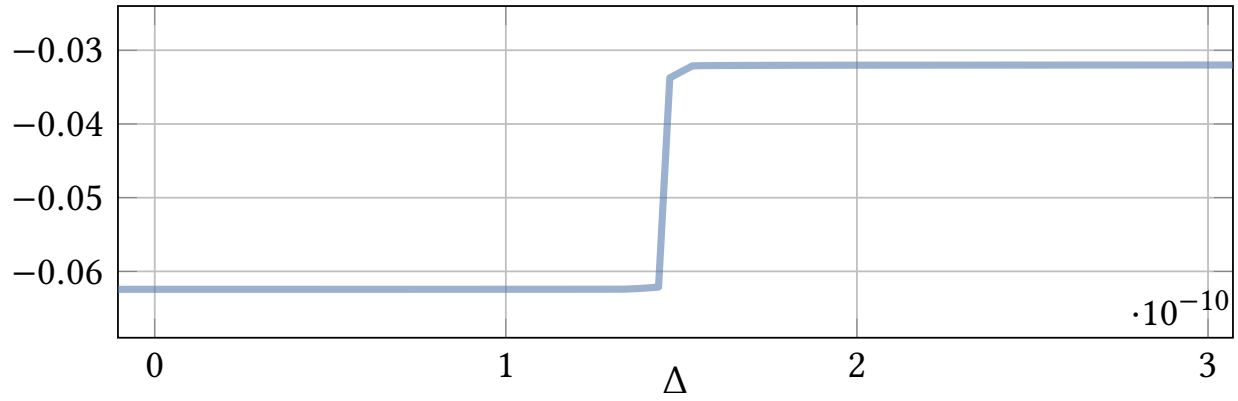


Figure 2.25: Projected gradient $\boldsymbol{d}^\intercal \boldsymbol{g}(\boldsymbol{u} + \lambda \boldsymbol{d})$ along the normalized Newton descent direction with step length $\lambda = 0.0217745227 + \Delta$.

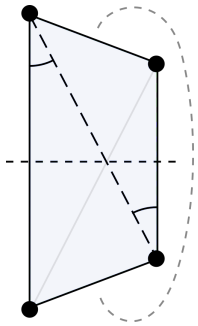


Figure 2.26: Configuration $(q, \|, q)$ showing the diagonal choice for evaluating the Delaunay criterion.

3 | Which Cross Fields can be Quadrangulated? Global Parameterization from Prescribed Holonomy Signatures

This chapter is adapted from the publication on ACM Transactions on Graphics [Shen et al. 2022], a joint work with Hanxiao Shen, Ryan Capouellez, Daniele Panozzo, Marcel Campen and Denis Zorin.

Abstract

We describe a method for the generation of seamless surface parametrizations with guaranteed local injectivity and full control over holonomy. Previous methods guarantee only one of the two. Local injectivity is required to enable these parametrizations' use in applications such as surface quadrangulation and spline construction. Holonomy control is crucial to enable guidance or prescription of the parametrization's isocurves based on directional information, in particular from cross-fields or feature curves, and more generally to constrain the parametrization topologically.

To this end we investigate the relation between cross-field topology and seamless parametrization topology. Building on prior work on locally injective parametrization and holonomy control, we propose an algorithm that meets these requirements. A key component is the observation that arbitrary surface cut graphs for global parametrization can be homeomorphically modified. This enables almost any set of turning numbers with respect to a target cross-field.

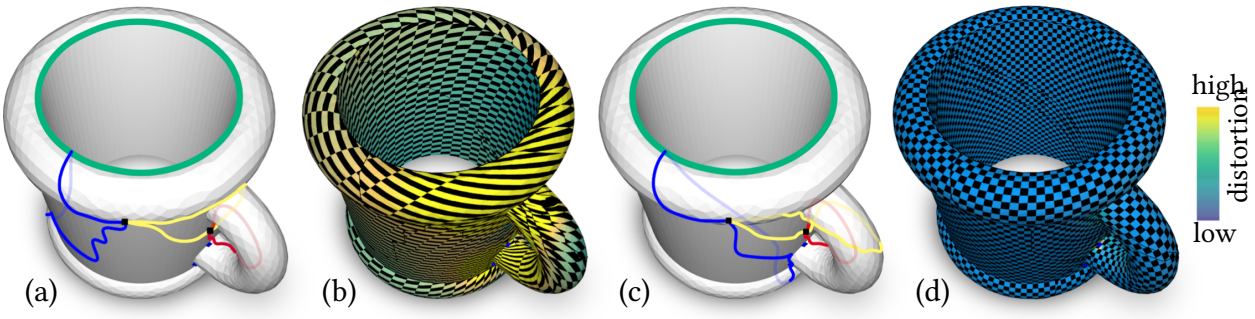


Figure 3.1: Our method is able to construct seamless parametrizations with prescribed *holonomy signature*, i.e. it offers full control over map topology. This topological information can be given as values (holonomy numbers) on a system of certain loops on the surface (a) or, e.g., be derived from a cross-field that the parametrization is supposed to align to. A state-of-the-art method [Campen et al. 2019] can reliably generate seamless parametrizations (b), but it ignores this information at the global level, i.e. it does not offer full control over map topology. Our method adjusts the loop system with associated numbers into a topologically equivalent state (c) of very particular type. Using a cut graph constructed from this loop system, we are then able to reliably generate a seamless parametrization (d) that perfectly matches the prescribed holonomy signature, and for instance allows for lower distortion and better cross-field alignment.

Contributions

- This chapter characterizes when a prescribed holonomy signature is realizable by a seamless parametrization under a mild condition; in particular, it is already sufficient that the signature contains at least one cone with angle deficit $+\pi/2$ or $-\pi/2$.
- A *rerouting* procedure homeomorphically modifies a loop/cut system to an equivalent one whose turning numbers match the target holonomy in a construction-friendly form.
- Based on the rerouted loop system, the chapter constructs seamless parametrizations with

full holonomy control while maintaining local injectivity, via a variant of Seamless Padding (SP).

Organization

We introduce holonomy and the motivation for holonomy control (Section 3.1) and review related work (Section 3.2). We then develop the theoretical foundation and existence results (Section 3.3), followed by an overview of the algorithmic pipeline (Section 3.5). The core construction modifies and reroutes the loop system to realize the target holonomy (Section 3.6) and builds the corresponding parametrization domain and map (Section 3.7). We evaluate the method (Section 3.8) and conclude with a discussion of limitations and future directions (Section 3.9).

3.1 Introduction

Seamless surface parametrization is one of the most common approaches to constructing seamless texture atlases, conforming surface quadrangulations, and high-order (spline or subdivision) approximations to surface data. A chart-based parametrization is called seamless if it satisfies certain conditions on its transitions between charts or across cuts.

In particular, a seamless parametrization of a discrete surface defines a metric, i.e., an edge length assignment on a mesh, that is intrinsically flat almost everywhere, i.e. angles around vertices sum to 2π, except at a (often small) set of *cone vertices* with an angle deficit (or excess) of some multiple of $\frac{\pi}{2}$. More generally, the *holonomy angle* for any closed loop on the surface is a multiple of $\frac{\pi}{2}$. The holonomy angle is the angle between the first and last edge when laying out a closed chain of mesh triangles in the plane according to the metric (Figure 3.4, cf. [Bright et al. 2017; Crane et al. 2010]). Informally, this holonomy condition on a parametrization's metric ensures that parametric lines continue seamlessly across cuts, although, e.g., a u-parametric line may

become a v-parametric line.

While the set of closed triangle chains on a discrete surface is infinite, all their holonomy angles are actually defined by a set of angles on a finite basis, the *holonomy signature*, which we define more precisely below. In essence, loops around individual vertices capture all *local* aspects of holonomy, while (in case of non-trivial topology, genus > 0) a system of non-contractible loops captures the additional *global* aspects.

Holonomy Control. To clarify the importance of parametrization topology defined by the holonomy signature, consider quad meshes or quad layouts obtained from (constrained classes of) seamless parametrizations by tracing a grid of parametric lines on the surface. The cones become the extraordinary vertices, where $n \neq 4$ quads meet. The holonomy angles determine how many quads meet at such extraordinary vertices, and more generally, how many turns the edges of the quads make along any closed curve on the surface, e.g., a feature line. As a consequence, controlling the parametrization's topology in the form of its holonomy angles is critical for obtaining a high-quality parametrization with intended behavior.

In many approaches to seamless parametrization, the target topology is provided as input, e.g., it is derived from a given cross-field or partially or completely specified by the user. At the same time, as we discuss in detail in Section 3.2, no existing general method guarantees that the target topology is fully respected, although significant progress was made towards this goal.

Existence. Moreover, to the best of our knowledge, the answer to the following question is not known:

> *For which holonomy signatures, seamless parametrizations with corresponding topology exist?*

Partially, this question was answered in [Jucovič and Trenkler 1973], and more specifically in [Campen et al. 2019], where angles at cones, but not complete signatures (including global aspects), were considered. In this paper, we resolve the question of existence for a broad class of signatures, subject to only a mild condition.

Remarkably, it turns out that for surfaces of genus $\neq 1$, there is a seamless parametrization for *any* holonomy signature (e.g. implied by a cross-field) under this condition. For genus 1, we show that in this class the one known example of holonomy signatures for which there is no seamless parametrization (signatures with exactly two cones, with angles $3\pi/2$ and $5\pi/2$) is the only one.

For the condition to be satisfied, it is already sufficient (but not necessary) to have one cone with angle deficit $+\frac{\pi}{2}$ or $-\frac{\pi}{2}$ in the signature (corresponding to at least one valence 3 or valence 5 extraordinary quad vertex). This is essentially always satisfied for holonomy signatures implied by cross-fields optimized for smoothness or curvature alignment [Vaxman et al. 2016].

Contribution. We describe an algorithm for the construction of seamless parametrizations with full control over holonomy. It extends the construction of [Campen et al. 2019] (referred to as Seamless Padding (SP) in the following) which provides control only over local holonomy aspects (i.e. cone angles). Our contribution includes:

- An existence result for seamless parametrizations with given holonomy signature, indicating a remarkably small topological gap between cross-fields and parametrizations;
- An algorithm for, given a holonomy signature, constructing an alternative system of loops on which the equivalent holonomy signature has arbitrary desired angles;
- A variant of the SP method that, based on the above, builds a valid seamless parametrization with prescribed holonomy.

We note that the topology of cross-fields—which are often used to guide the computation of seamless parametrizations—can be controlled very flexibly and precisely using existing discrete construction algorithms. In fact, one can easily construct a cross-field with any given turning number signature (the field analogue of a holonomy signature, cf. Section 3.4.1) by solving a linear system of equations [Crane et al. 2010]. The ability to near-universally match this signature, provided by our method, means that this possibility of precise topology control extends to parametrizations.

3.2 Related Work

Seamless Parametrization. Seamless surface parametrization with prescribed singularities (locations and indices) is a problem that has received significant attention recently. Results include [Tong et al. 2006; Kälberer et al. 2007; Bommes et al. 2009; Kovacs et al. 2011; Myles and Zorin 2013; Myles et al. 2014; Campen et al. 2015; Fu et al. 2015a; Chien et al. 2016; Ebke et al. 2016; Bright et al. 2017; Zhou et al. 2018; Fang et al. 2018; Hefetz et al. 2019; Campen et al. 2019; Zhou et al. 2020; Lyon et al. 2019]. Prominent use cases are surface quadrangulation [Bommes et al. 2013b; Campen 2017] and spline conversion [Marinov et al. 2019; Campen and Zorin 2017b].

Cross-Field Guidance. Most often such parametrizations are generated and optimized guided by a cross-field or frame field on the surface [Vaxman et al. 2016]. Seminal works on cross-field guided parametrization are [Knupp 1995; Kälberer et al. 2007]. Important ideas for cross-field generation are presented by [Ray et al. 2008; Crane et al. 2010; Li et al. 2006; Bommes et al. 2009; Ray et al. 2009]; many of these offer control over the fields' turning numbers.

Local Injectivity. For common use cases, parametrizations are valid only if they are locally injective, i.e. free of fold-overs. Local injectivity constraints required to ensure a valid parametrization

are, due to their challenging non-convex nature, not rarely omitted [Kälberer et al. 2007; Bommes et al. 2009; Kovacs et al. 2011; Myles and Zorin 2013; Ebke et al. 2016; Zhou et al. 2018] or convexified in a conservative manner [Lipman 2012; Bommes et al. 2013a; Campen et al. 2015; Bright et al. 2017; Hefetz et al. 2019].

Guarantees. In special cases (restricted genus, restricted cone configurations) convex formulations can be used to reliably yield locally injective seamless parametrizations [Gu and Yau 2003; Gortler et al. 2006; Aigerman and Lipman 2015]. Alternatively, additional user input like a surface partition may be exploited to ensure validity [Tong et al. 2006], or more general, non-piecewise-linear forms of parametrization may be employed [Aigerman and Lipman 2016].

Recently, first methods have emerged that provide validity guarantees while supporting arbitrary genus and general cone configurations [Campen et al. 2019; Zhou et al. 2020]. In this sense, they offer control over *local* holonomy aspects. No *global* control over holonomy is provided, though. Therefore, when for instance aiming to generate a cross-field guided parametrization, while locally cones are reproduced, there may be global topological mismatches between the given cross-field and the constructed parametrization, for instance precluding proper alignment, as in Figure 3.1 (b).

Existence. [Jucovič and Trenkler 1973] consider the question of existence of quadrilateral meshes with prescribed irregular vertex valences, [Campen et al. 2019] the very closely related question of existence of seamless parametrizations with prescribed cones. Both consider only local holonomy (irregular vertex valence, cone angles), not global holonomy.

Distortion Optimization. The task of parametrization optimization (with respect to varying measures of distortion) while maintaining properties such as local injectivity or seamlessness, is addressed in a number of recent works [Schüller et al. 2013; Hormann and Greiner 2000; Rabi-

novich et al. 2017a; Kovalsky et al. 2016; Zhu et al. 2018; Shtengel et al. 2017; Mandad and Campen 2020]. They are useful as a post-process in the context of our method as we initially focus mainly on validity, holonomy, and seamlessness.

Cone Choice. The choice of cones (and more generally guiding cross-fields, holonomy signatures) is an application dependent matter. Various approaches have been proposed for the selection of a cone configuration, for instance curvature-based (e.g. via cross-fields [Vaxman et al. 2016]), distortion-based [Kharevych et al. 2006b; Soliman et al. 2018b; Ben-Chen et al. 2008], or interactive [Ebke et al. 2016; Campen and Kobbelt 2014]. The problem of positioning cones such that conformal maps with these cones become seamless is addressed by [Chen et al. 2019, 2020].

Holonomy. Some of the above methods for parametrization construction (such as [Bommes et al. 2009; Bright et al. 2017]) offer full control over the resulting parametrizations' holonomy, but do not guarantee local injectivity. Those that guarantee local injectivity in a general setting (e.g. [Campen et al. 2019; Zhou et al. 2020; Myles et al. 2014]), in turn, do not offer full control over holonomy. The method of [Campen and Zorin 2017b] offers full holonomy control, albeit only for the broader class of seamless *similarity* parametrizations.

3.3 Existence of Seamless Parametrizations

Before discussing the algorithmic details, we first address the existence of seamless parametrizations for prescribed holonomy signatures. This guarantees that the rerouting operations described above can indeed be carried out as needed.

While any choice of homology basis loops will yield a holonomy signature, our method relies on bases whose loops' holonomy numbers are some specific values. As a first step towards achieving this, the following proposition gives us a simple way to "add" together two loops so that the

holonomy number of the new loop is determined by the holonomy numbers of the constituent loops.

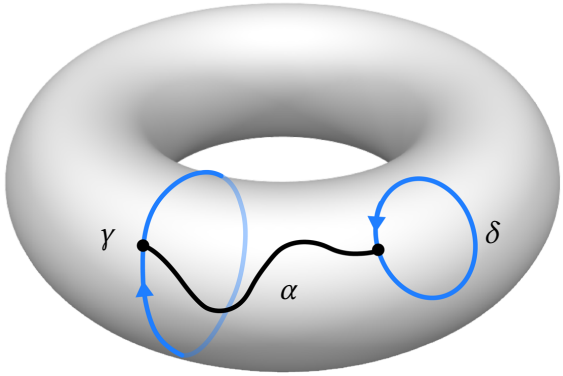


Figure 3.2: Illustration for Proposition 5 concerning quasi-additivity of holonomy numbers on loops.

Proposition 5 (Quasi-Additivity)**.** *Suppose γ and δ are two non-intersecting simple oriented loops and α is a path from the right side of γ to the right side of δ that only intersects these loops at its endpoints and does not contain any cones (see Figure 3.2). Then (provided the mesh is suitably refined) there is a simple loop γ_0—which can be made arbitrarily close to δ, γ, and α—such that*

$$k^F_{\gamma_0} = k^F_\gamma + k^F_\delta - 1$$

If α is a path from the left side of γ to the left side of δ we have nearly the same result, but the holonomy number of γ_0 is instead given by

$$k^F_{\gamma_0} = k^F_\gamma + k^F_\delta + 1$$

A proof and an illustration can be found in Section A.1.

Rerouting around a Cone. In particular, for a cone v_i, provided there is a path from the right side of γ to v_i^*, the above proposition tells us there is a loop γ_0 such that $k^F_{\gamma_0} = k^F_\gamma - I^F_{v_i}$. We refer to the construction of the latter loop as *rerouting γ around v_0* (with a counterclockwise orientation).

On the other hand, if there is a path from the left hand side of γ to v_i^* (the dual facet of vertex v_i), the above proposition gives us a loop γ_0 such that $k^F_{\gamma_0} = k^F_\gamma + I^F_{v_i}$, which we refer to as rerouting γ around v_i with a clockwise orientation. Moreover, it is clear from the construction of these loops that γ_0 is homotopic to γ on M, although the homotopy will necessarily cross the cone as otherwise the holonomy numbers of the two loops would be the same. Figure 3.3 shows an example of rerouting a loop around a cone of index $\frac{1}{4}$ twice, so as to yield a loop whose holonomy number differs by $\frac{1}{2}$.

These observations lead to the following key proposition.

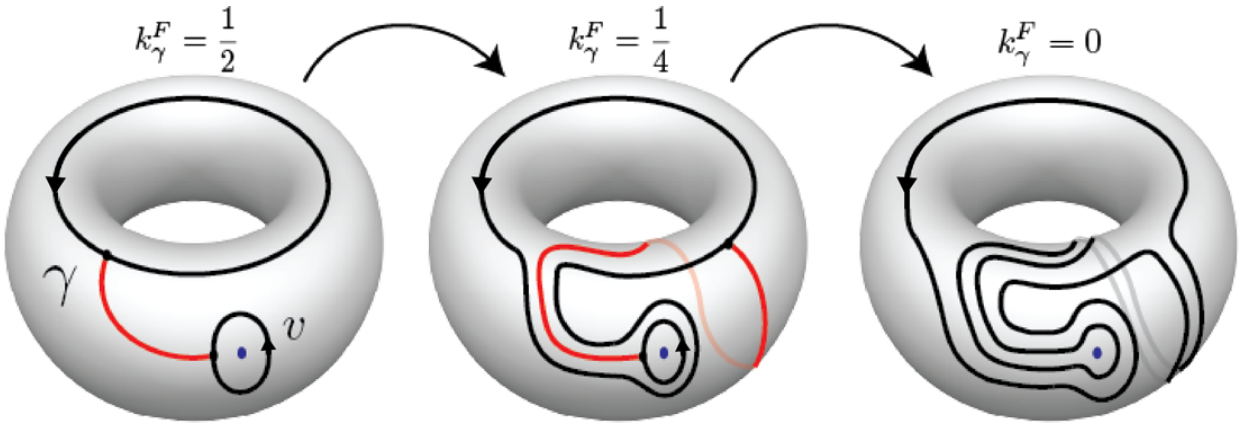


Figure 3.3: Rerouting (ccw, twice in a row) of a loop around a cone of index $\frac{1}{4}$.

Proposition 6. *Let $H = \{\gamma_1, ..., \gamma_{2g}\}$ a basis of loops for M that cuts M into a topological disk, and let $v_1, ..., v_m$ be vertices of M. Also, let $k_1, ..., k_{2g}, I_1, ..., I_m \in \frac{1}{4}\mathbb{Z}$ and assume $\gcd(I_1, ..., I_m) = \frac{1}{4}$. Then there is another basis of loops $\delta_1, ..., \delta_{2g}$ that cuts M into a disk such that $k^F_{\delta_i} = k^F_{\gamma_i} + k_i$, $i = 1, ..., 2g$, for any seamless parametrization F that has cones with indices $I^F_{v_j} = I_j$ at the vertices $v_1, ..., v_m$.*

For a constructive proof see Section A.2. Conceptually, we can reroute the loops γ_i one-by-one around suitable subsets of the cones in a manner that preserves the topology of H, such that their holonomy numbers change exactly by the desired values k_i.

For the purpose of our method, this result means that we can start from a cut graph formed by the union of $2g$ loops $\gamma_1, ..., \gamma_{2g}$, and modify these loops using an appropriate choice of inte-

gers $k_1, ..., k_{2g}$ to yield loops $\delta_1, ..., \delta_{2g}$ instead, with any holonomy numbers we want, forming an equivalent signature—under the only condition that $\gcd(I_1, ..., I_m) = \frac{1}{4}$. This ability is sufficient for the method presented in the following to construct a seamless parametrization with the desired holonomy, which constructively shows existence, under the above condition.

GCD-Condition. This condition is obviously satisfied as soon as there is even just one cone of index $\pm\frac{1}{4}$ among all prescribed cones. But this (practically very mild assumption) is not even necessary; even if all indices are of higher magnitude, they may have a greatest common divisor of $\frac{1}{4}$. If the gcd is indeed larger than $\frac{1}{4}$ (a potentially realistic scenario is one with indices restricted to multiples of $\frac{1}{2}$), note that while not all holonomy numbers can be achieved by rerouting, it may still be possible to achieve those desired.

3.4 Holonomy Signature

We consider a closed orientable manifold mesh M of genus g and a cut graph G on M that cuts M to one or more topological disks. We let M^c denote the resulting cut mesh, which has a canonical map $\pi : M^c \to M$ that is the identity on the interior and maps exactly two boundary edges in M^c to each edge in $G \subset M$. We call two edges e, e' in the boundary of M^c *mates* if $\pi(e) = \pi(e')$. We also define a *loop* as an oriented closed walk of facets (or dual vertices) of M and a *simple loop* as an oriented cycle of facets (or dual vertices).

Definition 3.1 (Seamless Parametrization)**.** A discrete seamless parametrization, as in [Myles and Zorin 2013], is a continuous piecewise linear, locally injective map $F : M^c \to \mathbb{R}^2$ such that for any boundary edge e with mate e', there is a rigid transformation $\sigma_e(x) = R_e x + t_e$, where R_e is a rotation by an integer multiple of $\frac{\pi}{2}$, that maps $F(e)$ to $F(e')$, i.e. $\sigma_e(F(e)) = F(e')$.

A seamless parametrization naturally induces a discrete metric $E \to \mathbb{R}^{>0}$ on M^c by letting the length of an edge e of M^c be the length of $F(e) \subset \mathbb{R}^2$. Since mated edges e, e' in the boundary

of M^c are related by a rigid transformation, $F(e)$ and $F(e')$ have the same length, so this metric extends to a well-defined metric on M. Moreover, the metric on M is flat except at isolated vertices $C = \{v_1, ..., v_m\}$ in G, i.e. in the boundary of M^c, called *cones*.

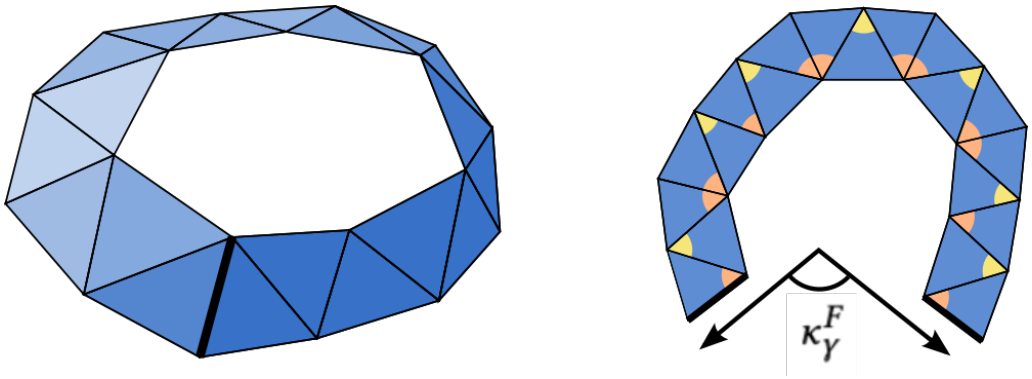


Figure 3.4: The holonomy angle κ_γ^F (Theorem 3.2) of a dual loop (cyclic triangle strip) under a metric F is the sum of signed inner angles (yellow and orange). Up to multiples of 2π (if the loop makes multiple turns) this corresponds to the angle between first and last edge when laying out the strip in the plane.

Definition 3.2 (Holonomy Angle)**.** For a loop γ, the holonomy angle (or discrete geodesic curvature [Crane et al. 2010]) of γ under a seamless parametrization F is

$$\kappa_\gamma^F = \sum_{f^* \in \gamma} \alpha_\gamma(f^*),$$

where f^* is the vertex dual to facet f, $\alpha_\gamma(f^*)$ is the signed angle of $F(f)$ at the vertex of f incident to the preceding and succeeding facets in the loop. The sign is positive (negative) for vertices on left (right) hand side of the loop. See Figure 3.4 for an illustration.

Let v^* denote the dual facet of vertex v, and ∂v^* the cycle of this dual facet's dual vertices. In other words, ∂v^* is the loop around the single vertex v.

Definition 3.3 (Holonomy Number)**.** For a loop γ, we define the *holonomy number* as $k_\gamma^F = \kappa_\gamma^F / 2\pi$. In the case of a vertex-loop, $\gamma = \partial v^*$, we additionally define the *index* of the vertex as $I_v^F = 1 - k_{\partial v^*}^F$.

Since the images of edges in the boundary of M^c under F are related by rotation angles that are

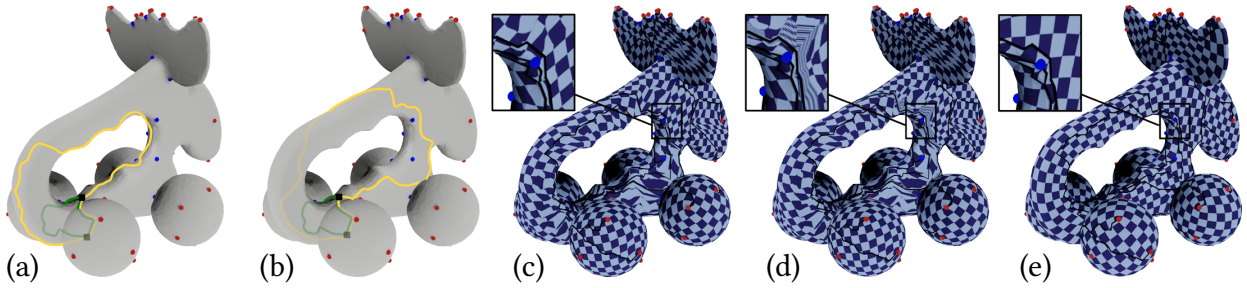


Figure 3.5: Algorithm overview: (a) Example input signature loops (yellow and green) and cones (red and blue). (b) Loops of an equivalent signature obtained by strategically modifying this input; notice that the yellow loop takes a different path between the cones. (c) Conformal parametrization respecting the prescribed cones and aligned with the cut graph that is formed by the loops; due to this alignment, it has a specific holonomy pattern along the loops. (d) The map is modified by parametric padding to make it seamless while preserving its holonomy properties. (e) Finally, the map can be continuously optimized for low distortion and possibly cross field alignment, naturally within its topological class.

integer multiples of $\frac{\pi}{2}$, the holonomy numbers are always integer multiples of $\frac{1}{4}$.

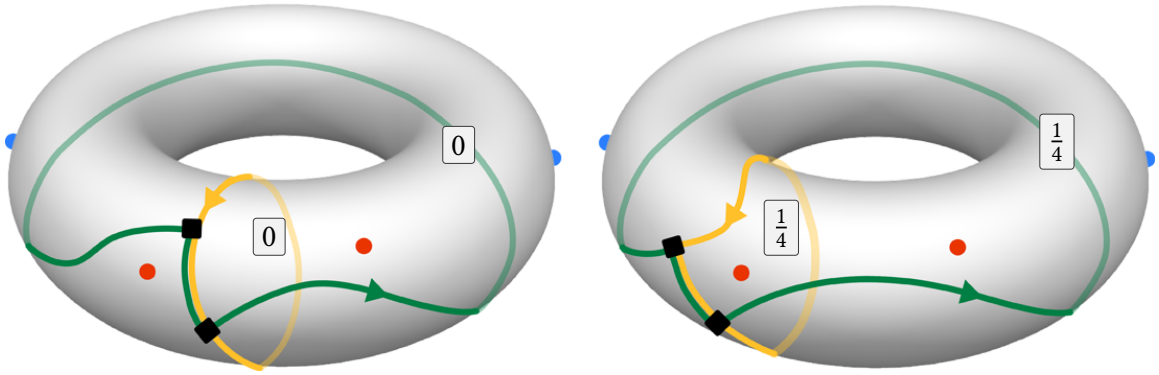


Figure 3.6: Example of two equivalent holonomy signatures. Red and blue cones have index $-\frac{1}{4}$ and $+\frac{1}{4}$, respectively; the holonomy numbers of the green and yellow loops are indicated. Note that from left to right, the loops are essentially deformed across a cone (the leftmost red cone), and this affects the loops' associated holonomy numbers accordingly.

Definition 3.4 (Holonomy Signature)**.** We call the holonomy numbers for a choice of homology basis loops $\gamma_1, ..., \gamma_{2g+m}$ of $M \setminus C$ the holonomy signature of F. A natural choice of basis is a homology basis $\gamma_1, ..., \gamma_{2g}$ of M together with cone vertex loops $\partial v_1^*, ..., \partial v_m^*$. The loops are referred to as signature loops.

A homology basis of M can, for instance, be chosen as a so-called *system of loops* [Erickson and Whittlesey 2005].

Importantly, such a finite holonomy signature completely captures the holonomy number of *any* loop: First, since the metric is flat except at cones, any two loops homotopic in $M \setminus C$ (i.e., loops that can be continuously deformed into each other within M without crossing a cone vertex) have the same holonomy number (cf. Prop. 1 in [Myles and Zorin 2012]). Second, given an arbitrary loop γ and a homology basis of $M \setminus C$, there is (by the nature of a homology basis [Hatcher 2002]) a surface $\bar{M}$ so that its boundary is composed of γ and some combination of the basis loops. The Gauss-Bonnet theorem then gives a formula for the holonomy number of this boundary in terms of the Euler characteristic of $\bar{M}$. Thus, the holonomy number of γ is determined by the holonomy numbers of the loops of a homology basis (cf. Prop. 2 in [Myles and Zorin 2012]). Figure 3.5 (a) shows an example of signature loops: the green and yellow loops form a homology basis of M, while the small cone vertex loops are visualized as red and blue dots at the respective vertices.

Note that the holonomy signature is not unique, neither its loops nor its numbers (Figure 3.6). For a fixed seamless parametrization F, a different choice of signature loops will lead to different associated holonomy numbers—even though the same parametrization topology is represented. Such holonomy signatures are called *equivalent*.

3.4.1 Relation to Cross-Fields

Seamless parametrizations are often employed in conjunction with cross-fields, most importantly when parametrizations are built and optimized for directional alignment with such a field. In such cases it is important for field topology and parametrization topology to match. In this context, there is a close connection between the holonomy of a seamless parametrization and the turning numbers of a cross-field.

For a smooth surface S, a cross-field $\mathbf{d}$ is a differentiable mapping of four tangent vectors to each point $p \in S$ (except at isolated singularities) that are invariant to rotation by $\pi/2$ around the normal $\hat{n}_p$ to S at p. Given such a field and a loop γ on the surface S, the field will make some

number of rotations along this loop, and due to the rotational symmetry of the field these turning numbers T_γ can be integer multiples of $\frac{1}{4}$.

Moreover, as shown in [Ray et al. 2008], there is a discrete analogue of cross-fields and turning numbers for triangle meshes, and turning numbers along loops satisfy a theorem analogous to the Poincaré-Hopf theorem for vector fields that states $T_{\partial S} = -\chi(S)$. This implies that the holonomy number of the boundary of a flat surface, which does not contain any cone, is the same as the turning number of a singularity-free cross-field along that boundary. Hence, if turning numbers of a cross field agree with holonomy numbers on a set of signature loops then they will agree for *any* loop on the surface. Consequently, by taking as our desired holonomy numbers the turning numbers of a given cross-field on a homology basis of $M \setminus C$, where C is the set of the cross-field's singularities, the seamless parametrization is fully constrained to topologically match the input cross-field—in terms of local (cone indices) as well as global behavior.

3.5 Approach Overview

Given a holonomy signature (or a cross-field implying a holonomy signature, cf. Section 3.4.1), consisting of loops associated with a holonomy number each, on a surface M, our goal is to construct a valid seamless parametrization F for M that respects this signature. In Section 3.3 we discuss the question for which signatures this is actually feasible.

Key Idea. We show in Section 3.3 that, given a holonomy signature, we can find an equivalent signature (by exchanging or modifying the signature loops) such that the associated holonomy numbers assume almost any desired values. Essentially, we are exploiting the above mentioned non-uniqueness of the signature (cf. Figure 3.6). We make use of this algorithmically in Section 3.6 in the following way: The SP method [Campen et al. 2019] enables constructing a seamless parametrization that has prescribed local holonomy (i.e. cones), but it lacks the ability to

prescribe holonomy globally. However, its result has not a random but a fixed holonomy pattern along the cut graph that is used in the construction. We therefore modify the $2g$ global signature loops such that their union forms a cut graph and such that their corresponding holonomy numbers in an equivalent signature match exactly the fixed pattern that SP will produce. Figure 3.5 illustrates the main steps of our construction process.

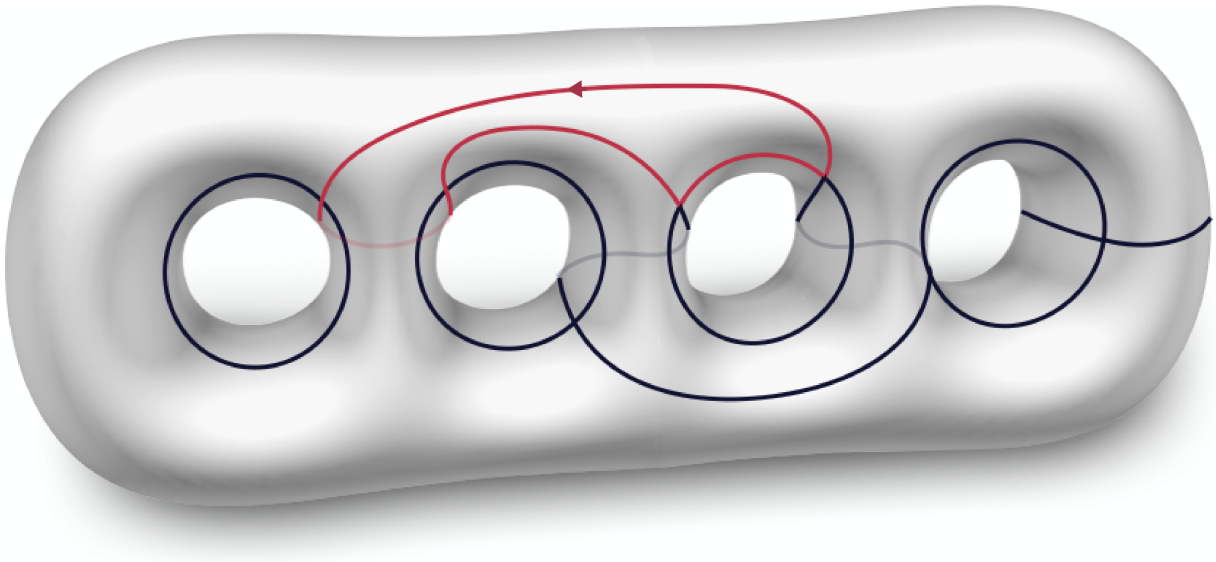


Figure 3.7: A hole-chain cut graph G, as used in [Campen et al. 2019]. As an example, the contained loop that is highlighted in red, because it makes two left turns (in ccw sense), will have holonomy number $\frac{2}{4}$ in the parametrization constructed by that method.

Cut Graph. The seamless parametrization construction by SP relies on using cut graphs with certain structural restrictions, so-called hole-chains (or modifications thereof). Figure 3.7 shows an example, details follow in Section 3.6.1. Let G be such a cut graph. Let H be a *system of loops* of M, i.e., H is a homology basis and cuts M into a topological disk. In particular, as G cuts M into a topological disk, H can be chosen such that the (non-disjoint) union of its loops equals G (Section 3.6.2).

Fixed-Holonomy Parametrization. By construction, SP will yield certain predetermined holonomy numbers along G, thus on these loops H, regardless of the loops' geometry. More concretely,

each branch point of the hole-chain G has four incident cut segments, conceptually forming a cross. When following a loop through G, at such a branch point it may therefore take a left-turn, a right-turn, or continue straight. The generated parametrization's holonomy number along any loop in G is simply the (signed) difference between its number of right-turns and its number of left-turns (times $\frac{1}{4}$). This is due to the SP -parametrization being aligned to all segments of G, i.e., they are geodesic under this metric, and the corners between segments at branch points form right angles under this metric.

Cut Graph Rerouting. Given target holonomy numbers on the loops of H (e.g., derived from an input cross-field), by Proposition 6 we can modify the loops of H, yielding H', such that their target holonomy numbers equal their left-right-turn balance. Conceptually, this rerouting of loops is described in Section 3.3; in Section 3.6.3 we describe algorithms that practically implement it. As this rerouting does not alter the loops' intersections, i.e., it preserves the branch points of G and the loops' left-right-turns, the union of loops from H' still forms a hole-chain, which we can then use as prescribed cut graph for the parametrization construction.

Holonomy Soft-Guidance. In order to yield parametrizations that are not just topologically correct but also (already initially, before distortion optimization) of reasonable geometric quality, we aim to reduce the need for cut graph rerouting as much as possible. To this end we construct the individual paths that the initial hole-chain G is made of in a holonomy-guided (e.g. cross-field guided) manner (Sections 3.6.1.1 and 3.6.1.2). This promotes hole-chains that largely have the desired holonomy properties right away.

3.5.1 Algorithmic Outline

Our method's overall algorithmic pipeline can be outlined as follows:

1. Construct Cut Graph

- Initial field-guided hole-chain G (Section 3.6.1)
- Extract loop basis H of G (Section 3.6.2)
- Reroute $H \rightarrow H'$, yield G' (Section 3.6.3)

2. Construct Seamless Parametrization
   - Construct G'-aligned mapping $f : M^c \rightarrow \Omega$ (Section 3.7.1)
   - Pad f to yield seamless map $f' : M^c \rightarrow \Omega'$ (Section 3.7.2)
   - Optimize f' and Ω', maintaining seamlessness (Section 3.7.3)

3.6 Holonomy-Constrained Cut Graph

The cut graph with particular holonomy pattern is built in three steps. We start by constructing a hole-chain G; in deviation from the algorithm described for this purpose in [Campen et al. 2019] we employ soft-guidance by a given input cross-field already in this step. Afterwards a holonomy basis of loops H is extracted from G, and its associated target holonomy numbers are derived from the cross-field. Finally, these loops are rerouted where necessary, i.e., where soft-guidance did not yield exactly those holonomy numbers we require for the subsequent stage.

3.6.1 Field-Guided Hole-Chain

A hole-chain G on M is built out of g loops (non-contractible, non-separating, non-homotopic) and $2g - 1$ connecting paths. Intuitively, cutting the surface by the g loops yields a topological sphere with $2g$ holes, and the $2g - 1$ paths connect these in a chain-like manner, further cutting the surface to a topological sphere with one hole, i.e. a disk. Figure 3.7 shows an example with 4 loops (circular, inside the tunnels) and 7 connectors. We here describe how to construct these

loops and connectors guided by a given cross-field.

Remark:. For certain special cases (genus $\leqslant 2$) a slightly modified hole-chain structure needs to be chosen. This is done exactly as in the SP method. Likewise, an extra connector path possibly needs to be added; this occurs after rerouting (Section 3.6.3).

3.6.1.1 Field-Guided Loops

We construct g non-contractible, non-separating, non-homotopic loops on $M \setminus C$ following the algorithm of [Diaz-Gutierrez et al. 2009]. To promote cross-field alignment (thus turning number zero along the loop) we employ the field-alignment metric of [Campen et al. 2012] in this process. This in particular means that the loop construction is performed on M_4, a four-sheeted covering of M, owing to the four different directions a cross-field specifies per point.

A loop resulting from this, while simple (i.e. intersection-free) on M_4 by construction, may in some cases be self-intersecting when projected down onto M. Conceptually, it may pass "over" itself on a different sheet of M_4, which corresponds to an actual crossing on M. In such a case we fall back to a non-guided construction of a replacement loop directly on M.

3.6.1.2 Field-Guided Connectors

Connector paths between the loops are selected in a Hamiltonian path manner as in the SP method. By contrast, however, we do not build these from simple shortest paths but again in a field-guided manner. As in the loop construction in Section 3.6.1.1 we use an anisotropic metric and perform the path search on M_4. As the above loops have an embedding in M_4 by construction, we know on which sheet of M_4 to start and end the search, respectively: one sheet lower or higher than the respective loop, as a connector will be orthogonal (rather than parallel) to its two incident loops under the final parametrization. Again, should a self-intersecting connector path occur, we fall back to a shortest path computed on M.

Remark:. Loops constructed by the fallback method (if any) have no native embedding on M_4. We locally (at the intended connector start or end point) assign them to the sheet on which the field direction best fits the loop's tangent, so as to have a reasonable setup for the computation of incident connectors. In any case, however, let us remind that the worst possible outcome of a locally suboptimal choice is a hole-chain that requires some more rerouting—structural correctness is not at stake in this soft-guided approach.

The resulting loops and connectors are embedded in edges of M and together form the discrete cut graph G.

3.6.2 Homology Basis Extraction

We construct a homology basis H of M in the form of $2g$ loops contained in the cut graph G. To this end we compute a spanning tree T in G. The remainder $G \backslash T$ consists of $2g$ edges, called bridges [Erickson and Whittlesey 2005]. For each bridge, its union with the two paths from its incident vertices to the root of T is a loop, and these $2g$ loops form a homology basis, and more specifically a system of loops.

Note that these loops may coincide partially. Each of the $2g$ bridge edges, however, is part of exactly one of these loops only. By rerouting the segments of G that contain these bridge edges (called bridge segments) we are therefore able to *individually* alter the holonomy number or turning number of each of these $2g$ loops with respect to a given field. Effectively, the bridge segments are the places where the conceptual α-path from the proof of Proposition 6 can be attached without intersecting any other basis loops. Ultimately, a modified cut graph G' with the desired holonomy number for each basis loop can be obtained in this way, as detailed in the following.

3.6.3 Segment Rerouting

For each loop of H we count its number of left turns m_l and right turns m_r (in ccw sense). The holonomy number along this loop in the parametrization we will construct will be $\frac{1}{4}(m_l - m_r)$ (cf. Figure 3.7). If its target holonomy number is t (e.g., the cross-field turning number along this loop), we need to reroute this loop such that this number changes by $k = \frac{1}{4}(m_l - m_r) - t$.

This is performed by rerouting the loop's bridge segment, which we tackle in a two-tier manner. We first attempt to find a replacement path for the segment by an efficient field-guided method, detailed in Section 3.6.3.1. As this method is not guaranteed to yield a *simple* path (which, however, is needed), where necessary a guaranteed (but less geometry aware) fallback strategy is employed, as described in Section 3.6.3.2. Note that the resulting loops are not unique; many different equivalent signatures exhibit the desired holonomy numbers. Due to equivalence, however, the final parametrization's topology is not affected by this (see Figure 3.8).

Remark:. Optionally, we may perform a pre-rerouting of the field-guided loops from Section 3.6.1.1 already before moving on to the connector computation. This is possible because these loops' target holonomy is known to be zero, regardless of how the connectors will interact with them. This pre-rerouting is not necessary for correctness, but empirically it reduces the total amount of rerouting required. As the g loops do not yet form a complete cut graph that cuts the surface to a disk, one needs to take one additional precaution, though, so as to ensure that a loop is rerouted homotopically. Namely, we cut M minus the loops to a disk by additional temporary cuts (using the method of [Erickson and Whittlesey 2005]) and perform rerouting within this disk.

3.6.3.1 Holonomy-Aware Dijkstra

Given a bridge segment ℓ of G, supposed to be rerouted such that the holonomy number of its unique containing loop γ from H changes by k, we employ a holonomy-constrained Dijkstra's

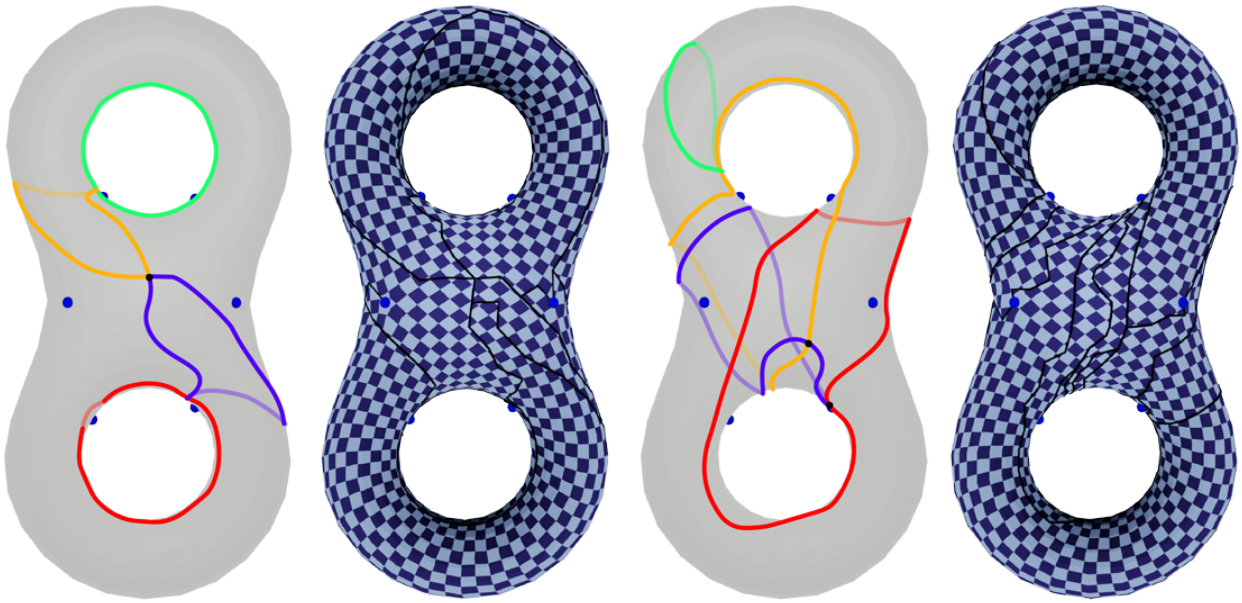


Figure 3.8: Two equivalent holonomy signatures, based on different signature loops; the different associated holonomy numbers are not shown in the figure. Both are the result of rerouting so as to achieve the required holonomy pattern, therefore the resulting optimized seamless parametrizations based on the cut graphs formed by these loop systems are identical (up to seamless transformation, due to a differently located cut graph).

algorithm, as described in [Campen et al. 2019, §5.1]. This entails the following. We compute a spanning tree of M^c (cut by the current G), rooted on ℓ. Indices of cone vertices are propagated through this tree towards the root, and tree edges are marked with the sum of indices propagated through them. By then keeping track of the sum of these values of edges crossed during Dijkstra's shortest path algorithm (applied to the dual mesh), we can read off the index sum of cones enclosed between a Dijkstra path ℓ' and bridge segment ℓ. The algorithm terminates when a path enclosing the desired index sum (which determines the change to the holonomy number of γ) is found.

Similar to the path construction on M_4 described in Section 3.6.1, this holonomy-constrained path search effectively occurs on an (in this case infinite) cover of M (akin to the universal cover of $M \setminus C$). Consequently, a non-simple path (in M) can be the result in some cases. The following guaranteed fallback strategy takes care of such cases.

3.6.3.2 Fallback Strategy

By following the rerouting construction used in the proof of Proposition 6, a proper simple replacement path for a bridge segment ℓ can safely be found. Let v be a vertex on a bridge segment ℓ whose loop's holonomy number needs to be increased by k.

Assume for a moment that there is a cone vertex v^* with $|I(v^*)| \leqslant k$. Let α be the shortest path from cone v^* to v—either meeting ℓ from the right if I and k have opposite sign, or from the left in the case of equal sign. Among all suitable cones, we choose the one for which the path α is shortest, so as to reduce the amount of modification.

Let the two vertices on ℓ which are directly adjacent to v be v^- and v^+. Closely following the conceptual rerouting from Figure A.1 we remove the edges v^-v and vv^+ from ℓ and replace them by the path from v^- to v^+ tightly along α and around v^*. Where necessary we split edges of M to make room so that this path does not touch any other part of G.

This changes the loop's holonomy number by $I(v^*)$ or $-I(v^*)$, depending on which side of ℓ the path α connects to. This procedure can be repeated as long as the remaining difference $k \leftarrow k \pm I(v^*)$ is not zero yet.

While this strategy proved sufficient in all practical test cases (cf. Section 3.8) (and indeed is guaranteed to work if there is at least one cone of index $\pm\frac{1}{4}$), in general a greedy selection of reroute-cones v^* in this greedy manner is insufficient. Instead, let V be a multiset of cones such that its index sum is k. Under the GCD-condition (Section 3.3), V exists, as also exploited in the proof of Proposition 6. By selecting the cones from V as v^* in the above one after the other (also considering multiplicity), the desired result is achieved. A suitable multiset V, i.e. a subset of cones and multiplicities, is easily computed using the Extended Euclidean Algorithm. The incorporation of distances also in this general case would enable smaller rerouting modification,

but given the practical irrelevance of this multiset-case, the effort would hardly pay off.

Remark:. In practice we tentatively perform the fallback-rerouting starting from multiple root vertices v (sampled equidistantly on the bridge segment; we use 10 samples in our experiments) and retain the result of shortest length to reduce the complexity of the final cut graph.

3.7 Seamless Parametrization

Once the final, i.e., rerouted and possibly extended (recall the remark in Section 3.6.1), cut graph G' is available, the next step is to construct a domain that is compatible with the cut surface M^c and suitable to serve as parameter domain for a seamless parametrization of M. The domain shape is derived from a conformal metric computed on M^c with prescribed cones and prescribed boundary curvature.

3.7.1 Cut Graph aligned Metric

A key role in the SP method that we build on is played by a discrete conformal metric computation on the cut mesh M^c. While the conformal metric algorithm from [Campen and Zorin 2017b] that is used in SP works adequately in most cases, it does not provide strict guarantees of convergence. The cut graph rerouting used in our method can sometimes lead to rather complex cut shapes, thus boundary shapes of M^c, implying additional metric distortion, making the problem instances particularly challenging.

Very recently, a novel algorithm for discrete metric computation with prescribed (boundary and cone) angles has been proposed [Campen et al. 2021b; Gillespie et al. 2021], based on mathematical insights [Gu et al. 2018b; Springborn 2020] that guarantee convergence. We employ an implementation based on this work.

Using this algorithm we compute a discrete metric (i.e., edge lengths) for M^c, prescribing the angles of cone vertices and the geodesic curvature on the boundary, i.e., along the cut G'. Concretely, the segments of G' are constrained to be straight under the resulting metric, and the corners (at branch points of G') are constrained to be right.

3.7.2 Padding

Under the computed metric the two boundary segments of M^c corresponding to a common segment of G' are straight, their mutual angle is a multiple of $\pi/2$ (as required for seamlessness), but their lengths may differ. The SP method uses so-called *padding*, i.e., parametrically stretching out strips of the surface under the metric along the boundary segments so that the boundary segments' lengths expand to equalize the lengths of all paired segments, and this provably is always possible.

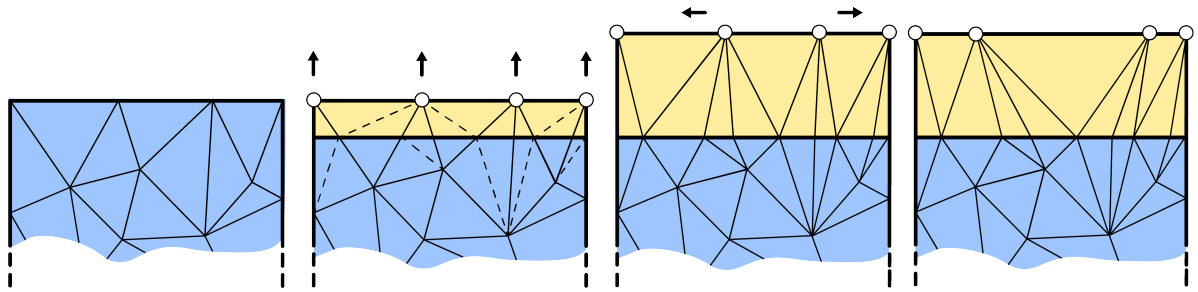


Figure 3.9: Illustration of padding operation (in parameter domain). A thin strip along the top straight cut segment (with no interior vertices) is stretched in vertical direction by its required padding width. Then, vertices are shifted horizontally to match their mates across the cut.

The following steps perform this padding, analogous to the original SP method from [Campen et al. 2019]:

1. Add cuts from all cones to the boundary of M^c. Make sure each boundary segment is reached in at most one point. Around each interior vertex the angles under the metric from Section 3.7.1 now sum up to 2π, i.e. the cut surface is flat.

2. Lay out this flat mesh in the plane, i.e., assign (u, v)-parameters to all vertices of the mesh, e.g., in a breadth-first traversal. The global rotation is chosen such that the straight boundary segments are aligned with u or v axis directions. This yields the (non-seamless) domain Ω.

3. For each straight boundary segment, compute the amount of padding (width w_i) required to equalize parametric lengths of the paired segments, using the equation system of the SP method.

4. Along each segment, determine a parametrically rectangular strip free of cones, and make the mesh conform to this strip by inserting the strip's boundary line by splitting. Then apply a stretch transformation to the (u, v)-coordinates inside the strip, so as to shift the segment in perpendicular direction by its padding width w_i (Figure 3.9).

5. In cases where the cut graph has cut the surface into more than one disk, glue these together parametrically along pairs of boundary segments by means of rigid transformations applied to the (u, v)-coordinates to finally obtain a map F' onto a single connected domain Ω'.

3.7.3 Optimization

As a final step, we optimize the established map for reduced distortion. As objective, we employ a local cross-field (orientation and sizing) alignment energy E_A [Bommes et al. 2009] and add (with a small factor of $s = 10^{-3}$) the symmetric Dirichlet energy E_D [Rabinovich et al. 2017a], which contributes its barrier behavior to prevent parametric inversions in the course of optimization. Linear constraints are added to preserve seamlessness. We use a projected Newton solver and use an explicit triangle inversion check in the line search [Smith and Schaefer 2015a], using exact predicates, to reliably maintain local injectivity. We experimentally discovered that using an unconstrained Newton optimizer over the set of independent variables computed using a reduced

row echelon form of the constraint matrix is numerically more stable than solving a KKT system at each iteration, leading to faster convergence.

We emphasize that we do not aim to address map optimality here; our focus is on constructing a topologically correct initial map, subject to further improvement geometrically.

3.8 Evaluation

We apply our method to a dataset of 3D models with cross-fields [Myles et al. 2014]. We restrict ourselves to models of genus > 0, as on topologically trivial surfaces there are no global holonomy aspects to account for. The method succeeds in generating a cut graph with exactly the needed holonomy numbers in all cases. As all cases satisfy the gcd=$\frac{1}{4}$ condition, and all crucial operations are combinatorial/discrete, the general success of this step is indeed to be expected. For each model, Table 3.1 lists the number of rerouting operations that our method performed.

The construction of a seamless parametrization based on this cut then succeeds in most cases; in six, however, the initial metric distortion is very high, causing subsequent steps (padding or the simple distortion optimization approach) to get into numerical trouble. In Figures 3.1 and 3.10 obtained optimized seamless parametrizations for examples from the dataset are shown, matching the input cross-field by construction. Table 3.2 reports the final distortion of these.

3.8.1 Comparison

To demonstrate the importance of our contribution in the context of guaranteed locally injective seamless parametrization construction, we also apply the bare SP method of [Campen et al. 2019] (which takes local holonomy (cones) but not global holonomy into account) to these models.

While SP is able to respect the singularities of the prescribed field by construction, whether or

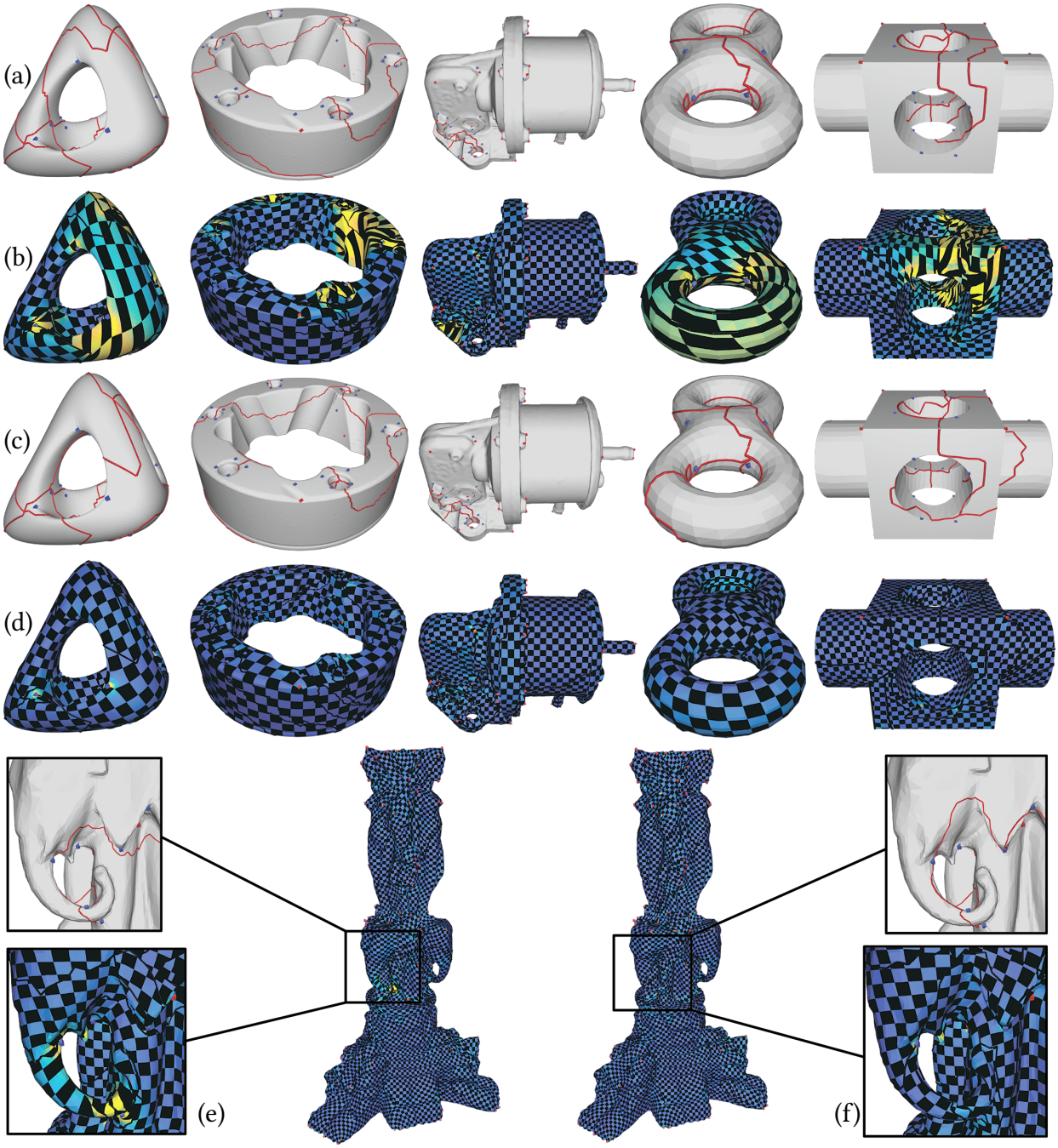


Figure 3.10: Comparison of seamless parametrizations on surfaces of non-trivial topology, computed by the bare SP method [Campen et al. 2019] (row b, e) and by our method (row d, f). The used cut graphs are shown in red, the initial hole-chain used for SP (row a, e) and the rerouted version used by our method (row c, f). Notice their topologically differing structure (i.e. they wind around some handles or cones differently), as well as the higher distortion of the results by the bare SP method due to being unable to properly align to the underlying smooth cross-field for topological reasons. Notice that this distortion cannot be reduced further by continuous optimization; there are topological obstacles.

not its resulting map matches the cross-field topologically is essentially a matter of chance. If the cross-field is very smooth (as generally is the case in this data set) and the cut graph for the map is constructed from certain shortest paths, the chance of a match may be higher than that of any particular mismatch. Nevertheless, we encounter a mismatch for a large number of models—in line with the fact that, as can be seen in Table 3.1, our method had to employ at least one rerouting operation in the majority of cases. In case of a mismatch, the resulting map cannot continuously be optimized to achieve reasonable alignment between map isolines and the field, as there is a topological obstacle. This can be observed in Table 3.2, where the remaining final distortion is significantly higher when not employing rerouting. The difference is also illustrated in Figure 3.10. Our method, in essence by adjusting the cut graph in the described manner, ensures a topological match between the signature induced by the cross-field and the signature of the generated seamless parametrization.

3.9 Conclusion and Future Work

We have explored the relation between cross-fields and seamless surface parametrizations (and therefore quadrangulations) on a topological level. A key insight is that there are hardly any practically important obstacles to generating a seamless parametrization (or quadrangulation) that topologically matches a given cross-field. We have described a method to generate such a seamless parametrization, given an input cross-field or an abstract topological specification in form of a holonomy signature. It is based on a variation of the SP method [Campen et al. 2019], with the main difference being:

- The initial hole chain cut graph is constructed taking cross-field guidance into account.
- The hole chain is then modified by extracting a loop basis and rerouting of loop segments based on our theory.

- The generation of a cut-aligned parametrization is performed using a different, theoretically sound conformal mapping method.

From the SP method that we employ for the parametrization construction we inherit the restriction to surfaces without boundary. While there are no fundamental obstacles to adding boundary support to our rerouting procedure, padding feasibility requires additional theory in this more general context. The situation regarding support for alignment to feature curves, which is of interest in some use cases of seamless parametrizations, is very similar.

The algorithm stage described in Section 3.6, in particular the holonomy-constrained cut graph generation using rerouting, relies on discrete operations and therefore is not only sound theoretically, but can be executed without the risk of numerical issues and limits in practice. The algorithm stage described in Section 3.7 (initial parametrization followed by constrained optimization), by contrast, involves numerical computations, with consequent limits in practice. While for initial parametrization a discrete approach is imaginable [Zhou et al. 2020], at least for the final distortion optimization a numerical approach is inevitable.

While we observe the choice of loops that form the initial cut graph to not affect the final result conceptually (Figure 3.8), the distortion of the initial parametrization, and therefore the numerical challenges in the final optimization, can depend strongly on this choice. By testing various random root placements for the loop construction [Diaz-Gutierrez et al. 2009] employed in Section 3.6.1.1, initial parametrizations of low distortion could be found, but a more direct approach—or a more resilient final distortion optimization technique—is desirable.

The GCD-condition asserts that, for any given signature, there is an equivalent signature whose loops have any desired set of holonomy numbers. It therefore is a sufficient condition for the existence of a seamless parametrization that topologically matches a given signature. It is not necessary, though. While likely of limited practical relevance, the exploration of even tighter

conditions may be interesting.

Table 3.1: Statistics about the number of cut segment reroutings performed. It is further split into the numbers of field-guided and fallback reroutings.

Model	Genus	#Reroutings	#Field-Guided	#Fallback
twirl	1	0	0	0
robocatdeci	1	0	0	0
knot1	1	0	0	0
holes3	3	0	0	0
dancer2	1	0	0	0
sculpt	2	0	0	0
fertilitytri	4	0	0	0
rockerarm	1	1	1	0
genus3	3	1	1	0
elk	1	1	1	0
trimstar	1	1	1	0
wrench50K	1	1	1	0
bumpytorus	1	1	1	0
dancer25k	1	1	1	0
camel	1	1	1	0
dragonstandrecon	1	1	1	0
pulley	1	1	1	0
kitten	1	1	1	0
knot	1	1	1	0
mastercylinder	3	1	1	0
eight	2	1	1	0
femur	2	2	2	0
block	3	2	2	0
greeksculpture	4	2	2	0
elephant	3	2	2	0
thaistatue	3	2	2	0
oilpump	4	2	2	0
neptune0	3	2	2	0
carter	7	2	2	0
cup	2	2	2	0
botijo	5	3	3	0
chair	7	3	3	0
rollingstage	7	3	3	0
helmet	3	4	2	2
pegaso	6	4	4	0
chair	7	4	4	0
bozbezbozzel	5	5	5	0
dancingchildren	8	5	5	0
grayloc	9	6	6	0
seahorse2	8	10	5	5
raptor50K	10	12	6	6
heptoroid	22	15	14	1
gearbox	78	57	43	14
filigree	65	73	40	33
brain	57	83	70	13
vhskin	79	128	21	107

Table 3.2: Residual energy (normalized by surface area) for the models from Figures 3.1 and 3.10. The columns "without rerouting" correspond to the direct application of SP , without regard for global holonomy. From the last column the advantage in terms of field alignment and distortion becomes clear.

		with rerouting		**without rerouting**		
Fig.	**Model**	E_A+sE_D	E_A	E_A+sE_D	E_A	**ours/SP**
1	cup	0.0125	0.0125	0.3288	0.2882	3.8%
10	block	0.0136	0.0136	0.1115	0.1052	12.2%
10	eight	0.0350	0.0328	0.1524	0.1432	22.9%
10	genus3	0.0221	0.0208	0.1891	0.1747	11.7%
10	oilpump	0.0296	0.0293	0.0450	0.0370	65.7%
10	rollingstage	0.0127	0.0124	0.2666	0.1163	4.8%
10	thaistatue	0.0225	0.0225	0.0272	0.0257	82.7%

4 BijectiveRemesh: Maintaining Bijective Mappings for Data Transfer Across Remeshed Manifolds

Abstract

We introduce *BijectiveRemesh*, a robust algorithm for maintaining a continuous, bijective mapping across complex remeshing sequences on both 2D triangle surfaces and 3D tetrahedral meshes. Unlike traditional data transfer methods that rely on interpolation or projection, our approach constructs a mathematically rigorous composite map $f : \mathcal{M}_{\text{input}} \to \mathcal{M}_{\text{output}}$ by chaining local bijective atlases defined for each primitive operation.

Our framework represents the overall mapping as a composition of local bijective atlases, one per remeshing operation. Building upon successive self-parameterization (SSP) Liu et al. [2021], we introduce a *Shared Scaffold* structure for 2D triangle meshes that enforces global bijectivity through local orientation preservation. We extend this approach to handle edge splits, edge

swaps, and vertex smoothing beyond the original edge collapses. For 3D tetrahedral meshes, we generalize the local atlas construction using Steinitz's Theorem and Maxwell-Cremona lifting to ensure valid embeddings. This enables exact tracking of geometric entities—points, curves, and surfaces—across remeshing, with applications from texture transfer to volumetric simulations.

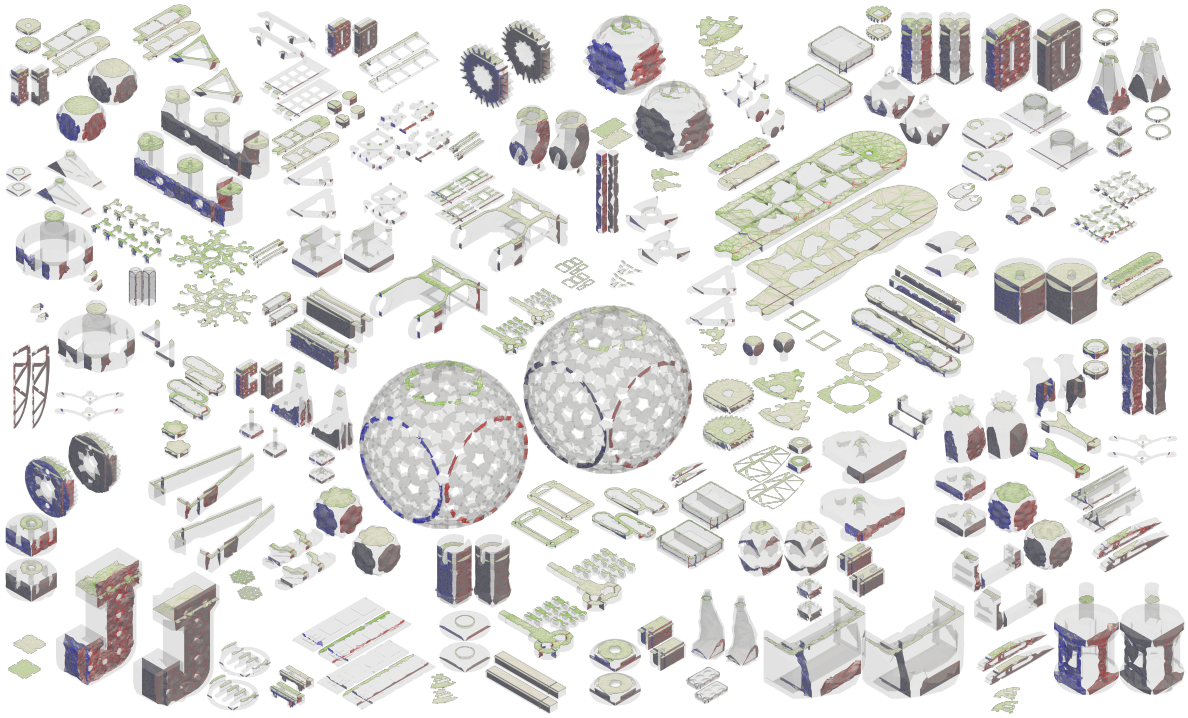


Figure 4.1: Bijective surface tracking through tetrahedral mesh simplification. We track axis-aligned planar surfaces (parallel to the xy, xz, and yz planes) through tetrahedral mesh simplification on models from the Thingi10K dataset Zhou and Jacobson [2016]. For each model, surfaces are sampled on the simplified output mesh $\mathcal{M}_{\text{output}}$ (right) and back-tracked to the original mesh $\mathcal{M}_{\text{input}}$ (left). Our bijective framework rigorously preserves the intersection topology among surfaces: where surfaces intersect on the output, they intersect consistently on the input, maintaining the combinatorial structure of intersection curves throughout the tracking process.

Contributions

- We introduce *BijectiveRemesh*, a framework that maintains a continuous, bijective map across complex remeshing sequences by composing local bijective atlases induced by atomic operations.

- For 2D triangle meshes, we develop local atlas constructions based on a shared scaffold that enforces bijectivity through local orientation preservation, and we extend prior SSP-style ideas beyond edge collapses to handle edge splits, edge swaps, and vertex smoothing.

- For 3D tetrahedral meshes, we generalize atlas construction using convex embedding arguments (via Maxwell–Cremona lifting), enabling robust, topology-preserving tracking of points, curves, and surfaces with applications to texture transfer and volumetric simulation.

Organization

We motivate the need for bijective correspondence under remeshing and discuss prior approaches (Section 4.2). We then present the overall formulation as a composition of per-operation maps (Section 4.3), followed by atlas constructions for triangle meshes (Section 4.3.1.1) and tetrahedral meshes (Section 4.3.1.2). Next, we describe how the composite map supports exact tracking of geometric entities (Section 4.3.2) and report experimental results and applications (Section 4.4), concluding with limitations and future work (Section 4.5).

4.1 Introduction

Remeshing is ubiquitous in computer graphics and scientific computing, encompassing tasks such as quality improvement, resolution adaptation, and artifact repair. A fundamental challenge across all these operations is *data transfer*: propagating scalar fields, texture coordinates, material attributes, or boundary conditions from the original mesh to the remeshed version with high fidelity. As meshes undergo decimation, refinement, or optimization, maintaining precise correspondences between initial and remeshed states is critical to prevent information loss and numerical diffusion.

Conventional solutions for maintaining correspondences between meshes [Kobbelt et al. 1998] rely on heuristics such as barycentric interpolation or closest-point projections [Botsch et al. 2010]. While effective for minor adjustments, these methods lack topological guarantees. Furthermore, when topological operations induce goes substantial changes (e.g., edge collapses or swaps), these heuristic correspondence often fail to be bijective, causing artifacts like UV seam tearing, texture deviation, or loss of feature lines. Constructing a global bijective parameterization could resolve these issues, but is computationally prohibitive and prone to failure on complex non-disk-topology shapes [Kraevoy and Sheffer 2004].

We propose a scalable framework that maintains strict bijective mappings throughout remeshing, enabling accurate data transfer between the original and remeshed states. Rather than computing a single global parameterization, our method encodes the overall map as a composition of local maps induced by atomic operation. This mesh evolution "history", enables precise bidirectional data transfer regardless of connectivity changes.

Our approach builds on two key ideas. First, for 2D triangle meshes, we construct local atlases using a shared scaffold structure [Jiang et al. 2017] that ensures bijectivity through locally injective optimization [Rabinovich et al. 2017b]. Second, for 3D tetrahedral meshes, we leverage Steinitz's Theorem [Ribó Mor et al. 2011] to construct convex polyhedral embeddings that geometrically prevent tetrahedral overlaps. These bijective local atlases enable robust tracking of geometric entities—including points, curves, and surfaces—throughout complex remeshing sequences. We demonstrate our framework on applications including texture transfer across remeshed models Figures 4.9 and 4.10 and preserving organ segmentations in medical CT data under mesh adaptation (Figure 4.13) and validate our topology preservation on the Thingi10K dataset [Zhou and Jacobson 2016].

4.2 Related Work

Maintaining bijective correspondences is a long-standing challenge in geometry processing, spanning surface parameterization, volumetric mapping, and attribute transfer.

Strict Surface Homeomorphisms. Bijective surface parameterization traditionally relied on Tutte's embedding theorem [Tutte 1963], which guarantees an injective mapping for 3-connected planar graphs into convex domains using barycentric coordinates [Floater 2003]. While robust, these linear methods are restricted to fixed convex boundaries. To allow boundary movement and minimize isometric distortion, non-linear optimization frameworks such as SLIM [Rabinovich et al. 2017b] and Total Lifted Content (TLC) [Du et al. 2020] have been proposed. However, these methods either require an already-injective initialization or lack global bijectivity guarantees on complex topologies. Our 2D Shared Scaffold structure builds upon the Simplicial Complex Augmentation Framework (SCAF) [Jiang et al. 2017], which ensures global bijectivity by reducing the problem to local orientation preservation within an augmented tessellation of the ambient space [Lipman 2014].

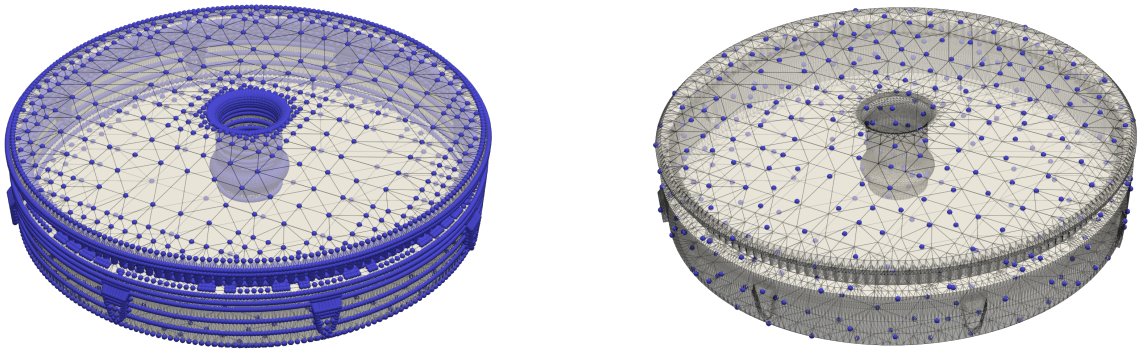
Volumetric Mapping and Topological Obstructions. Extending bijectivity guarantees to 3D tetrahedral meshes is significantly more complex due to topological obstructions. Notably, the 3D analog of Tutte's theorem does not hold universally; even with convex boundaries, because internal tetrahedra can invert if the mesh contains specific forbidden minors like K_6 or $K_{3,3,1}$ [Alexa 2023; Floater and Pham-Trong 2006]. Constructive methods such as Simplicial Foliations [Campen et al. 2016] and the Shrink-and-Expand (SaE) framework [Nigolian et al. 2023, 2024] provide theoretical guarantees for shellable meshes but often incur extreme computational costs and memory-intensive mesh refinement. In contrast, our approach utilizes Steinitz's Theorem [Steinitz 1922] and Maxwell-Cremona lifting [Ribó Mor et al. 2011] to construct local convex

polyhedral embeddings for boundary operations. This ensures validity by exploiting the geometric property that convexity of the boundary polyhedron prevents interior overlap, avoiding the need for expensive global constructive schemes.

Mapping Preservation and Attribute Transfer. The "Bijective Prism Shell" [Jiang et al. 2020, 2021] establishes a common domain for spatially close surfaces by constructing a volumetric shell between them, though the approach is limited to surfaces within a narrow distance threshold and requires careful shell thickness tuning. Another direction is the use of "common triangulations" [Schmidt et al. 2023], which decouple map resolution from input complexity by adaptively refining a shared triangulation. However, this requires maintaining and updating a global structure across the entire remeshing sequence, incurring substantial computational overhead. Alternative approximate methods such as reversible harmonic maps [Ezuz et al. 2019] and low-resolution correspondence [Maggioli et al. 2024] trade accuracy for efficiency, but inherently incur information loss during transfer due to their continuous relaxation or downsampling strategies.

The closest work to ours is the successive self-parameterization (SSP) framework [Liu et al. 2021], which constructs bijective mappings by composing local atlases across remeshing operations. The atlases are constructed by a joint-flattening strategy where the 3D coordinates of pre- and post-operation patches (the localized mesh regions affected by each operation) are simultaneously parameterized into a shared 2D domain. While this approach successfully distributes distortion between mesh states, it suffers from two key limitations: First, the optimization lacks robustness guarantees and can fail to converge or produce inverted elements when mesh quality is poor. We observe frequent failures on models from the Thingi10K dataset [Zhou and Jacobson 2016]—for instance, model #1706476 exhibits failure of SSP because it prevents any collapse operations from completing (Figure 4.2). Second, SSP is designed specifically for edge collapse operations in the context of mesh coarsening and does not generalize to other remeshing primi-

tives that we support, such as edge splits (for refinement), edge swaps (for quality improvement), or vertex smoothing (for geometry optimization). Additionally, the framework does not extend to 3D tetrahedral meshes, where boundary operations require fundamentally different geometric constructions.



(a) SSP method fails to complete any edge collapses

(b) Our method robustly generates coarse-to-fine mapping

Figure 4.2: Comparison on Thingi10K model #1706476. We visualize the coarse-to-fine mapping by transferring points from the decimated mesh back to the original fine mesh using successive parameterization. **Left:** The SSP framework [Liu et al. 2021] fails during optimization, preventing any edge collapse operations from completing. **Right:** Our shared scaffold method successfully constructs bijective mappings throughout the decimation process, enabling robust data transfer between mesh representations.

4.3 Method

Given an input manifold mesh $\mathcal{M}_{\text{input}}$ and a sequence of remeshing operations $O = \{o_1, o_2, \ldots, o_n\}$, our goal is to maintain a continuous, bijective mapping $f : \mathcal{M}_{\text{input}} \to \mathcal{M}_{\text{output}}$ throughout the entire remeshing process. We assume the input $\mathcal{M}$ is a manifold simplicial complex, which can be either a 2D triangle mesh embedded in $\mathbb{R}^3$ or a 3D tetrahedral mesh in $\mathbb{R}^3$. The operation sequence O may consist of various topological and geometric modifications, including edge collapses, edge

splits, edge swaps, and vertex smoothing.

We follow the insight of [Liu et al. 2021] by noting that, while the cumulative effect of remeshing is global, each individual operation $o_i : \mathcal{M}_{i-1} \to \mathcal{M}_i$ only acts on a small, localized patch (the Region of Interest, Figure 4.4). We denote these local patches as $\mathcal{P}_i^{\text{before}} \subset \mathcal{M}_{i-1}$ and $\mathcal{P}_i^{\text{after}} \subset \mathcal{M}_i$, corresponding to the mesh state before and after the operation.

To construct a bijection between these two patches, we embed both into a shared geometric domain C_i via embeddings $e_i^{\text{before}} : \mathcal{P}_i^{\text{before}} \to C_i$ and $e_i^{\text{after}} : \mathcal{P}_i^{\text{after}} \to C_i$. (Figure 4.7) This establishes a canonical bijection:

$$\varphi_i = (e_i^{\text{after}})^{-1} \circ e_i^{\text{before}} : \mathcal{P}_i^{\text{before}} \to \mathcal{P}_i^{\text{after}}. \tag{4.1}$$

Since the operation only modifies the local region, this local bijection extends to a global map $\varphi_i : \mathcal{M}_{i-1} \to \mathcal{M}_i$ that is the identity outside of $\mathcal{P}_i^{\text{before}}$. The global bijective mapping is then obtained by composition:

$$\Phi = \varphi_n \circ \varphi_{n-1} \circ \cdots \circ \varphi_1. \tag{4.2}$$

This formulation allows us to handle diverse operations—including splits for refinement and collapses for coarsening—within a unified framework.

4.3.1 Constructing Bijective Maps Using Local Atlases

We use different bijective maps for triangle meshes and tetrahedral mesh, but they follow the same fundamental principle of constructing a shared parametric space through which we can perform mapping. The main difficulty is that remeshing can change the domain of the mesh, so our bijective maps must be able to handle a certain amount of distortion. For triangle meshes we are minimizing the distortion of points in 3D and most operations introduce some distortion to

the geometry that must be minimized. On the other hand, tetrahedral meshes are embedded in 3D so for operations on the interior the identity map to $\mathbb{R}^3$ is the perfect bijective map and the entire challenge is how to deal with operations on the boundary that distort the domain of the tetrahedral mesh.

4.3.1.1 2D: Triangle Mesh

Background: Joint Flattening.

Our 2D approach builds upon the joint flattening strategy introduced by [Liu et al. 2021]. The key idea is to leverage the fact that when the patch $\mathcal{P}^{\text{before}}$ of a local operation lies on the interior of $\mathcal{M}_{i-1}$ then $\mathcal{P}^{\text{after}}$ lies on the interior of $\mathcal{M}_i$ and their boundaries remain geometrically consistent: $\partial\mathcal{P}^{\text{before}} \equiv \partial\mathcal{P}^{\text{after}}$. This allows both patches to be parameterized into a shared 2D domain by minimizing a joint distortion energy:

$$\min_{\mathbf{U}} \quad E(\mathcal{P}^{\text{before}}, \mathbf{U}^{\text{before}}) + E(\mathcal{P}^{\text{after}}, \mathbf{U}^{\text{after}}) \tag{4.3}$$

subject to shared boundary coordinates:

$$\mathbf{u}_v^{\text{before}} = \mathbf{u}_v^{\text{after}}, \quad \forall v \in \partial\mathcal{P}. \tag{4.4}$$

where $\mathbf{U}^{\text{before}}$ and $\mathbf{U}^{\text{after}}$ are the UV coordinates for all vertices in $\mathcal{P}^{\text{before}}$ and $\mathcal{P}^{\text{after}}$ respectively, $\mathbf{u}_v \in \mathbb{R}^2$ denotes the UV position of vertex v, and E is a standard distortion metric such as Symmetric Dirichlet [Smith and Schaefer 2015b] or ARAP energy [Sorkine and Alexa 2007].

For edge collapses on the boundary, where vertex i is collapsed toward vertex j, a colinearity constraint is applied: the collapsed position $\mathbf{u}_i$ must remain colinear with its neighboring boundary vertices j and k in the parametric domain. Although this treatment ensures the boundary

shape remains well-defined for both connectivity states, it does not guarantee bijectivity: the optimization can produce overlapping or inverted triangles, particularly on poor-quality meshes. To rigorously enforce bijectivity, we augment this framework with a *shared scaffold structure*. We first review the Simplicial Complex Augmentation Framework (SCAF) [Jiang et al. 2017] that provides the theoretical foundation, then describe our construction of the shared scaffold for joint flattening.

Background: Simplicial Complex Augmentation Framework. SCAF guarantees bijectivity by adding an auxiliary "scaffold" triangulation around a mesh patch to make it so that global bijectivity can preserved by maintaining local injectivity. Given a 2D mesh patch $\mathcal{P}$ to be parameterized, SCAF constructs a scaffold $\mathcal{S}$ between $\mathcal{P}$ and its bounding box so $\mathcal{D} = \mathcal{S} \cup \mathcal{P}$ is a simplicial complex that tessellates a convex domain (Figure 4.3).

The critical property of SCAF is that any potential self-intersections of $\mathcal{P}$ are captured by checking for *local injectivity* (i.e., has positive Jacobian determinant per element) on $\mathcal{D}$ so local and global bijectivity become identical. then the entire mapping is *globally bijective*. As such, global overlaps can be reduced to per-element orientation checks. We can therefore apply locally injective optimization methods like SLIM [Rabinovich et al. 2017b] with an orientation-preserving line search to optimize distortion while guaranteeing global bijectivity.

Shared Scaffold for Joint Flattening. Since $\mathcal{P}^{\text{before}}$ and $\mathcal{P}^{\text{after}}$ are both parameterized into the same 2D domain and share a common boundary $\partial\mathcal{P}$, we can naturally construct a *single* shared scaffold $\mathcal{S}$ for *both* patches. Specifically, we form two augmented complexes:

$$\mathcal{D}^{\text{before}} = \mathcal{P}^{\text{before}} \cup \mathcal{S}, \qquad \mathcal{D}^{\text{after}} = \mathcal{P}^{\text{after}} \cup \mathcal{S}. \tag{4.5}$$

We then minimize the joint distortion energy over the vertex positions of both augmented com-

plexes:

$$\min_{\mathbf{U}} \quad E(\mathcal{D}^{\text{before}}, \mathbf{U}^{\text{before}}) + E(\mathcal{D}^{\text{after}}, \mathbf{U}^{\text{after}}), \tag{4.6}$$

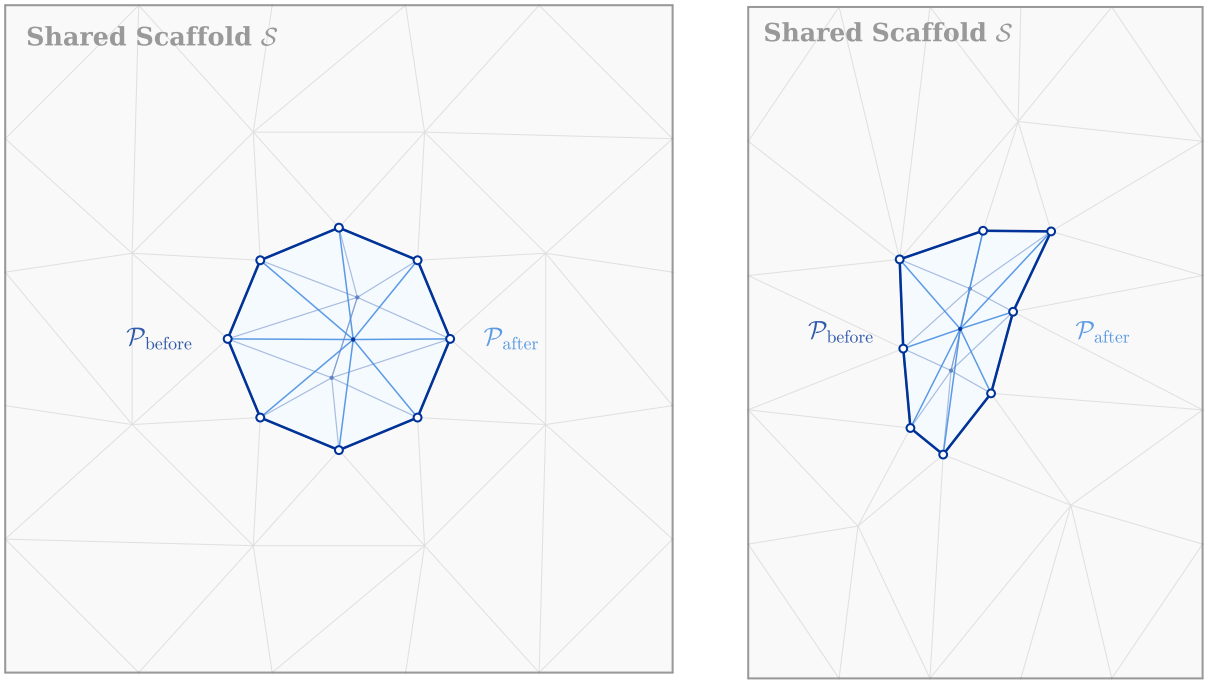
subject to:

- **Shared boundary**: $\mathbf{u}_v^{\text{before}} = \mathbf{u}_v^{\text{after}}$ for all $v \in \partial\mathcal{P}$.
- **Shared scaffold**: $\mathbf{u}_v^{\text{before}} = \mathbf{u}_v^{\text{after}}$ for all $v \in \mathcal{S}$.

By using SCAF and SLIM the result of our optimization is guaranteed to produce a globally bijective embedding of $\mathcal{P}^{\text{before}}$ and $\mathcal{P}^{\text{after}}$. Furthermore, since both patches share the same non-overlapping scaffold $\mathcal{S}$, a bijective correspondence between $\mathcal{P}^{\text{before}}$ and $\mathcal{P}^{\text{after}}$ can be established through the resulting common parametric domain. Figure 4.3 illustrates this shared scaffold framework.

Application to Different Operation Types. The shared scaffold framework applies uniformly across various remeshing operations. Figure 4.4 illustrates the the local patches for each operation type. Using the open star St ([Munkres 2018]), the local patches we use are:

1. *Edge Collapse* $(i, j) \to k$: $\mathcal{P}^{\text{before}} = \text{St}(i) \cup \text{St}(j)$ (union of 1-rings of both vertices), $\mathcal{P}^{\text{after}} = \text{St}(k)$ (1-ring of merged vertex).
2. *Edge Split* on edge (i, j) creating vertex k: $\mathcal{P}^{\text{before}} = \text{St}(i, j)$ (triangles incident to the edge), $\mathcal{P}^{\text{after}} = \text{St}(k)$ (1-ring of new vertex).
3. *Edge Swap*: $\mathcal{P}^{\text{before}}$ and $\mathcal{P}^{\text{after}}$ are both the two triangles sharing the edge.
4. *Vertex Smoothing* of vertex v: $\mathcal{P}^{\text{before}} = \mathcal{P}^{\text{after}} = \text{St}(v)$ (1-ring of the vertex, connectivity unchanged).



(a) Shared Scaffold Structure **(b)** After Joint Optimization

Figure 4.3: Shared scaffold framework for bijective atlas construction. (a) Both $\mathcal{P}^{\text{before}}$ and $\mathcal{P}^{\text{after}}$ are embedded within the same scaffold $\mathcal{S}$ (gray), sharing a common boundary $\partial\mathcal{P}$ (dark blue). (b) After joint optimization, the scaffold $\mathcal{S}$ is deformed to minimize distortion while maintaining the sharing constraint ($\mathbf{u}_v^{\text{before}} = \mathbf{u}_v^{\text{after}}$ for all $v \in \mathcal{S} \cup \partial\mathcal{P}$). Local injectivity via SLIM optimization guarantees global bijectivity for both augmented complexes through the SCAF property, establishing bijective correspondence via the shared parametric domain.

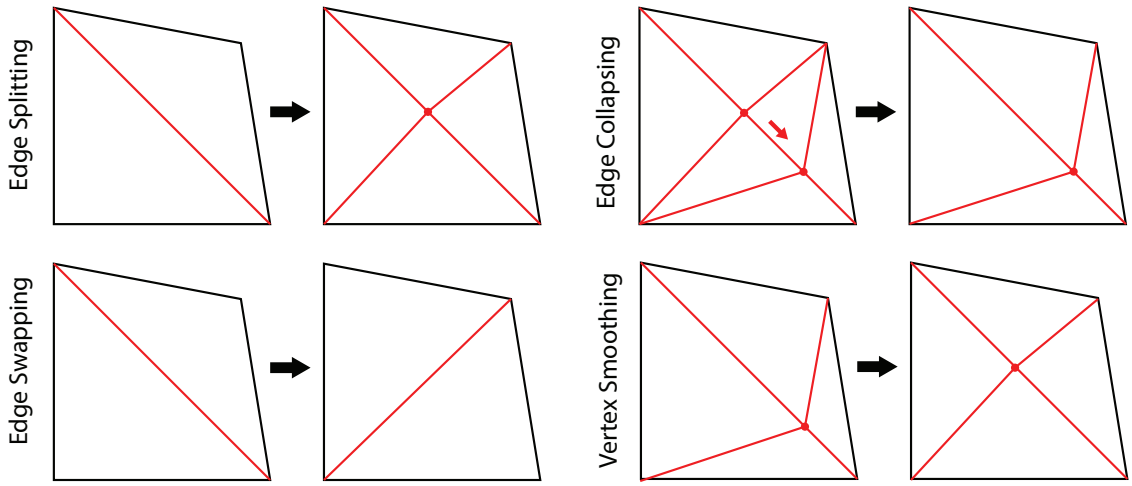


Figure 4.4: Patches for different remeshing operations.

4.3.1.2 3D: Tetrahedral Mesh

Recall that for tetrahedral meshes interior operations constructing bijective mapping is trivial and no deformation is necessary at all. In such cases, we extract the affected region (the closed star of the operation) as the local atlas and use identity mapping between the old and new connectivity to maintain bijectivity.

The primary challenge arises when handling **boundary operations**, which introduce geometric changes to the mesh surface. Of the four operations we choose use in our system, three can be implemented concisely:

Boundary Edge Split. Splitting a boundary edge does not alter the geometric shape of the boundary surface and only refine the mesh connectivity. Since the boundary geometry remains unchanged we can extract the affected local region and establish a canonical bijective correspondence, identical to the treatment of interior operations.

Boundary Vertex Smoothing. Vertex smoothing preserves the mesh connectivity while modifying vertex positions. In this case the set of tetrahedra in $\mathcal{P}^{\text{before}}$ and $\mathcal{P}^{\text{after}}$ have identical combinatorial structure and we presume that when a vertex of a tetrahedron is moved the points in the tetrahedron are all moved linearly. As such, we move each point in each tetrahedron so that its coordinate with respect to that tetrahedron's barycentric coordinates remains the same before and after the operation.

Boundary Edge Swap. An edge swap decomposes into edge split followed by edge collapse. As the split introduces no distortion and we have to implement the collapse anyway we chose to implement the swap atlas as the composition of the split atlas and the collapse atlas.

4.3.1.3 Boundary Edge Collapse

The remaining atlas to describe is the **boundary edge collapse**, which genuinely alters the boundary geometry and connectivity simultaneously. The object is, therefore, to construct a reparametrization that embeds both $\mathcal{P}^{\text{before}}$ and $\mathcal{P}^{\text{after}}$ into a consistent geometric domain, thereby establishing a bijective correspondence between them.

Consider a mesh $\mathcal{M}^\ell$ at step ℓ of the remeshing sequence, with boundary surface $\partial\mathcal{M}^\ell$. Suppose we perform a boundary edge collapse on edge (i, j), yielding the updated mesh $\mathcal{M}^{\ell+1}$. Without loss of generality, we assume vertex i is collapsed toward vertex j.

Local Patch Extraction. Consider a boundary edge collapse operation that collapses vertex i toward vertex j. We define the local patches as follows:

$$\mathcal{P}^{\text{before}} = \text{St}(i)^{\text{before}}, \tag{4.7}$$

$$\mathcal{P}^{\text{after}} = \text{St}(j)^{\text{after}} \cap \text{St}(i)^{\text{before}}, \tag{4.8}$$

where $\text{St}(v)$ denotes the star of vertex v (i.e., the set of all tetrahedra incident to v). The superscripts indicate the mesh state before or after the collapse operation. Here $\mathcal{P}^{\text{after}}$ consists of the tetrahedra in $\mathcal{P}^{\text{before}}$ that "survive" the collapse operation.

Coplanarity Constraint. Similar to the 2D case for handling boundary edges, where we impose colinearity constraints to ensure a consistent boundary curve, in 3D we adopt an analogous strategy: during reparametrization, we constrain the boundary triangles affected by the collapse to lie in a common plane. Specifically, we require that the portions of the local patch boundary lying on the global mesh boundary,

$$\partial\mathcal{P}^{\text{before}} \cap \partial\mathcal{M}^l \quad \text{and} \quad \partial\mathcal{P}^{\text{after}} \cap \partial\mathcal{M}^{l+1}, \tag{4.9}$$

are reparametrized to lie in the same plane. Therefore, the collapse operation will not change the shape of the parametrization domain. Our problem then reduces to finding a reparametrization that satisfies this coplanarity constraint and can accommodate both the pre-collapse connectivity $\mathcal{P}^{\text{before}}$ and the post-collapse connectivity $\mathcal{P}^{\text{after}}$.

Reduction to Convex Polyhedron Construction. As established above, our problem reduces to finding a common reparametrization domain that can accommodate both $\mathcal{P}^{\text{before}}$ and $\mathcal{P}^{\text{after}}$. By Lemma B.1, this problem further simplifies to constructing a **convex polyhedron** that can simultaneously embed the boundaries $\partial\mathcal{P}^{\text{before}}$ and $\partial\mathcal{P}^{\text{after}}$. Crucially, to satisfy the coplanarity constraint, we require that the portions $\partial\mathcal{P}^{\text{before}} \cap \partial\mathcal{M}^l$ and $\partial\mathcal{P}^{\text{after}} \cap \partial\mathcal{M}^{l+1}$ are embedded onto a single face of this polyhedron. Specifically, the one-ring boundary vertices of vertex i, together with vertex i itself, must all lie on the same polyhedral face on the constructed convex polyhedron.

Combinatorial Formulation. We formulate this as a combinatorial problem. The boundary surface $\partial\mathcal{P}^{\text{before}}$ (equivalently, $\partial\mathcal{P}^{\text{after}}$) consists of two parts: (1) the $k-1$ boundary triangles that do not contain vertex i, denoted $f_0, \ldots, f_{k-1}$, and (2) the one-ring of vertex i on $\partial\mathcal{M}^\ell$, which forms an additional combinatorial face f_k (e.g., the face $[0, 6, 5, 4, 7, 1]$ in Figure 4.5(a)). Our question becomes: *Can the combinatorial graph G formed by $\{f_1, \ldots, f_k\}$ be realized as the 1-skeleton of a convex polyhedron?*

The following classical result provides an affirmative answer:

Theorem 4.1 (Steinitz's Theorem)**.** *A graph G is the 1-skeleton of a convex polyhedron if and only if G is simple, planar, and 3-connected.*

Constructive Algorithm. We follow the approach of [Ribó Mor et al. 2011] to realize a convex polyhedra embedding of the graph $\mathcal{G}$. We first select a triangle from $\{f_0, \ldots, f_{k-1}\}$ as the outer boundary and compute a planar Tutte embedding [Tutte 1963] (algorithm 2 and Figure 4.5(a)).

The Maxwell-Cremona lifting (algorithm 3 and Figure 4.5(b)) then elevates this 2D embedding into a 3D convex polyhedron by assigning heights to each vertex.

Algorithm 2: Tutte Barycentric Embedding

Input: Planar graph given by face list $F = \{f_1, f_2, \ldots\}$ and outer triangle $f_0 = (v_1, v_2, v_3)$
Output: 2D embedding $p : V \to \mathbb{R}^2$

Construct weighted Laplacian;
foreach *undirected edge* (i, j) **do**

$$\omega_{ij} \leftarrow \begin{cases} 1, & \text{if } (i,j) \text{ is an interior edge,} \\ 0, & \text{if } i, j \in f_0 \end{cases};$$

Assemble the $n \times n$ weighted Laplacian: $L_{ii} = \sum_j \omega_{ij}$, $L_{ij} = -\omega_{ij}$ for $i \neq j$;

Partition variables;
Let B be the three boundary vertex indices in f_0, and I be the interior vertices;
Partition: $L = \begin{pmatrix} L_{BB} & L_{BI} \\ L_{IB} & L_{II} \end{pmatrix}$;

Fix boundary vertices;
$p_{v_1} \leftarrow (0,0)$, $p_{v_2} \leftarrow (1,0)$, $p_{v_3} \leftarrow (0,1)$;
$x_B \leftarrow (0,1,0)^\top$, $y_B \leftarrow (0,0,1)^\top$;

Solve for interior vertices;
Solve $L_{II} x_I = -L_{IB} x_B$ and $L_{II} y_I = -L_{IB} y_B$;
return Crossing-free planar embedding $p : V \to \mathbb{R}^2$;

Initial Embedding and Joint Optimization. Having constructed the convex polyhedron embedding of $\mathcal{G}$, we can now recover valid embeddings for both $\mathcal{P}^{\text{before}}$ and $\mathcal{P}^{\text{after}}$ by adding back their interior connectivity. By Lemma B.1, the convexity of the boundary polyhedron guarantees that these embeddings are non-overlapping and non-inverted. For $\mathcal{P}^{\text{before}}$, we additionally need to place vertex i inside face f_k. A simple choice is the barycenter of f_k:

$$\mathbf{p}_i = \frac{1}{|f_k|} \sum_{v \in f_k} \mathbf{p}_v,$$

where $|f_k|$ denotes the number of vertices in face f_k (see Fig. 4.6).

To obtain a low-distortion embedding, we perform joint optimization over both $\mathcal{P}^{\text{before}}$ and $\mathcal{P}^{\text{after}}$

Algorithm 3: Maxwell-Cremona Lifting to Convex Polyhedron

Input: Planar embedding $p : V \to \mathbb{R}^2$ from Algorithm 2
Output: Convex polyhedra embedding $\{(p_v, z_v) \mid v \in V\} \subset \mathbb{R}^3$

Assign equilibrium stress;
foreach *interior edge* (i, j) **do**
  $\omega_{ij} \leftarrow 1$;

Define plane per face;
For each face f_k, seek plane $H_k : z = \langle (x, y), a_k \rangle + d_k$;

Maxwell-Cremona propagation;
Choose one interior face f_1 as base: $a_1 \leftarrow (0, 0)$, $d_1 \leftarrow 0$;
foreach *pair of adjacent faces* f_r, f_ℓ *sharing edge* (i, j) **do**
  where f_ℓ lies to the left of directed edge $i \to j$;
  $a_\ell \leftarrow \omega_{ij}(p_i - p_j)^\perp + a_r$; // $(x, y)^\perp = (-y, x)$
  $d_\ell \leftarrow \omega_{ij}\langle p_i, (p_j)^\perp \rangle + d_r$;

Compute vertex heights;
foreach *vertex* $v \in V$ **do**
  Pick any incident face f_k;
  $z_v \leftarrow \langle p_v, a_k \rangle + d_k$;
return Convex polyhedron $\{(p_v, z_v)\}$ whose projection onto the xy-plane is the Tutte embedding;

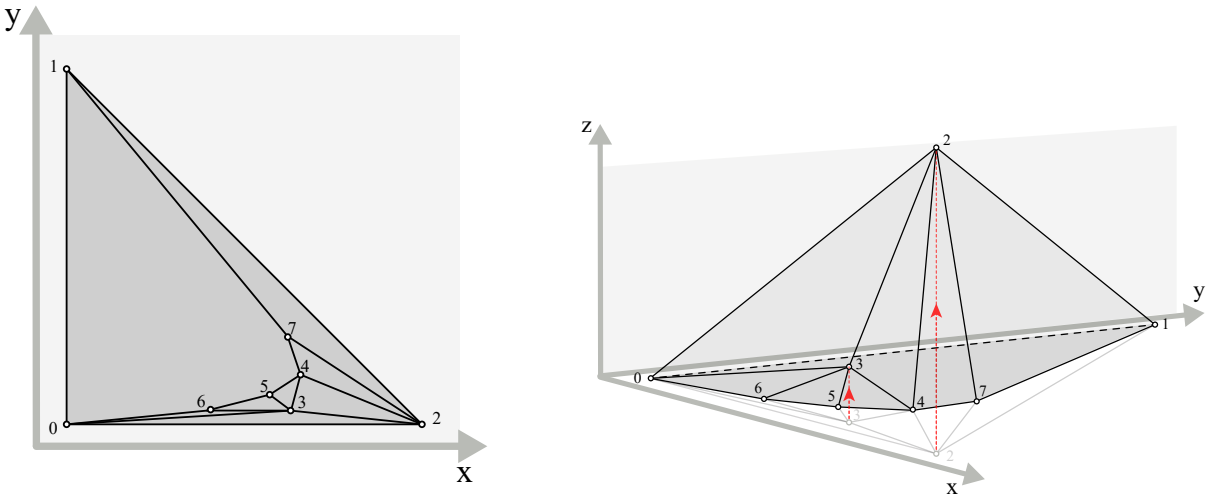
simultaneously. We minimize the AMIPS [Fu et al. 2015b] distortion energy:

$$\min_{\{\mathbf{p}_v\}} \quad E_{\text{AMIPS}}(\mathcal{P}^{\text{before}}) + E_{\text{AMIPS}}(\mathcal{P}^{\text{after}}),$$

subject to the **coplanarity constraint**: vertex i remains on the plane containing face f_k. Specifically, let $\mathbf{n}_k$ be the normal vector of face f_k, and let $v_0 \in f_k$ be any vertex on f_k. The constraint is:

$$(\mathbf{p}_i - \mathbf{p}_{v_0}) \cdot \mathbf{n}_k = 0.$$

The joint optimization framework, combined with the coplanarity constraint, yields a bijective mapping between $\mathcal{P}^{\text{before}}$ and $\mathcal{P}^{\text{after}}$ with minimal distortion, thus completing the construction of the local atlas for the boundary edge collapse operation. Note that the shared scaffold structure



(a) Tutte barycentric embedding **(b)** Maxwell-Cremona lifting

Figure 4.5: Constructive algorithm for convex polyhedra embedding. (a) The Tutte embedding (Algorithm 2) produces a crossing-free 2D planar embedding with the outer triangle $f_0 = [0, 1, 2]$ fixed at positions $(0, 0)$, $(1, 0)$, and $(0, 1)$. Interior vertices (labeled 3–7) are positioned at their barycentric coordinates, yielding a valid planar layout. (b) The Maxwell-Cremona lifting (Algorithm 3) elevates this 2D embedding into a 3D convex polyhedron by assigning heights to each vertex.

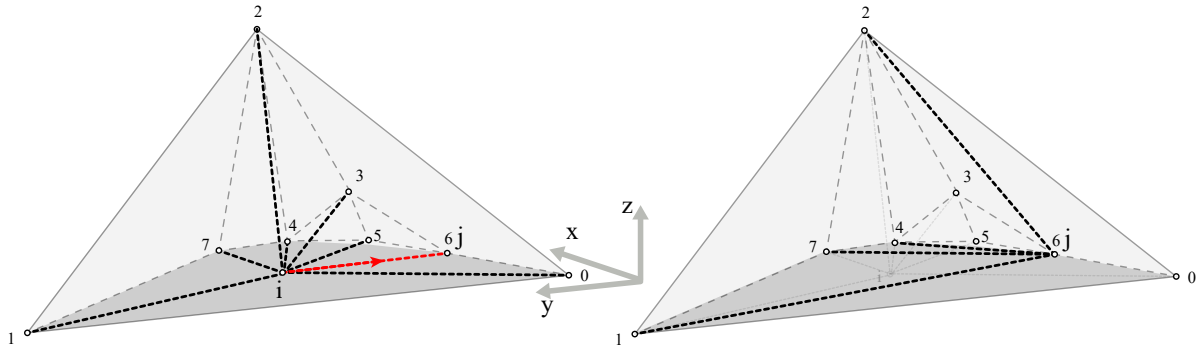


Figure 4.6: Initial valid embedding for both $\mathcal{P}^{\text{before}}$ and $\mathcal{P}^{\text{after}}$. After constructing the convex polyhedron C via Algorithms 2 and 3, vertex i is placed at the barycenter of face f_k (the one-ring of i on the boundary). By Lemma B.1, this configuration is valid (non-overlapping) for both the pre-collapse connectivity (with edge (i, j)) and the post-collapse connectivity (where i has collapsed to j). This provides a valid starting point for joint optimization.

can naturally extend to this 3D setting as well.

4.3.2 Tracking Using Local Atlases

Having constructed local bijective atlases for each type of remeshing operation, we can now use them to track geometric entities, including points, curves, and surfaces, throughout the remeshing sequence. A key advantage of our bijective approach is that the local maps enable *bidirectional tracking*: we can map entities forward from the original mesh $\mathcal{M}$ to the remeshed result $\mathcal{M}'$, or backward from $\mathcal{M}'$ to $\mathcal{M}$. For clarity of exposition, we only consider the backward tracking perspective throughout this section, though the forward direction follows by symmetry.

4.3.2.1 Point Tracking

Point Representation. A point p on a simplicial mesh is represented by its barycentric coordinates with respect to a simplex:

$$p = (t, \mathbf{b}), \tag{4.10}$$

where t is the ID of a triangle in a triangle mesh or a tetrahedron in a tetrahedral mesh $\mathbf{b} = (b_0, b_1, \ldots, b_d)$ are the barycentric coordinates with respect to the vertices of t ($d = 2$ for triangles, $d = 3$ for tetrahedra).

Backward Tracking Algorithm. Consider a remeshing operation $o_i : \mathcal{M}^{i-1} \to \mathcal{M}^i$ with its associated local atlas $(U_i, \mathcal{P}_i^{\text{before}}, \mathcal{P}_i^{\text{after}})$, where U_i is the shared parametric domain, $\mathcal{P}_i^{\text{before}} \subset \mathcal{M}^{i-1}$ and $\mathcal{P}_i^{\text{after}} \subset \mathcal{M}^i$ are the local patches affected by the operation, and U_i is the shared parametric domain into which both patches are embedded.

Now let us consider where to map a point $p' = (t', \mathbf{b}')$ on $\mathcal{M}^i$ to $\mathcal{M}^{i-1}$ given operation o_i.. If it lies outside of $\mathcal{P}_i^{\text{after}}$ then it will not move, so its backtracked position is the trivially the same, i.e $p = p'$. On the other hand, we first then map p' to the parametric domain using barycentric

interpolation

$$\mathbf{u}' = \sum_{j=0}^{d} b'_j \mathbf{u}_{v_j} \tag{4.11}$$

, where $\mathbf{u}_{v_j} \in U_i$ are the parametric coordinates of the vertices of t'.

We then locate the simplex $t \in \mathcal{P}_i^{\text{before}}$ containing $\mathbf{u}'$ and compute its barycentric coordinates $\mathbf{b}$ in t to obtain the result $p = (t, \mathbf{b})$.

The correctness follows from the fact that both $\mathcal{P}_i^{\text{before}}$ and $\mathcal{P}_i^{\text{after}}$ share the same geometric boundary and are embedded into the same parametric domain U_i, ensuring that any point $\mathbf{u}' \in U_i$ has a well-defined interpretation in both mesh states.

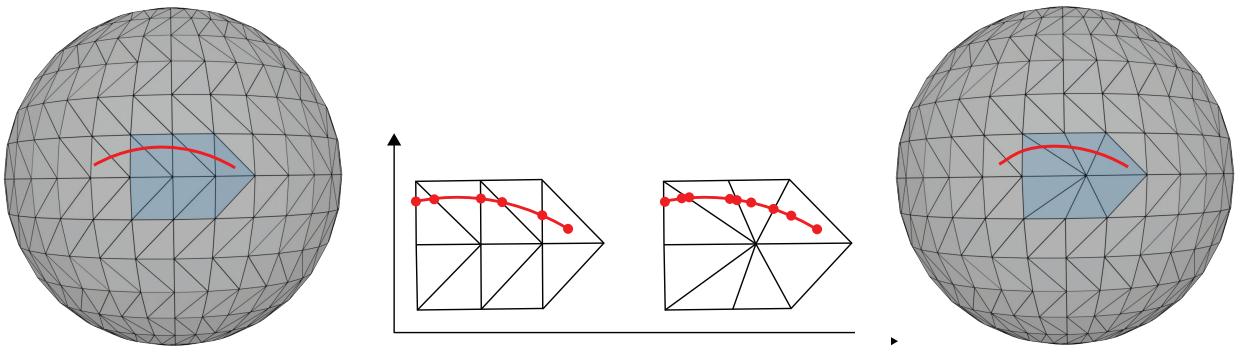


Figure 4.7: Curve tracking via local atlas. The portion of the curve (red) on the patch is first mapped to the local atlas (middle), where it is subdivided through arrangement with the new connectivity to generate intersection endpoints. This yields the tracked curve on the output mesh (right) with preserved topology.

4.3.2.2 Curve Tracking

Curve Representation. The task of tracking piecewise-linear curves on $\mathcal{M}'$ is not as trivial as simply projecting the endpoints of those curves to $\mathcal{M}$ because a single linear segment might not lie on $\mathcal{M}$. In general, when the linear segments lie on more than one curve their embedding in 3D ceases to unambiguously lie on a mesh. As such, the inputs and outputs of our curve tracking procedure are piecewise linear curves are comprised of linear segments that each lie on a single simplex (triangle or tetrahedron), though multiple segments may exist in a single simplex. A

curve C on a simplicial mesh is therefore represented as a sequence of segments:

$$C = \{\text{seg}_1, \text{seg}_2, \ldots, \text{seg}_k\}, \tag{4.12}$$

where each segment seg_i is defined within a single simplex:

$$\text{seg}_i = (t_i, \mathbf{b}_i^{(1)}, \mathbf{b}_i^{(2)}), \tag{4.13}$$

with t_i being the simplex and $\mathbf{b}_i^{(1)}, \mathbf{b}_i^{(2)}$ the barycentric coordinates of the two endpoints. We require that consecutive segments are connected: the second endpoint of seg_i coincides with the first endpoint of seg_{i+1}, i.e., $(t_i, \mathbf{b}_i^{(2)}) \equiv (t_{i+1}, \mathbf{b}_{i+1}^{(1)})$ as points on the mesh. We use $\equiv$ because when segments end on simplex boundaries there are multiple triangles (respectively, tetrahedra) for which to construct barycentric coordinates from and we only care that the segment endpoints hold equivalent barycentric coordinates.

Backward Tracking Problem. Given a curve $C' = \{\text{seg}'_1, \ldots, \text{seg}'_m\}$ on $\mathcal{M}^i$ and a local atlas $(U_i, \mathcal{P}_i^{\text{before}}, \mathcal{P}_i^{\text{after}})$, we seek to compute the pre-image curve $C = \{\text{seg}_1, \ldots, \text{seg}_n\}$ on $\mathcal{M}^{i-1}$. Since the connectivity changes within the local patch, a segment in C' that lies entirely within a single simplex in $\mathcal{P}_i^{\text{after}}$ may intersect multiple simplices in $\mathcal{P}_i^{\text{before}}$ due to the different mesh topology. As such, a single segment in C' may result in multiple segments in C to guarantee that each segment of C lies in a single simplex in $\mathcal{M}$.

Segment Arrangement Algorithm. We perform our segment arrangement in the joint parametric space, so for a given segment $(t', \mathbf{b}'^{(1)}, \mathbf{b}'^{(2)})$ with $t' \in \mathcal{P}_i^{\text{after}}$, we map its endpoints to the parametric domain:

$$\mathbf{u}^{(k)} = \sum_j b_j'^{(k)} \mathbf{u}_{v_j}, \; k \in \{1, 2\}. \tag{4.14}$$

Our goal is to compute the arrangement of the segment $[\mathbf{u}^{(1)}, \mathbf{u}^{(2)}]$ with respect to $\mathcal{P}_i^{\text{before}}$ in the joint parametric space U_i. To do this employ a ray-marching: starting from $\mathbf{u}^{(1)}$, we trace along direction $\mathbf{k} = \mathbf{u}^{(2)} - \mathbf{u}^{(1)}$ until reaching $\mathbf{u}^{(2)}$, computing intersections with simplex boundaries (edges for 2D, faces for 3D) in $\mathcal{P}_i^{\text{before}}$ at each step. These intersection points will all lie on simplex boundaries, yielding sub-segments $\{\widetilde{\text{seg}}_1, \ldots, \widetilde{\text{seg}}_\ell\}$ where each lies entirely within a single simplex of $\mathcal{P}_i^{\text{before}}$ (Figure 4.7).

Intersection Candidate Selection. At each step, given the current position $\mathbf{V}$ in the parametric domain and direction $\mathbf{k}$, we determine candidate facets (edges for 2D, faces for 3D) based on the location of $\mathbf{V}$ (Figure 4.8).

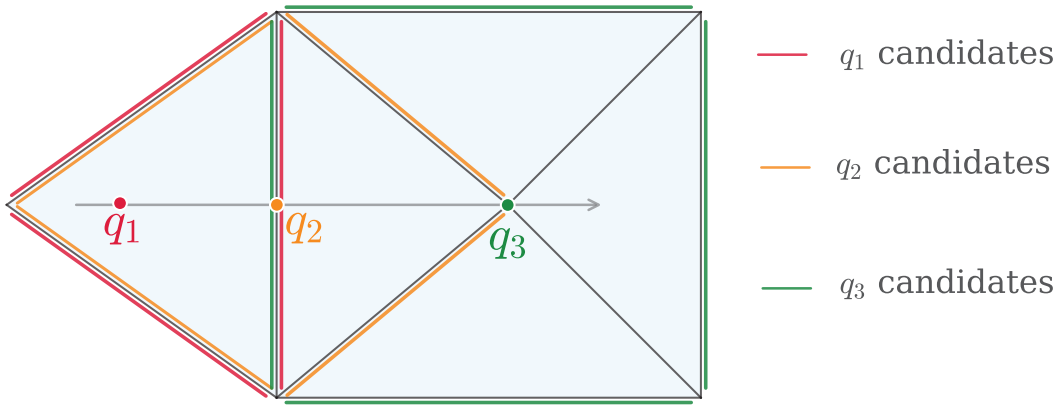


Figure 4.8: Candidate facet selection during curve tracking: Given three query points q_1, q_2, q_3 along a ray, we select candidate edges (highlighted in color) based on each point's location. Point q_1 lies in the interior of a triangle, so all three edges are candidates. Point q_2 lies on an edge, so candidates come from the link of that edge (opposite edges from incident triangles). Point q_3 lies at a vertex, so candidates are edges opposite to that vertex in all incident triangles. The algorithm selects the intersection with the smallest positive parameter t to advance to the next point.

If $\mathbf{V}$ lies strictly inside a simplex t, the candidates are all $(d-1)$-dimensional facets of t: three edges for triangles, four faces for tetrahedra.

If $\mathbf{V}$ lies on a lower-dimensional sub-simplex σ (a vertex, an edge for 2D/3D, or a face for 3D), we collect candidates from the link [Munkres 2018] of σ, i.e all $(d-1)$-dimensional facets from

simplices incident to σ excluding those facets that contain σ itself.

- **In 2D:** if σ is an edge (i, j), then $\text{link}((i, j))$ includes the opposite edges from all triangles sharing (i, j). If σ is a vertex v, then $\text{link}(v)$ includes all edges opposite to v in triangles containing v.

- **In 3D:** if σ is a face, then $\text{link}(\sigma)$ includes the opposite faces from tetrahedra sharing that face. If σ is an edge, then $\text{link}(\sigma)$ includes all faces from incident tetrahedra that do not contain the edge. If σ is a vertex v, then $\text{link}(v)$ includes all faces opposite to v in tetrahedra containing v.

Ray-Facet Intersection. For each candidate facet f, we compute the ray-facet intersection (ray-edge for 2D, ray-triangle for 3D) and select the one with smallest positive parameter $t > 0$ as the next intersection point. The detailed formulation is provided in Appendix B.2. This process repeats until reaching $\mathbf{u}^{(2)}$, producing the sub-segments $\{\widetilde{\text{seg}}_1, \ldots, \widetilde{\text{seg}}_\ell\}$ in $\mathcal{P}_i^{\text{before}}$.

4.3.2.3 Topology-Preserving Multi-Curve Tracking

In applications such as tracking UV seams or material boundaries on triangle meshes, we will need to track multiple curves or loops simultaneously. In such scenarios, the primary requirement is preserving the topological relationships among curves. Specifically, this means maintaining their intersection and overlap patterns in the correct order. Although our maps are bijective, running our procedure using floating point arithmetic occasionally changes the topological relationships between curves. On the other hand, exact rational arithmetic becomes computationally prohibitive. To balance these two extremes we take the approach of defining invariants that can help us determine whether a floating point computation introduced topological changes, and if a topological change does occur we re-run our tracking algorithm using exact rational predicates.

Intersection and Overlap Events. Consider a set of curves $C = \{C_1, \ldots, C_\ell\}$ to be tracked. As we traverse a curve C_i from start to end, it encounters a sequence of events where it interacts with other curves (including itself for self-intersections). We classify these events into two types:

- **Intersection**: Two segments cross at a single point (either at shared endpoints or at a transversal intersection within a common simplex).
- **Overlap**: Two segments coincide along a positive-length interval (not just a single point).

For each curve C_i, we record the ordered sequence of events:

$$\mathcal{E}(C_i) = \langle e_1, e_2, \ldots, e_m \rangle, \tag{4.15}$$

where each event e_k specifies:

- The type (intersection or overlap),
- The interacting curve index j (where j may equal i for self-interactions),
- The parametric location(s) along C_i where the event occurs.

Topology Preservation Requirement. Our goal is to ensure that for each curve $C_i \in C$ and its tracked result C_i', the event sequence is preserved:

$$\mathcal{E}(C_i) \cong \mathcal{E}(C_i'), \tag{4.16}$$

where $\cong$ denotes combinatorial equivalence: the sequences have the same length, and corresponding events have the same type and involve the same curve pairs.

Since intersections and overlaps are defined intrinsically within each simplex, we verify topol-

ogy preservation locally for each operation. After tracking all curves through the local atlas $(U_i, \mathcal{P}_i^{\text{before}}, \mathcal{P}_i^{\text{after}})$, we compute the event sequences in $\mathcal{P}_i^{\text{before}}$ and verify that they match the original sequences in $\mathcal{P}_i^{\text{after}}$.

Implementation Strategy. We track all curves within a local patch simultaneously and extract their event sequences by:

1. For each pair of curves (C_i, C_j) (including $i = j$), enumerate all segment pairs and classify their interactions.

2. Sort events along each curve according to their parametric positions.

3. Compare the resulting event sequences before and after tracking.

If the event sequences do not match ,typically due to numerical errors in double-precision arithmetic, we re-run tracking using exact rational arithmetic to ensure correctness.

Curve Simplification via Edge Collapse. To improve efficiency when tracking many curves over long remeshing sequences, we simplify curves by collapsing short segments. Given a segment $\text{seg} = (t, \mathbf{b}^{(1)}, \mathbf{b}^{(2)})$ with parametric length below a threshold ϵ, we consider collapsing it by merging its endpoints.

Crucially, edge collapse must preserve the event sequences $\mathcal{E}(C_i)$ for all curves. Before collapsing a segment in curve C_i, we verify that the local event sequence near the segment (including events involving C_i and other curves) remains unchanged after collapse and that no new intersections or overlaps are created, and no existing ones are destroyed.

This verification can be performed by computing event sequences in a local neighborhood before and after each potential collapse. By iteratively simplifying curves while preserving their

event sequences, we significantly reduce computational cost without compromising topological correctness.

4.3.2.4 Surface Tracking on Tetrahedral Meshes

For tetrahedral meshes, we extend the tracking framework to handle surfaces, which are essential for applications such as tracking material interfaces or segmentation boundaries in volumetric simulations.

Surface Representation. A surface S on a tetrahedral mesh is represented as a triangle mesh embedded in the volume, where each vertex is a point on the tetrahedral mesh:

$$p = (t, \mathbf{b}), \tag{4.17}$$

with t being a simplex(tetrahedron) ID and $\mathbf{b} = (b_0, b_1, b_2, b_3)$ the barycentric coordinates. The surface consists of triangular faces:

$$S = \{f_1, f_2, \ldots, f_k\}, \tag{4.18}$$

where each face $f_i = (p_i^{(1)}, p_i^{(2)}, p_i^{(3)})$ is defined by three vertices. We require that all three vertices of each face lie within the same tetrahedron, including its boundary. That is, if $p_i^{(j)} = (t_i^{(j)}, \mathbf{b}_i^{(j)})$ for $j \in \{1, 2, 3\}$, then either $t_i^{(1)} = t_i^{(2)} = t_i^{(3)}$, or they share a common face, edge, or vertex of the tetrahedral mesh.

Surface Arrangement Algorithm. Given a surface S' on $\mathcal{M}^i$ and a local atlas $(U_i, \mathcal{P}_i^{\text{before}}, \mathcal{P}_i^{\text{after}})$, we track the portion of S' that intersects the local patch $\mathcal{P}_i^{\text{after}}$.

For each face $f \in S'$ with $f \subset \mathcal{P}_i^{\text{after}}$, we:

1. Map the three vertices of f to the parametric domain U_i, obtaining a triangle $\tilde{f}$ in the parametric domain.

2. Extract the boundary surface of $\mathcal{P}_i^{\text{before}}$ restricted to the region covered by $\tilde{f}$.

3. Compute the arrangement of $\tilde{f}$ with the boundary triangles of $\mathcal{P}_i^{\text{before}}$ using the `autorefine_triangle_soup` function from CGAL [Coeurjolly et al. 2025], which subdivides overlapping triangles into non-overlapping pieces and retriangulates the result.

4. Convert the resulting triangles back to barycentric coordinates with respect to tetrahedra in $\mathcal{P}_i^{\text{before}}$.

The CGAL arrangement procdure [The CGAL Project 2025] guarantees that the output is a valid triangulation where each triangle lies entirely within a single tetrahedron or on its boundary. To ensure robustness, this computation can be performed using exact rational arithmetic.

Topology for Multiple Surfaces. When tracking multiple surfaces $\mathcal{S} = \{S_1, \ldots, S_m\}$, we preserve their topological relationships by tracking intersections between surface pairs.

Just like with curve tracking, we will guarantee topology by developing an invariant. For surfaces, the relevant topological invariant is the number and connectivity of intersection curves. If surfaces S_i and S_j intersect, their intersection forms a set of curves (possibly multiple disjoint components). We define the *intersection graph* $\mathcal{G}(\mathcal{S})$ where:

- Nodes represent connected components of pairwise surface intersections.

- Edges connect components that share endpoints or form junctions.

Our goal is to ensure that, for the tracked surfaces $\mathcal{S}'$, the intersection graph remains combinatorially equivalent: $\mathcal{G}(\mathcal{S}) \cong \mathcal{G}(\mathcal{S}')$.

Surface Simplification via Edge Collapse. Similar to curve tracking, we simplify surface representations by collapsing short edges to improve efficiency. Given an edge $e = (p^{(1)}, p^{(2)})$ in a surface, we collapse it if its length is below a threshold ϵ, if the collapse keeps all incident triangles remain within a single tetrahedron or itsboundary, and if the intersection graph $\mathcal{G}(\mathcal{S})$, is preserved.

To verify topology preservation, we compute the number of connected components in the intersection curves before and after collapse. If the component count changes, we reject the collapse. By iteratively applying valid collapses, we maintain a simplified surface representation throughout the tracking process without compromising topological correctness.

4.4 Results

4.4.1 Texture Transfer

A practical application of our point tracking framework is texture transfer across remeshed models (Figures 4.9 and 4.10). Given an input mesh with an existing UV parameterization and texture, our goal is to generate equivalent textures for remeshed versions of the model that may use entirely different parameterizations.

The key insight is that our bijective mapping decouples remeshing from UV maintenance. Traditional approaches must carefully preserve UV coordinates throughout remeshing operations, which becomes increasingly difficult when the mesh contains complex UV seams or multiple charts[Sander et al. 2001]. In contrast, our method allows the remeshed model to be reparameterized independently of any choice in standard parameterization algorithm. The texture is then transferred by composing the new parameterization with our tracked bijective correspondence: we evalute each texel in the new texture domain by first mapping to the remeshed surface, track-

ing it backward to the original mesh, and finally sample the original texture at the corresponding UV coordinates.

Figures 4.9 and 4.10 demonstrate texture transfer through both mesh decimation and refinement operations. The center images show the original models with their input UV parameterizations and textures. The top row displays the remeshed results after decimation (left) and refinement (right), both rendered with successfully transferred textures. The bottom row shows the UV layouts and corresponding texture maps generated for each remeshed model. Despite significant changes in mesh resolution and connectivity our bijective framework ensures accurate texture correspondence without introducing artifacts such as seam tearing or texture deviation.

4.4.2 Curve Tracking on Triangle Meshes.

We demonstrate our curve tracking algorithm on triangle meshes undergoing remeshing operations (Figure 4.11). Given an input mesh $\mathcal{M}_{\text{input}}$, we generate a collection of curves by intersecting the surface with axis-aligned planes parallel to the xy, yz, and xz coordinate planes. These curves form a complex network with several intersection points where curves from different plane families cross each other.

We then perform forward tracking through the remeshing sequence to obtain the corresponding curves on $\mathcal{M}_{\text{output}}$. As shown in Figure 4.11, the tracked curves faithfully preserve their intersection topology: the combinatorial pattern of curve crossings remains identical between input and output, with no spurious intersections introduced and no existing intersections lost. This topological guarantee is achieved through our topology-preserving multi-curve tracking algorithm (Section 4.3.2.3), which maintains ordered sequences of intersection events throughout the tracking process. Critically, during the segment arrangement phase, we employ exact arithmetic with coordinates represented as rational numbers to evaluate geometric predicates, ensuring numerical robustness and preventing topological inconsistencies that could arise from floating-point

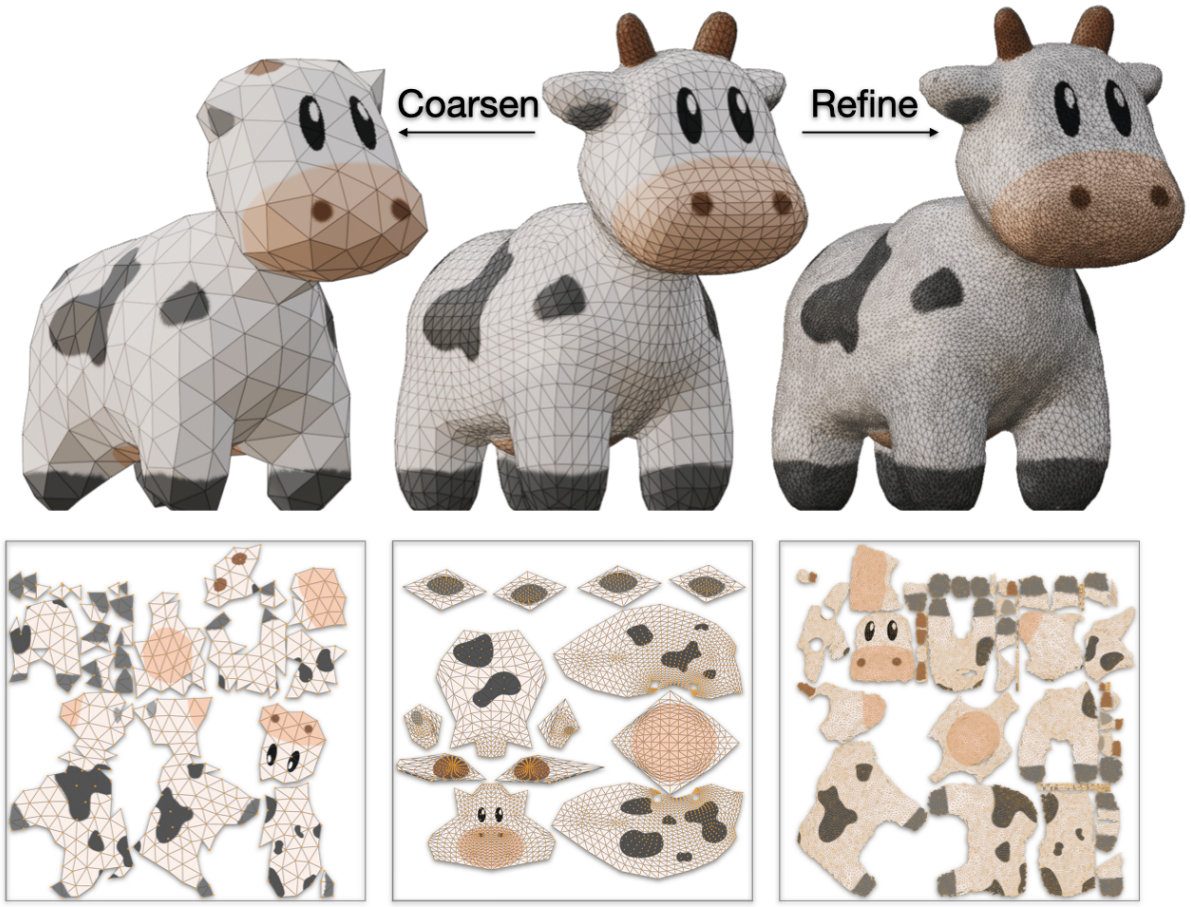


Figure 4.9: Texture transfer through mesh decimation and refinement. **Center**: the input spotted animal model[Crane et al. 2013] with its original UV parameterization and texture. **Top row**: remeshed results after decimation (left) and refinement (right), rendered with the transferred texture. **Bottom row**: the UV layouts and generated textures corresponding to fresh parameterizations of each remeshed model. Our bijective mapping enables texture transfer by composing the new parameterization with the tracked correspondence, eliminating the need to maintain UV coordinates during remeshing operations.

rounding errors.

We evaluated our curve tracking algorithm on 5,139 manifold triangle meshes from the Thingi10K dataset [Zhou and Jacobson 2016]. For each model, we performed isotropic remeshing and tracked axis-aligned planar curves through the operation sequence. Of these models, 4,998 (97.3%) completed successfully with topology preservation verified throughout. The remaining 141 models timed out after 12 hours, primarily due to extensive operation counts exceeding 700,000 per

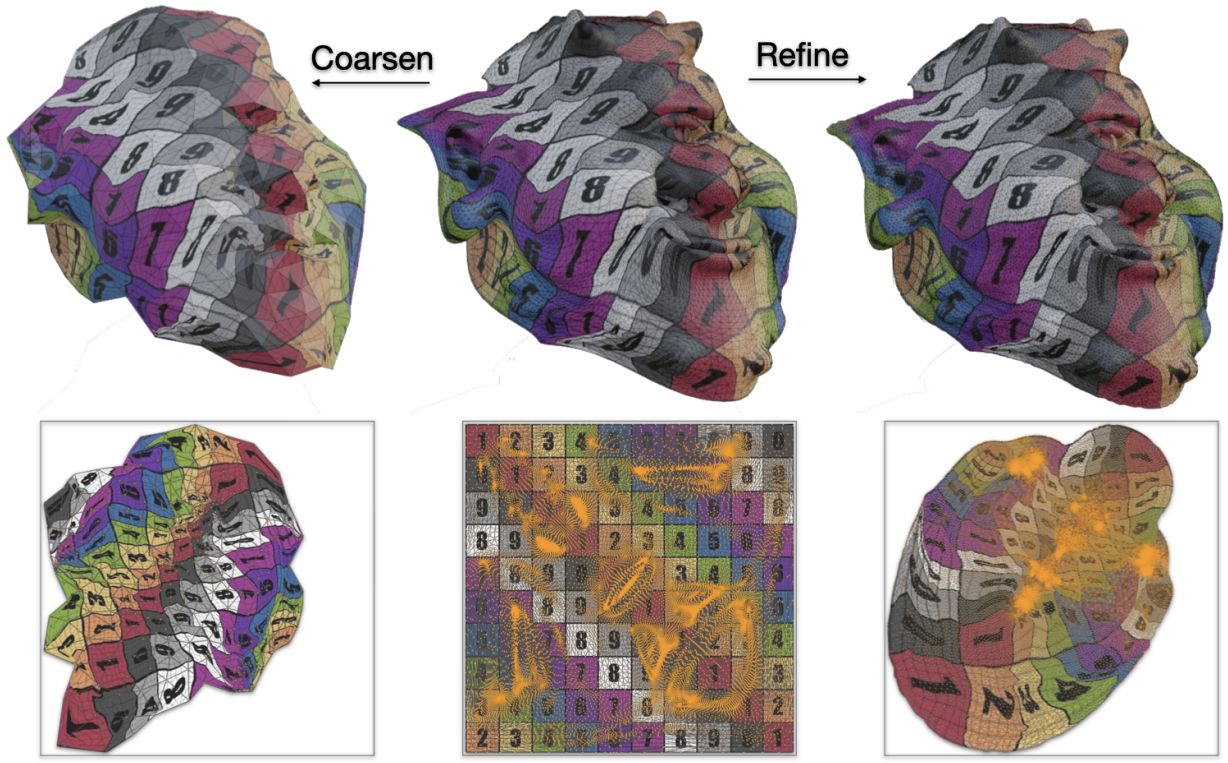


Figure 4.10: Texture transfer on the Ogre model with checkerboard pattern. The regular grid structure of the checkerboard provides a visual metric for evaluating mapping quality and distortion. Despite extensive geometric and connectivity changes from decimation (top left) and refinement (top right), the checkerboard pattern remains smooth and continuous, demonstrating effective distortion control through our bijective framework.

model. These timeouts represent computational resource limits rather than algorithmic failures—given sufficient computation time, they would yield topologically correct results identical to the successful cases. Furthermore, as we note in Section 4.5, the majority of this cost is due to computing each atlas as part of each topological operation. In reality this could be performed afterwards and in parallel.

Figure 4.12 showcases stress test examples where meshes undergo aggressive simplification with dramatic geometric changes. Despite surface features being substantially altered or eliminated, our method maintains topologically correct curve tracking—intersection patterns and connectivity remain intact. This demonstrates that our bijective framework preserves topological co-

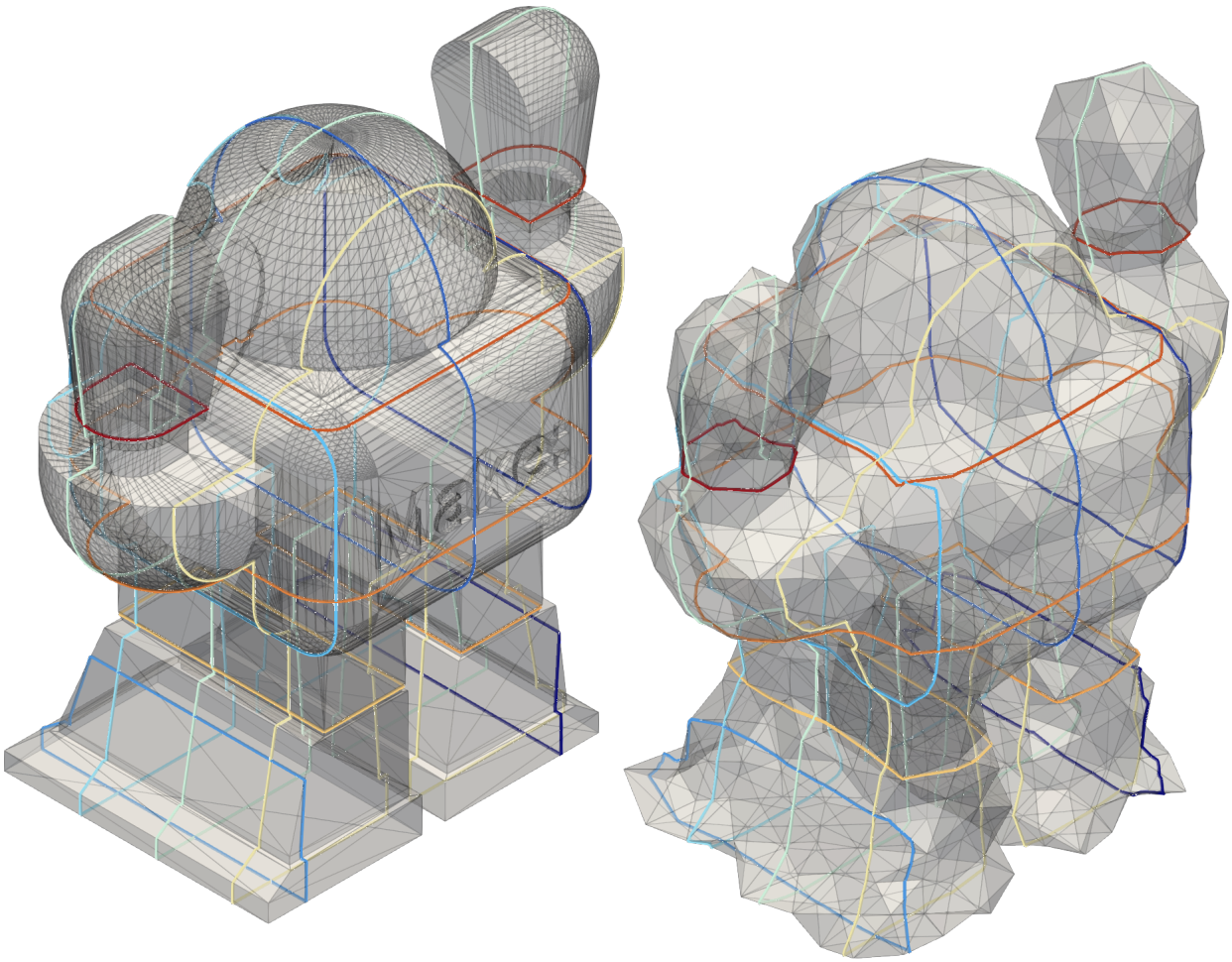


Figure 4.11: Curve tracking on a triangle mesh. **Left**: curves on the input mesh $\mathcal{M}_{\text{input}}$, obtained as intersections of the surface with axis-aligned planes (parallel to the xy, yz, and xz planes), shown in distinct colors. **Right**: the forward-tracked curves on the output mesh $\mathcal{M}_{\text{output}}$ after isotropic remeshing. Our method preserves the intersection topology of the curves throughout the remeshing sequence—curves that intersect on the input mesh maintain their intersection relationships on the output mesh.

herence even under severe geometric deformations: as long as the remeshing operations are topologically valid, our composed mappings produce correspondingly coherent results.

4.4.3 Surface Tracking on Tetrahedral Meshes

We demonstrate our surface tracking framework on medical imaging data by tracking organ segmentation boundaries through volumetric mesh simplification (Figure 4.13).

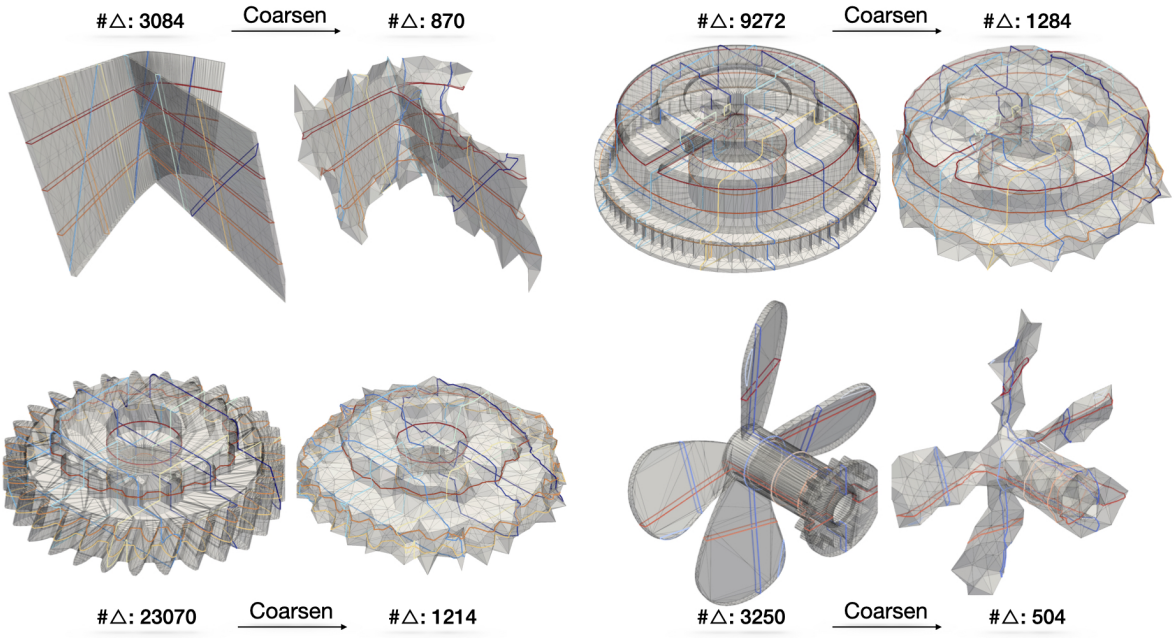


Figure 4.12: Stress test for curve tracking under extreme remeshing. Each pair shows input mesh (left) and aggressively simplified output (right) with tracked curves in color. Despite dramatic geometric changes, our method preserves topological correctness—curve intersections and connectivity remain intact throughout. Thingi10K models: #636811 (top-left), #1036656 (top-right), #1312974 (bottom-left), #1505135 (bottom-right).

Our input data is derived from the CTA Abdomen (Panoramix) sample dataset provided in 3D Slicer. We obtain organ segmentations using TotalSegmentator [Wasserthal et al. 2023], a deep learning-based tool for automatic whole-body CT segmentation. We extract the surface triangulation of six major organs (heart, liver, gallbladder, stomach, and kidneys) as separate surfaces to be tracked, with each surface embedded in the volumetric mesh via barycentric coordinates.

We perform mesh simplification on the background body mesh while tracking the organ surfaces through each local atlas. As shown in Figure 4.13, the tracked organ surfaces maintain their geometric shapes and relative positions after simplification. The topological relationships among organs are preserved: surfaces that were previously disjoint remain non-intersecting throughout the remeshing sequence.

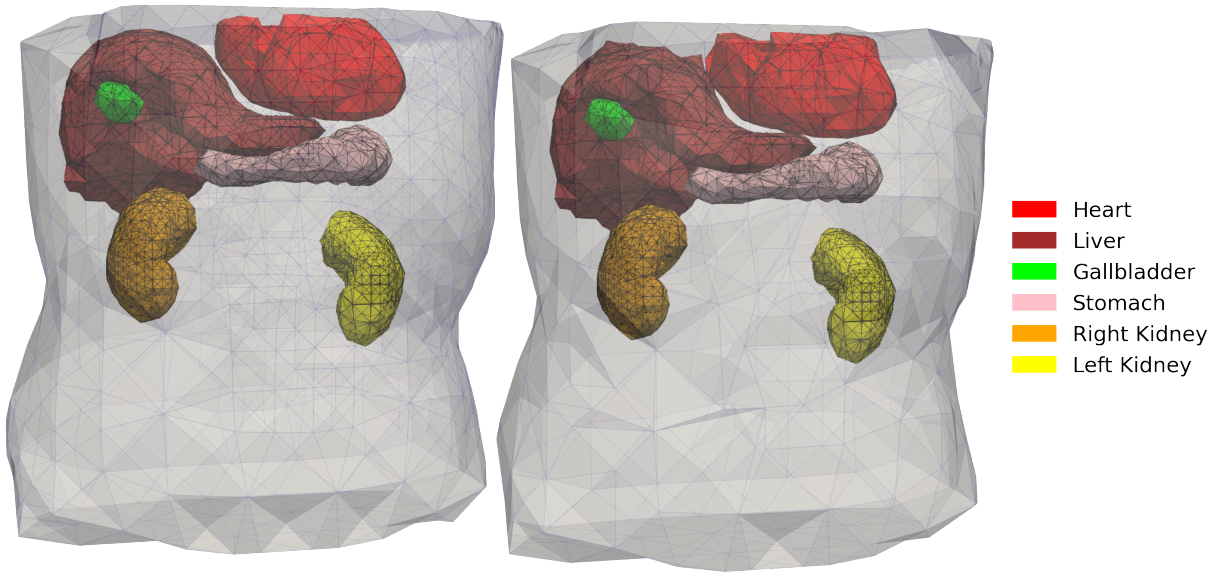


Figure 4.13: Surface tracking on a tetrahedral body mesh from CT scan data. We track multiple organ segmentation surfaces (heart, liver, gallbladder, stomach, and kidneys) through volumetric mesh simplification. **Left**: organ surfaces on the input tetrahedral mesh $\mathcal{M}_{\text{input}}$. **Right**: the tracked surfaces on the simplified output mesh $\mathcal{M}_{\text{output}}$. Our bijective framework maintains the topological relationships among organs throughout the remeshing process.

To evaluate surface tracking on a broader range of geometric models, we perform large-scale testing on tetrahedral meshes generated from the Thingi10K dataset [Zhou and Jacobson 2016] using TetWild [Hu et al. 2018] (Figure 4.1). For each input model, we first perform mesh simplification to obtain a coarsened output mesh $\mathcal{M}_{\text{output}}$. We then sample axis-aligned planar surfaces on this simplified mesh by intersecting it with planes parallel to the xy, xz, and yz coordinate planes.

We perform backward tracking to map these surfaces from $\mathcal{M}_{\text{output}}$ to the original high-resolution mesh $\mathcal{M}_{\text{input}}$. Critically, the intersection topology is preserved throughout the tracking process: the intersection graph $\mathcal{G}(\mathcal{S})$ representing how surfaces intersect remains combinatorially equivalent. As shown in Figure 4.1, surfaces that intersect on the simplified output mesh maintain their intersection relationships when back-tracked to the input mesh, with intersection curves faithfully reconstructed.

4.5 Conclusions

We have presented BijectiveRemesh, a robust framework for maintaining bijective mappings throughout complex remeshing sequences on both 2D triangle meshes and 3D tetrahedral meshes. Our approach guarantees global bijectivity through local atlas construction using two key innovations: shared scaffold structures for 2D operations and convex polyhedra embeddings for 3D boundary operations. These bijective mappings enable exact tracking of geometric entities—points, curves, and surfaces—with rigorous preservation of topological relationships, eliminating the artifacts common in projection-based transfer methods.

Limitations and Future Work. Constructing the bijective local atlases adds approximately 110× overhead per operation compared to performing remeshing operations alone. This overhead reflects the fact that our prototype augments each local patch with the auxiliary triangulation and performs iterative energy minimization with inversion-preventing line searches. We believe a substantial proportion of this overhead can be reduced through parallelization. Our current prototype implementation processes operations serially—constructing each local atlas sequentially as operations are applied. In practice, the atlas construction for different operations is independent: we can first record the local patch information for all operations during remeshing, then construct their corresponding local atlases in parallel. Since each operation's atlas construction is self-contained, this parallel formulation would reduce total execution time.

We currently also depend on exact rational arithmetic for geometric predicates in our current tracking implementation for the sake of robustness. While we employ various optimizations including rounding to double and curve simplification, the computational overhead of performing long sequences of rational arithmetic remains substantial, making the tracking process can become prohibitively slow.

An important direction for future work is developing floating-point tracking algorithms that maintain topological guarantees without exact arithmetic. This would require novel geometric predicates and consistency checks that can tolerate numerical errors while still preventing topological corruption in the tracked curves and surfaces.

5 Conclusion

5.1 Summary

This thesis studied how to construct mappings in geometry processing pipelines with explicit guarantees at geometric, topological, and correspondence levels.

In Chapter 2, we developed an efficient and robust method for computing discrete conformal metrics with prescribed Gaussian curvature in the interior and prescribed geodesic curvature along the boundary. By treating the intrinsic triangulation as a degree of freedom and carefully handling Delaunay-critical configurations, the method supports challenging inputs with high distortion and provides a practical path from curvature prescription to usable parametrizations.

In Chapter 3, we investigated the relation between cross-field topology and seamless parametrization topology. Building on this connection, we proposed a construction that simultaneously enforces local injectivity and provides full control over holonomy signatures, enabling global parametrizations (and downstream quadrangulations) that match prescribed topological structure.

In Chapter 4, we introduced *BijectiveRemesh*, a framework for maintaining a continuous, bijective mapping through complex remeshing sequences in both 2D triangle and 3D tetrahedral settings.

By composing local bijective atlases, the framework enables exact tracking of points, curves, and surfaces and avoids artifacts and inconsistencies common in projection-based transfer.

5.2 Limitations and Future Work

Several directions remain open. First, for discrete conformal metric computation, it is desirable to further reduce reliance on extended precision arithmetic via filtered strategies that preserve robustness while improving performance. Second, extending holonomy-controlled parametrization to surfaces with boundary and to stronger forms of feature-curve alignment would broaden applicability and requires additional theory. Third, for bijective remeshing and tracking, improving efficiency (e.g., through parallel atlas construction) and developing floating-point tracking methods that retain topological guarantees without exact arithmetic are important steps toward scaling to longer and more complex remeshing sequences.

Overall, the results of this thesis suggest that combining discrete-geometric constructions with careful topological reasoning makes strong guarantees attainable in practical mapping pipelines.

Appendices

A Proofs of Lemmas for Chapter 3

A.1 Proof of Proposition 1

Proof. We consider the case where α is a path between the right hand sides of the two loops. If the mesh is suitably refined, there is a topological disk D in the dual mesh that contains α and does not contain any cones (Figure A.1 left). Without loss of generality, we may assume that ∂D intersects γ along a single nontrivial path β_γ and intersects δ along a similar path β_δ. We denote the endpoints of β_γ as f_1^* and f_2^* and the endpoints of β_δ as g_1^* and g_2^*. We now can define γ_0 as the simple loop given by traversing $\gamma \setminus \beta_\gamma$ (with respect to the orientation of this loop) starting at f_2^*, then traversing the component of ∂D (with boundary orientation) from f_1^* to g_2^*, then traversing $\delta \setminus \beta_\delta$, and finally by traversing ∂D from g_1^* to f_2^*. We note that we can choose D so that the boundary is arbitrarily close to α and thus so γ_0 is arbitrarily close to the original loops and path.

Since ∂D is the boundary of a topological disk that does not contain any cones, we have that $\kappa^F_{\partial D} = 2\pi$. In the computation of $\kappa^F_{\gamma_0}$, we have that the signed angles satisfy $\alpha_{\gamma_0}(f^*) = \alpha_\gamma(f^*)$ for $f^* \in \gamma \setminus \beta_\gamma$ and $\alpha_{\partial D}(f^*) = -\alpha_\gamma(f^*)$ for $f^* \in \beta_\gamma \setminus \{f_1^*, f_2^*\}$ (Figure A.1 right). Furthermore, since α intersects γ on the right, we must have that the angle of f_1 that corresponds to $\alpha_\gamma(f_1^*)$ is on the left hand side of γ, a different angle of f_1 corresponds to $\alpha_{\partial D}(f_1^*)$ and is on the left hand side of

∂D, and the final angle of f_1 corresponds to $\alpha_{\gamma_0}(f_1^*)$ and is on the right hand side of γ_0. Therefore, we have that

$$\alpha_\gamma(f_1^*) + \alpha_{\partial D}(f_1^*) - \alpha_{\gamma_0}(f_1^*) = \pi,$$

and by a similar analysis the same result for f_2^* is obtained. The situation is similar for δ and ∂D, so we have that

$$\begin{aligned}\kappa_{\gamma_0}^F &= \sum_{f^* \in \gamma_0} \alpha_{\gamma_0}(f^*) \\ &= \sum_{f^* \in \gamma} \alpha_\gamma(f^*) + \sum_{f^* \in \delta} \alpha_\delta(f^*) + \sum_{f^* \in \partial D} \alpha_{\partial D}(f^*) - 4\pi \\ &= \kappa_\gamma^F + \kappa_\delta^F - 2\pi.\end{aligned}$$

Thus, we have that

$$k_{\gamma_0}^F = k_\gamma^F + k_\delta^F - 1.$$

The proof where α is on the left hand side of the two loops is analogous. □

A.2 Proof of Proposition 2

Proof. We have that cutting M along $\gamma_1, ..., \gamma_{2g}$ results in a disk, so, if the mesh is sufficiently refined, for any γ_i there is a path α_j/α_j' (Figure A.2 left) from either side of γ_i to any v_j^* such that neither path intersects any of the other basis loops or cones. Thus, by the above, we may reroute γ_i around v_j clockwise or counterclockwise to obtain a new loop γ_i' such that $H' = (H \setminus \{\gamma_i\}) \cup \{\gamma_i'\}$ also cuts M into a topological disk and such that, for any seamless parametrization F satisfying the properties listed in the proposition, we have

$$k_{\gamma_i'}^F = k_{\gamma_i}^F \pm I_{v_j}^F = k_{\gamma_i}^F \pm I_j$$

Since H' still cuts M to a disk, we may still reroute any loop around any cone with either orientation, so we may iteratively reroute the basis loops to modify their holonomy number by integer

multiples of I_j. Since $\frac{1}{4}$ is the greatest common divisor of $I_1, ..., I_m$, we have there are integers a_i such that

$$\frac{1}{4} = \sum_{i=1}^{m} a_i I_i$$

Thus, we have that iteratively rerouting each loop γ_i around the cone v_j $|4k_i a_j|$ times, with orientation determined by the signs of k_i and a_j, will give us the desired system of loops. □

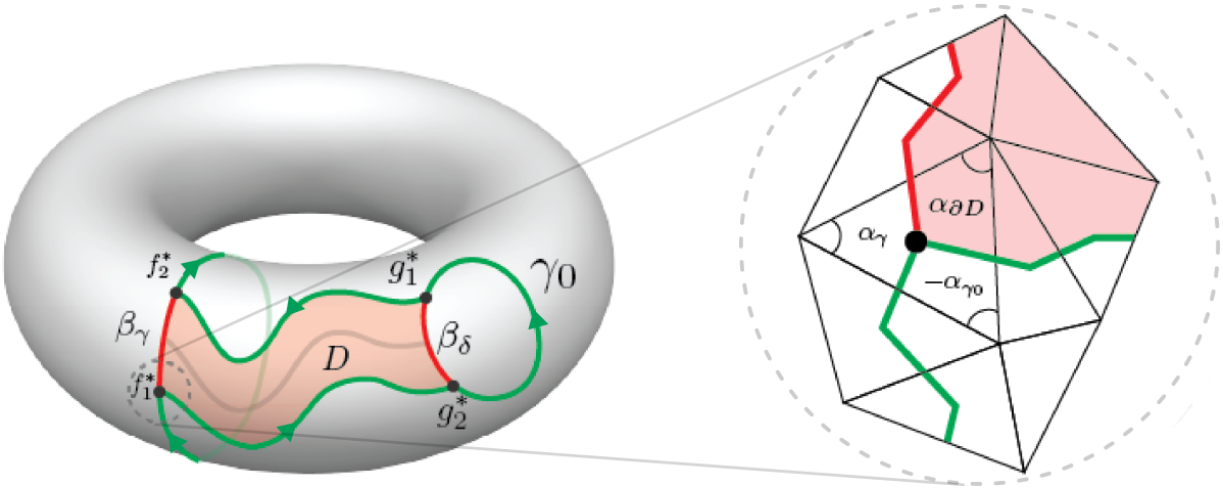


Figure A.1: Quasi-additivity of holonomy numbers, on the same example as in Figure 3.2. The inset on the right is a blow-up of the spot circled on the left.

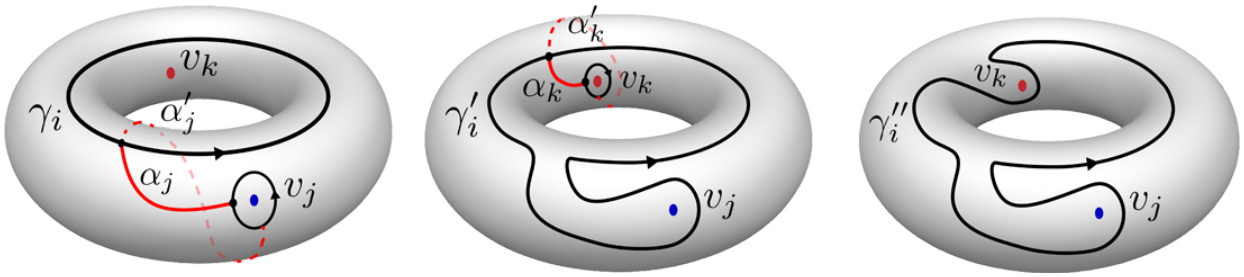


Figure A.2: Example of iteratively rerouting one loop around two singularities. Left: initial state with given loop γ_i and two paths α_j, α'_j connecting to the singularity v_j. Center: reroute around singularity v_j and find paths α_k α'_k for the next singularity v_k. Right: result after rerouting around v_j and v_k

B Supplementary Details for Chapter 4

B.1 Non-overlapping via Convex Boundary Embedding

Lemma B.1 (Non-overlapping via Convex Boundary Embedding)**.** *Let $\mathcal{P}_{before}$ be the local patch of boundary tetrahedra incident to vertex i. The boundary of this patch consists of:*

- *Face f_k: the one-ring of vertex i on $\partial\mathcal{M}^l$ (containing vertex i),*
- *Faces $f_1, \ldots, f_{k-1}$: the adjacent boundary triangles.*

Suppose the vertices of $\{f_1, \ldots, f_{k-1}, f_k\}$ (excluding i) are embedded to form a convex polyhedron C via Algorithms 2 and 3, where we select one triangle $f_0 \in \{f_1, \ldots, f_{k-1}\}$ as the outer boundary for the Tutte embedding.

Then for any position of vertex i on or inside face f_k, all tetrahedra in $\mathcal{P}_{before}$ are non-overlapping.

Proof. A tetrahedral mesh is non-overlapping if and only if for every internal face, the two opposite vertices lie on opposite sides of that face.

Consider an arbitrary internal face $f = (i, v_a, v_b)$ shared by two tetrahedra $T_1 = (i, v_a, v_b, v_c)$ and $T_2 = (i, v_a, v_b, v_d)$. We must show that v_c and v_d lie on opposite sides of the plane Π containing f.

Key observation. The face $f = (i, v_a, v_b)$ contains the edge (v_a, v_b), which is an edge of the convex polyhedron C. The vertices v_c and v_d are the two boundary vertices adjacent to this edge, forming boundary faces (v_a, v_b, v_c) and (v_a, v_b, v_d) of C.

Convexity guarantee. By the convexity of C, the dihedral angle at edge (v_a, v_b) is less than π. This means that for *any* plane Π containing edge (v_a, v_b), the vertices v_c and v_d lie on opposite sides of Π.

In particular, since vertex i is positioned on or inside face f_k of the convex polyhedron C, and the plane Π through face (i, v_a, v_b) contains edge (v_a, v_b), we have:

$$[(\mathbf{p}_c - \mathbf{p}_a) \cdot \mathbf{n}] \cdot [(\mathbf{p}_d - \mathbf{p}_a) \cdot \mathbf{n}] < 0,$$

where $\mathbf{n} = (\mathbf{p}_b - \mathbf{p}_a) \times (\mathbf{p}_i - \mathbf{p}_a)$ is the normal of Π.

Since this holds for every internal face, all tetrahedra in $\mathcal{P}_{\text{before}}$ are non-overlapping. □

B.2 Ray-Facet Intersection Predicates

This appendix provides the ray-facet intersection predicates used in curve tracking (Section 4.3.2.2).

B.2.1 Ray-Edge Intersection (2D)

Given a ray starting at $\mathbf{V} \in \mathbb{R}^2$ with direction $\mathbf{k}$, and an edge with endpoints $\mathbf{a}, \mathbf{b} \in \mathbb{R}^2$, we seek parameters t and u satisfying:

$$\mathbf{V} + t\mathbf{k} = \mathbf{a} + u(\mathbf{b} - \mathbf{a}). \tag{B.1}$$

Let $\mathbf{v}_1 = \mathbf{b} - \mathbf{a}$ and $\mathbf{v}_2 = \mathbf{a} - \mathbf{V}$. The determinant is:

$$\Delta = \mathbf{k}_x \mathbf{v}_{1y} - \mathbf{k}_y \mathbf{v}_{1x}. \tag{B.2}$$

If $\Delta \neq 0$:

$$t = \frac{\mathbf{v}_{2x}\mathbf{v}_{1y} - \mathbf{v}_{2y}\mathbf{v}_{1x}}{\Delta}, \qquad u = \frac{\mathbf{k}_x \mathbf{v}_{2y} - \mathbf{k}_y \mathbf{v}_{2x}}{\Delta}. \tag{B.3}$$

An intersection exists if $t > 0$ and $u \in [0, 1]$.

B.3 Ray-Triangle Intersection (3D)

Given a ray starting at $\mathbf{V} \in \mathbb{R}^3$ with direction $\mathbf{k}$, and a triangle with vertices $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^3$, we compute the normal $\mathbf{n} = (\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})$ and the ray-plane intersection:

$$t = \frac{\mathbf{n} \cdot (\mathbf{a} - \mathbf{V})}{\mathbf{n} \cdot \mathbf{k}}. \tag{B.4}$$

If $t > 0$, compute $\mathbf{p} = \mathbf{V} + t\mathbf{k}$ and its barycentric coordinates (w_0, w_1, w_2) with respect to $(\mathbf{a}, \mathbf{b}, \mathbf{c})$. The point lies inside the triangle if $w_i \geqslant 0$ for all i.

B.4 Implementation Notes

For robustness in topology-preserving curve tracking, these predicates should be evaluated using exact arithmetic. We represent all coordinates as rational numbers and perform all arithmetic operations exactly. While this incurs computational overhead, it guarantees that intersection tests are consistent and prevents numerical errors from violating topological invariants.

In practice, we first attempt intersection tests using double-precision floating-point arithmetic. If the result is near-degenerate (e.g., $\Delta \approx 0$ in 2D, or barycentric coordinates near boundary values), we fall back to exact rational arithmetic to ensure correctness.

Bibliography

Noam Aigerman and Yaron Lipman. 2015. Orbifold Tutte Embeddings. *ACM Trans. Graph.* 34, 6 (2015), 190:1–190:12.

Noam Aigerman and Yaron Lipman. 2016. Hyperbolic Orbifold Tutte Embeddings. *ACM Trans. Graph.* 35, 6, Article 217 (2016), 14 pages.

Marc Alexa. 2023. Tutte Embeddings of Tetrahedral Meshes. *Discrete & Computational Geometry* 73 (2023), 197–207.

Julien Basch, Leonidas J Guibas, and John Hershberger. 1999. Data structures for mobile data. *Journal of Algorithms* 31, 1 (1999), 1–28.

Mirela Ben-Chen, Craig Gotsman, and Guy Bunin. 2008. Conformal Flattening by Curvature Prescription and Metric Scaling. *Computer Graphics Forum* 27, 2 (2008).

Alexander I Bobenko and Boris A Springborn. 2007. A discrete Laplace–Beltrami operator for simplicial surfaces. *Discrete & Computational Geometry* 38, 4 (2007), 740–756.

David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. 2013a. Integer-Grid Maps for Reliable Quad Meshing. *ACM Trans. Graph.* 32, 4 (2013), 98:1–98:12.

David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis

Zorin. 2013b. Quad-Mesh Generation and Processing: A Survey. *Computer Graphics Forum* 32, 6 (2013).

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph.* 28, 3 (2009), 77.

Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. 2010. *Polygon mesh processing*. CRC press.

Alon Bright, Edward Chien, and Ofir Weber. 2017. Harmonic Global Parametrization with Rational Holonomy. *ACM Trans. Graph.* 36, 4 (2017).

Marcel Campen. 2017. Partitioning Surfaces Into Quadrilateral Patches: A Survey. *Computer Graphics Forum* 36, 8 (2017), 567–588.

Marcel Campen, David Bommes, and Leif Kobbelt. 2012. Dual Loops Meshing: Quality Quad Layouts on Manifolds. *ACM Trans. Graph.* 31, 4 (2012).

Marcel Campen, David Bommes, and Leif Kobbelt. 2015. Quantized global parametrization. *ACM Trans. Graph.* 34, 6 (2015), 192.

Marcel Campen, Ryan Capouellez, Hanxiao Shen, Leyi Zhu, Daniele Panozzo, and Denis Zorin. 2021a. Efficient and robust discrete conformal equivalence with boundary. *ACM Trans. Graph.* 40, 6 (2021).

Marcel Campen, Ryan Capouellez, Hanxiao Shen, Leyi Zhu, Daniele Panozzo, and Denis Zorin. 2021b. Efficient and Robust Discrete Conformal Equivalence with Boundary. *ACM Trans. Graph.* 40, 6 (2021).

Marcel Campen and Leif Kobbelt. 2014. Dual Strip Weaving: Interactive Design of Quad Layouts

Using Elastica Strips. *ACM Trans. Graph.* 33, 6 (2014), 183:1–183:10.

Marcel Campen, Hanxiao Shen, Jiaran Zhou, and Denis Zorin. 2019. Seamless Parametrization with Arbitrary Cones for Arbitrary Genus. *ACM Trans. Graph.* 39, 1 (2019).

Marcel Campen, Cláudio T. Silva, and Denis Zorin. 2016. Bijective maps from simplicial foliations. *ACM Transactions on Graphics* 35, 4 (2016), 74:1–74:15.

Marcel Campen and Denis Zorin. 2017a. *On Discrete Conformal Seamless Similarity Maps.* arXiv:1705.02422 [cs.GR]

Marcel Campen and Denis Zorin. 2017b. Similarity Maps and Field-Guided T-Splines: a Perfect Couple. *ACM Trans. Graph.* 36, 4 (2017).

Wei Chen, Xiaopeng Zheng, Jingyao Ke, Na Lei, Zhongxuan Luo, and Xianfeng Gu. 2019. Quadrilateral mesh generation I: Metric based method. *Computer Methods in Applied Mechanics and Engineering* 356 (2019), 652–668.

Wei Chen, Xiaopeng Zheng, Jingyao Ke, Na Lei, Zhongxuan Luo, and Xianfeng Gu. 2020. Quadrilateral Mesh Generation II: Meoromorphic Quartic Differentials and Abel-Jacobi Condition. *Computer Methods in Applied Mechanics and Engineering* 366 (2020).

Edward Chien, Zohar Levi, and Ofir Weber. 2016. Bounded Distortion Parametrization in the Space of Metrics. *ACM Trans. Graph.* 35, 6 (2016).

David Coeurjolly, Jaques-Olivier Lachaud, Konstantinos Katrioplas, Sébastien Loriot, Ivan Paden, Mael Rouxel-Labbé, Hossam Saeed, Jane Tournois, Sébastien Valette, and Ilker O. Yaz. 2025. Polygon Mesh Processing. In *CGAL User and Reference Manual* (6.1 ed.). CGAL Editorial Board. https://doc.cgal.org/6.1/Manual/packages.html#PkgPolygonMeshProcessing

Keenan Crane. 2020. Discrete Conformal Geometry. In *Proceedings of Symposia in Applied Mathematics*. American Mathematical Society.

Keenan Crane, Mathieu Desbrun, and Peter Schröder. 2010. Trivial Connections on Discrete Surfaces. *Computer Graphics Forum* 29, 5 (2010), 1525–1533.

Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.

Mathieu Desbrun, Mark Meyer, and Pierre Alliez. 2002. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum* 21, 3 (2002), 209–218.

Pablo Diaz-Gutierrez, David Eppstein, and Meenakshisundaram Gopi. 2009. Curvature aware fundamental cycles. *Computer Graphics Forum* 28, 7 (2009), 2015–2024.

Xingyi Du, Noam Aigerman, Qingnan Zhou, Shahar Z Kovalsky, Yajie Yan, Danny M. Kaufman, and Tao Ju. 2020. Lifting Simplices to Find Injectivity. *ACM Trans. Graph.* 39, 4 (2020), 120:1–120:18.

Hans-Christian Ebke, Patrick Schmidt, Marcel Campen, and Leif Kobbelt. 2016. Interactively Controlled Quad Remeshing of High Resolution 3D Models. *ACM Trans. Graph.* 35, 6 (2016), 218:1–218:13.

Jeff Erickson and Kim Whittlesey. 2005. Greedy optimal homotopy and homology generators. In *SODA*, Vol. 5. 1038–1046.

Danielle Ezuz, Justin Solomon, and Mirela Ben-Chen. 2019. Reversible Harmonic Maps between Discrete Surfaces. *ACM Transactions on Graphics* 38, 2 (2019).

Xianzhong Fang, Hujun Bao, Yiying Tong, Mathieu Desbrun, and Jin Huang. 2018. Quadrangulation through morse-parameterization hybridization. *ACM Trans. Graph.* 37, 4 (2018), 92.

Matthew Fisher, Boris Springborn, Peter Schröder, and Alexander I Bobenko. 2007. An algorithm for the construction of intrinsic Delaunay triangulations with applications to digital geometry processing. *Computing* 81, 2-3 (2007), 199–213.

Michael S. Floater. 1997. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design* 14, 3 (1997), 231 – 250.

Michael S. Floater. 2003. One-to-one piecewise linear mappings over triangulations. *Math. Comp.* 72, 242 (2003), 685–696.

Michael S. Floater and Valérie Pham-Trong. 2006. Convex combination maps over triangulations, tilings, and tetrahedral meshes. *Advances in Computational Mathematics* 25, 4 (2006), 347–356.

Xiao-Ming Fu, Yang Liu, and Baining Guo. 2015a. Computing Locally Injective Mappings by Advanced MIPS. *ACM Trans. Graph.* 34, 4 (2015).

Xiao-Ming Fu, Yang Liu, and Baining Guo. 2015b. Computing locally injective mappings by advanced MIPS. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–12.

Mark Gillespie, Boris Springborn, and Keenan Crane. 2021. Discrete Conformal Equivalence of Polyhedral Surfaces. *ACM Trans. Graph.* 40, 4 (2021).

Steven J. Gortler, Craig Gotsman, and Dylan Thurston. 2006. Discrete one-forms on meshes and applications to 3D mesh parameterization. *Computer Aided Geometric Design* 23, 2 (2006), 83 – 112.

Xianfeng Gu, Ren Guo, Feng Luo, Jian Sun, and Tianqi Wu. 2018a. A discrete uniformization

theorem for polyhedral surfaces II. *Journal of Differential Geometry* 109, 3 (2018), 431–466.

Xianfeng Gu, Feng Luo, Jian Sun, and Tianqi Wu. 2018b. A discrete uniformization theorem for polyhedral surfaces. *Journal of Differential Geometry* 109, 2 (2018), 223–256.

Xianfeng Gu and Shing-Tung Yau. 2003. Global conformal surface parameterization. In *Proc. Symp. Geometry Processing 2003*. 127–137.

Allen Hatcher. 2002. *Algebraic Topology*. Cambridge University Press.

Eden Fedida Hefetz, Edward Chien, and Ofir Weber. 2019. A Subspace Method for Fast Locally Injective Harmonic Mapping. *Computer Graphics Forum* 38, 2 (2019), 105–119.

K. Hormann and G. Greiner. 2000. MIPS: An Efficient Global Parametrization Method. In *Curve and Surface Design: Saint-Malo 1999*. Vanderbilt University Press, 153–162.

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4, Article 60 (July 2018), 14 pages. doi:10.1145/3197517.3201353

Zhongshi Jiang, Scott Schaefer, and Daniele Panozzo. 2017. Simplicial complex augmentation framework for bijective maps. *ACM Transactions on Graphics* 36, 6 (2017), 186:1–186:9.

Zhongshi Jiang, Teseo Schneider, Denis Zorin, and Daniele Panozzo. 2020. Bijective Projection in a Shell. *ACM Transactions on Graphics* 39, 6 (2020).

Zhongshi Jiang, Ziyi Zhang, Yixin Hu, Teseo Schneider, Denis Zorin, and Daniele Panozzo. 2021. Bijective and Coarse High-Order Tetrahedral Meshes. *ACM Transactions on Graphics* 40, 4 (2021).

Miao Jin, Junho Kim, and Xianfeng David Gu. 2007. Discrete surface Ricci flow: Theory and applications. In *IMA International Conference on Mathematics of Surfaces*. Springer, 209–232.

Ernest Jucovič and Marián Trenkler. 1973. A theorem on the structure of cell–decompositions of orientable 2–manifolds. *Mathematika* 20, 01 (1973), 63–82.

F. Kälberer, M. Nieser, and K. Polthier. 2007. QuadCover: Surface Parameterization using Branched Coverings. *Computer Graphics Forum* 26, 3 (2007), 375–384.

Liliya Kharevych, Boris Springborn, and Peter Schröder. 2006a. Discrete conformal mappings via circle patterns. *ACM Trans. Graph.* 25, 2 (2006), 412–438.

Liliya Kharevych, Boris Springborn, and Peter Schröder. 2006b. Discrete conformal mappings via circle patterns. *ACM Trans. Graph.* 25 (April 2006), 412–438. Issue 2.

P. Knupp. 1995. Mesh Generation Using Vector Fields. *J. Comput. Phys.* 119, 1 (1995), 142 – 148.

Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. 1998. Interactive multi-resolution modeling on arbitrary meshes. *SIGGRAPH* (1998), 105–114.

Denis Kovacs, Ashish Myles, and Denis Zorin. 2011. Anisotropic quadrangulation. *Computer Aided Geometric Design* 28, 8 (2011), 449 – 462. Solid and Physical Modeling 2010.

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

Vladislav Kraevoy and Alla Sheffer. 2004. Cross-parameterization and compatible remeshing of 3D models. *ACM Transactions on Graphics (TOG)* 23, 3 (2004), 861–869.

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least squares conformal

maps for automatic texture atlas generation. *ACM Transactions on Graphics* 21, 3 (2002), 362–371.

W. Li, B. Vallet, N. Ray, and B. Levy. 2006. Representing Higher-Order Singularities in Vector Fields on Piecewise Linear Surfaces. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (2006), 1315–1322.

Yaron Lipman. 2012. Bounded Distortion Mapping Spaces for Triangular Meshes. *ACM Trans. Graph.* 31, 4 (2012), 108:1–108:13.

Yaron Lipman. 2014. Bijective Mappings of Meshes with Boundary and the Degree in Mesh Processing. *SIAM Journal on Imaging Sciences* 7, 2 (2014), 1263–1283.

Hsueh-Ti Derek Liu, Jiayi Eris Zhang, Mirela Ben-Chen, and Alec Jacobson. 2021. Surface multigrid via intrinsic prolongation. *ACM Trans. Graph.* 40, 4, Article 80 (jul 2021), 13 pages. doi:10.1145/3450626.3459768

Feng Luo. 2004. Combinatorial Yamabe flow on surfaces. *Communications in Contemporary Mathematics* 6, 05 (2004), 765–780.

Max Lyon, Marcel Campen, David Bommes, and Leif Kobbelt. 2019. Parametrization Quantization with Free Boundaries for Trimmed Quad Meshing. *ACM Trans. Graph.* 38, 4 (2019).

Filippo Maggioli, Daniele Baieri, Emanuele Rodolà, and Simone Melzi. 2024. Rematching: Low-resolution representations for scalable shape correspondence. In *European Conference on Computer Vision*. Springer, 183–200.

Manish Mandad and Marcel Campen. 2020. Efficient piecewise higher-order parametrization of discrete surfaces with local and global injectivity. *Computer-Aided Design* 127 (2020).

Martin Marinov, Marco Amagliani, Tristan Barback, Jean Flower, Stephen Barley, Suguru Furuta, Peter Charrot, Iain Henley, Nanda Santhanam, G. Thomas Finnigan, Siavash Meshkat, Justin Hallet, Maciej Sapun, and Pawel Wolski. 2019. Generative Design Conversion to Editable and Watertight Boundary Representation. *Computer-Aided Design* 115 (2019), 194 – 205.

James R. Munkres. 2018. *Elements of Algebraic Topology*. CRC Press. doi:10.1201/9780429493911

Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust Field-aligned Global Parametrization. *ACM Trans. Graph.* 33, 4 (2014), 135:1–135:14.

Ashish Myles and Denis Zorin. 2012. Global parametrization by incremental flattening. *ACM Trans. Graph.* 31, 4 (2012), 109.

Ashish Myles and Denis Zorin. 2013. Controlled-distortion constrained global parametrization. *ACM Transactions on Graphics* 32, 4 (2013), 105.

Valentin Z. Nigolian, Marcel Campen, and David Bommes. 2023. Expansion Cones: A Progressive Volumetric Mapping Framework. *ACM Transactions on Graphics* 42, 4 (2023).

Valentin Z. Nigolian, Marcel Campen, and David Bommes. 2024. A Progressive Embedding Approach to Bijective Tetrahedral Maps driven by Cluster Mesh Topology. *ACM Transactions on Graphics* 43, 6 (2024).

Daniele Panozzo, Yaron Lipman, Enrico Puppo, and Denis Zorin. 2012. Fields on Symmetric Surfaces. *ACM Trans. Graph.* 31, 4 (2012).

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

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017b. Scalable Locally Injective Mappings. *ACM Transactions on Graphics* 36, 2 (2017).

Nicolas Ray, Bruno Vallet, Laurent Alonso, and Bruno Levy. 2009. Geometry-Aware Direction Field Processing. *ACM Trans. Graph.* 29, 1 (2009).

Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. 2008. N-Symmetry Direction Field Design. *ACM Trans. Graph.* 27, 2 (2008).

Ares Ribó Mor, Günter Rote, and André Schulz. 2011. Small grid embeddings of 3-polytopes. *Discrete & Computational Geometry* 45, 1 (2011), 65–87.

Igor Rivin. 1994. Euclidean structures on simplicial surfaces and hyperbolic volume. *Annals of mathematics* 139, 3 (1994), 553–580.

Pedro V Sander, John Snyder, Steven J Gortler, and Hugues Hoppe. 2001. Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 409–416.

Rohan Sawhney and Keenan Crane. 2017. Boundary First Flattening. *ACM Trans. Graph.* 37, 1 (2017).

Patrick Schmidt, Marcel Campen, Janis Born, and Leif Kobbelt. 2020. Inter-Surface Maps via Constant-Curvature Metrics. *ACM Transactions on Graphics* 39, 4 (2020).

Patrick Schmidt, Dörte Pieper, and Leif Kobbelt. 2023. Surface Maps via Adaptive Triangulations. *Computer Graphics Forum* 42, 2 (2023).

Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. 2013. Locally Injective Mappings. *Computer Graphics Forum* 32, 5 (2013), 125–135.

Nicholas Sharp and Keenan Crane. 2020. A Laplacian for nonmanifold Triangle Meshes. *Computer Graphics Forum* 39, 5 (2020), 69–80.

Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019. Navigating intrinsic triangulations. *ACM Transactions on Graphics* 38, 4 (2019), 1–16.

Hanxiao Shen, Leyi Zhu, Ryan Capouellez, Daniele Panozzo, Marcel Campen, and Denis Zorin. 2022. Which cross fields can be quadrangulated? global parameterization from prescribed holonomy signatures. *ACM Trans. Graph.* 41, 4, Article 59 (July 2022), 12 pages.

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

Jason Smith and Scott Schaefer. 2015a. Bijective Parameterization with Free Boundaries. *ACM Trans. Graph.* 34, 4, Article 70 (2015), 9 pages.

Jason Smith and Scott Schaefer. 2015b. Bijective parameterization with free boundaries. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–9.

Yousuf Soliman, Dejan Slepčev, and Keenan Crane. 2018a. Optimal cone singularities for conformal flattening. *ACM Transactions on Graphics* 37, 4 (2018), 1–17.

Yousuf Soliman, Dejan Slepčev, and Keenan Crane. 2018b. Optimal Cone Singularities for Conformal Flattening. *ACM Trans. Graph.* 37, 4 (2018), 105:1–105:17.

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry Processing*, Vol. 4. 109–116.

Boris Springborn. 2020. Ideal Hyperbolic Polyhedra and Discrete Uniformization. *Discrete &*

Computational Geometry 64, 1 (2020), 63–108.

Boris Springborn, Peter Schröder, and Ulrich Pinkall. 2008. Conformal Equivalence of Triangle Meshes. *ACM Transactions on Graphics* 27, 3 (2008), 1–11.

Ernst Steinitz. 1922. Polyeder und Raumteilungen. In *Encyklopädie der Mathematischen Wissenschaften*. Vol. 3.

Jian Sun, Tianqi Wu, Xianfeng Gu, and Feng Luo. 2015. Discrete conformal deformation: algorithm and experiments. *SIAM Journal on Imaging Sciences* 8, 3 (2015), 1421–1456.

The CGAL Project. 2025. *CGAL User and Reference Manual* (6.1 ed.). CGAL Editorial Board. https://doc.cgal.org/6.1/Manual/packages.html

Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun. 2006. Designing quadrangulations with discrete harmonic forms. *Symposium on Geometry Processing* (2006), 201–210.

William T. Tutte. 1963. How to Draw a Graph. *Proceedings of the London Mathematical Society* 13, 1 (1963), 743–767.

Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. 2016. Directional Field Synthesis, Design, and Processing. *Comp. Graph. Forum* 35, 2 (2016).

Jakob Wasserthal, Hanns-Christian Breit, Manfred T Meyer, Maurice Pradella, Daniel Hinck, Alexander W Sauter, Tobias Heye, Daniel T Boll, Joshy Cyriac, Shan Yang, et al. 2023. TotalSegmentator: robust segmentation of 104 anatomic structures in CT images. *Radiology: Artificial Intelligence* 5, 5 (2023), e230024.

Jeffrey R Weeks. 1993. Convex hulls and isometries of cusped hyperbolic 3-manifolds. *Topology*

and its Applications 52, 2 (1993), 127–149.

Tianqi Wu. 2014. *Finiteness of Switches in discrete Yamabe flow.* Master's thesis. Tsinghua University.

Jiaran Zhou, Marcel Campen, Denis Zorin, Changhe Tu, and Claudio T Silva. 2018. Quadrangulation of non-rigid objects using deformation metrics. *Computer Aided Geometric Design* 62 (2018), 3–15.

J. Zhou, C. Tu, D. Zorin, and M. Campen. 2020. Combinatorial Construction of Seamless Parameter Domains. *Computer Graphics Forum* 39, 2 (2020), 179–190.

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).

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

Denis Zorin. 2021. *Convergence Analysis of the Algorithm in "Efficient and Robust Discrete Conformal Equivalence with Boundary".* arXiv:2109.03436 [math.NA]