Thesis Proposal

Dynamic Intrinsic Geometry Processing

Mark Gillespie

December 2023

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Last updated: December 1, 2023

Abstract

This thesis presents algorithms and data structures for performing robust computation on surfaces that evolve over time. Throughout scientific and geometric computing, surfaces are often modeled as *triangle meshes*. However, finding high-quality meshes remains a challenge because meshes play two distinct and often-conflicting roles: defining both the surface geometry and a space of functions on that surface.

One solution to this dilemma, which has proven quite powerful in recent years, is the use of *intrinsic triangulations* to decouple these two concerns. The key idea is that given a triangle mesh representing an input surface, one can find many alternative triangulations which encode the exact same intrinsic geometry but offer alternative function spaces to work in. This technique makes it easy to find high-quality intrinsic triangle meshes, sidestepping the tradeoffs of classical mesh construction. However, the fact that intrinsic triangulations exactly preserve the input geometry—one of the central benefits of the technique—also makes it challenging to apply to surfaces whose geometry changes over time.

In this thesis we relax the assumption of exact geometry preservation, allowing the intrinsic perspective to be applied to time-evolving surfaces. We take as examples the problems of mesh simplification and surface parameterization. In the case of mesh simplification, we provide a general-purpose data structure for intrinsic triangulations which share only the topological class of the input surface, but may feature different geometry. In the case of surface parameterization, we build more efficient data structures and algorithms for the special case where the geometry changes *conformally*, using a connection between discrete conformal maps and hyperbolic geometry. In both cases, we find that the intrinsic perspective leads to simple algorithms which are still robust and efficient on a variety of examples.

Contents

Chapter 1

Introduction

"You see," Mrs. Whatsit said, "if a very small insect were to move from the section of skirt in Mrs. Who's right hand to that in her left, it would be quite a long walk for him if he had to walk straight across."

Madeleine L'Engle, *A Wrinkle in Time*

The distinction between intrinsic and extrinsic properties has played a central role throughout the history of differential geometry, dating back to pioneering work by Gauß [1825] and Riemann [1854] in the early 19th century. Intrinsic geometry describes the properties of a surface which depend on local measurements *along* the surface like lengths or angles, independent of the surface's embedding in space. A common metaphor—referenced above—is that intrinsic geometry takes the perspective of an ant walking along the surface. By contrast, extrinsic geometry encompasses the properties which do depend on the embedding of a surface. Importantly, one can consider surfaces which have only intrinsic geometry, without any embedding into ambient space. This intrinsic perspective is famously used in general relativity, where one considers the curved spacetime of our universe without requiring any larger space for the universe to "curve into".

More recently, the intrinsic perspective has become a useful tool in geometry processing, where it allows one to work with triangle meshes which are not embedded into $\mathbb{R}^3$—and even meshes which *cannot* be embedded into $\mathbb{R}^3$. This opens up a larger space of meshes to work in, providing meshes of much higher quality than is possible extrinsically, while still supporting a wide variety of geometry processing tasks. However, existing techniques apply only as a precomputation for static objects: they find alternative representations of fixed objects, but these representations immediately become invalid if the object deforms.

This thesis explores two settings where intrinsic triangulations sit atop changing geometries:

1. Maintaining an intrinsic triangulation while coarsening a surface to perform intrinsic simplification (Chapter 3, [Liu et al. 2023]).
2. Using an intrinsic triangulation while parameterizing a surface via discrete conformal maps (Chapter 4, [Gillespie et al. 2021b]).

In addition, Chapter 5 proposes a possible extension of the intrinsic coarsening technique to simplify nonmanifold meshes, and concludes with my planned graduation timeline.

Chapter 2

Background & Related Work

The start of intrinsic geometry was made by Gauss' paper "Disquisitiones generales circa superficies curvas," which appeared in 1827. Since that time, intrinsic geometry has advanced so far that, at present, all of its major issues can be considered solved, at least those that deal with the geometry of small pieces of regular surfaces … Meanwhile, irregular surfaces merit no less consideration, as they often occur in real life and can be made from, say, a sheet of paper. For example, any polyhedron or cone, or the surface of a lens with sharp edges are not regular. It is no wonder then that there is a need to study irregular surfaces, too.

A. D. Alexandrov [1948]

Perhaps the first major result concerning the intrinsic geometry of *polyhedra*—rather than smooth surfaces—was Alexandrov's uniqueness theorem for embeddings of convex polyhedra in the 1940s [Alexandrov 1942]. A decade and a half later Regge [1961] took inspiration from Alexandrov's work and used intrinsic polyhedra in his study of numerical general relativity. It is fitting that general relativity, which motivated much of the development of smooth differential geometry, was also a key impetus behind the devlopment of discrete differential geometry.

From there, the study of polyhedral intrinsic geometry branched in several direction: Troyanov [1986] developed the smooth theory of polyhedra in his study of smooth conformal maps between polyhedral surfaces, while Rivin [1994] defined intrinsic Delaunay triangulations of polyhedral surfaces and introduced the deep connections between Euclidean and ideal hyperbolic polyhedra. He also introduced the flip algorithm for computing intrinsic Delaunay triangulations. Several years later Indermitte et al. [2001] fixed a flaw in Rivin's proof of correctness of the flip algorithm, although they themselves left open the possibility of a topological obstruction which was only ruled out by Bobenko & Springborn [2007]. Glickenstein [2005, 2023] generalized the intrinsic Delaunay triangulation and edge flip algorithm, introducing increasingly broad classes of triangulations such as weighted, Thurston, and duality triangulations, sharing many of the important properties of intrinsic Delaunay triangulations. Bobenko & Izmestiev [2008] used weighted Delaunay triangulations to provide a constructive proof of Alexandrov's theorem on isometric embeddings of convex polytopes.

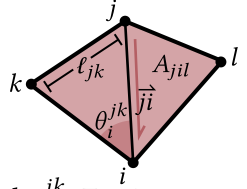
In parallel, intrinsic geometry has developed through the study of discrete conformal maps, starting with the work of Roček & Williams [1984] on discrete conformal field theories. Discrete conformal maps were rediscovered in mathematics by Luo [2004] in his work on combinatorial Yamabe flow, starting a line of work which culiminated in the discrete uniformization theorem for polyhedral surfaces [Gu et al. 2018a; b; Springborn 2019].

The computer graphics community was introduced to intrinsic geometry by the work of Kharevych et al. [2006] and Springborn et al. [2008] on discrete conformal parameterization, and the work of Fisher et al. [2007] exploring the benefits of intrinsic Delaunay triangulations. de Goes et al. [2014] went on to explore applications of weighted triangulations in geometry processing, including architectural design and mesh generation. More recently, Sharp et al. [2019] proposed a lightweight data structure for computing with intrinsic triangulations, alongside a suite of novel intrinsic retriangulation algorithms beyond the intrinsic Delaunay flips introduced by Rivin in the '90s. Gillespie et al. [2021a] introduced a more robust data structure for encoding intrinsic triangulations. Since then, intrinsic triangulation have proved useful in a variety of contexts, from spectral geometry processing [Fumero et al. 2020] to mesh deformation [Finnendahl et al. 2023]. A more exhaustive survey of the literature on intrinsic triangulations and their applications was provided by Sharp et al. [2021].

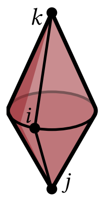
Notably, all past work has considered intrinsic triangulations which are *isometric* to the reference input surface—*i.e.* intrinsic triangulations which preserve the input geometry exactly. In this thesis, we introduce data strucures an algorithms for manipulating intrinsic triangulations of surfaces whose geometry changes over time. These data structures necessarily accomodate intrinsic triangulations whose geometry differs in various ways from the input geometry.

In the rest of this chapter, we review our notation and conventions (Section 2.1), before providing an introduction to the concepts from smooth (Section 2.2) and discrete (Section 2.3) differential geometry used in this thesis, and a discussion of intrinsic triangulations (Section 2.4).

2.1 Notation & Conventions



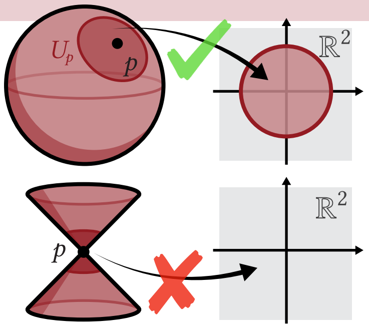
Throughout, we consider a manifold triangle mesh M with vertex set V, edge set E and face set F. We denote vertices by indices $i \in V$ and edges and faces by tuples $ij \in E$, $ijk \in F$ respectively. We also denote oriented halfedges by $\overrightarrow{ij} \in H$. The value of a function $u : V \to \mathbb{R}$ at vertex i is written as u_i; similarly, values on edges are denoted u_{ij} and values on faces are denoted u_{ijk}. A value at the corner of face ijk incident on vertex i is denoted u_i^{jk}. For instance, the position of a vertex may be denoted p_i, the length of an edge ℓ_{ij}, the area of a face A_{ijk}, or the angle of a corner θ_i^{jk}. Since our meshes are allowed to be general Δ-complexes, they may feature *e.g.* multiple edges between the same pair of vertices (the details are discussed more in Section 2.3.1). Consequently, our edge notation ij may not specify a unique edge from the mesh—it is merely used to indicate a particular edge that happense to go from vertex i to vertex j (where i may equal j). Similar caveats apply to the notation for faces.



Defining the *degree* of a vertex in a Δ-complex also requires some care, as the same edge may be incident on a vertex more than once. We define the degree $\deg(i)$ as the number of incident edges counted with multiplicity, *i.e.*, +2 for a self-edge from i back to i, and +1 for any other edge ij with $j \neq i$. For instance, in the inset figure vertex i has degree four, even though it is contained in only three distinct edges; vertices j and k both have degree one.

When our mesh changes over times, we denote the original mesh by $M = (V, E, F)$ and the modified mesh by $\widetilde{M} = (\widetilde{V}, \widetilde{E}, \widetilde{F})$. Quantities on the modified mesh are indicated in the same way, so *e.g.* the edge lengths on the modified mesh are denoted by $\tilde{\ell} : \widetilde{E} \to \mathbb{R}_{\geq 0}$.

2.2 Manifolds

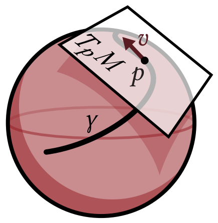


Informally speaking, a manifold is a space which "looks like $\mathbb{R}^n$ everywhere". More formally, a manifold M is a topological space where every point $p \in M$ is contained in some open set U_p which is in continuous bijection with an open set in $\mathbb{R}^n$. So a sphere is a manifold, since every point on the sphere is contained in a disk which can be mapped continuously to the unit disk in the plane. By contrast, a double-knapped cone is not a manifold, because no neighborhood of the tip can be mapped to the plane by a continuous bijection. Each mapping to the plane is called a *chart*, as it describes the "terrain" around a point p, and the collection of all of these charts is referred to as an *atlas*.

So far, our notion of a manifold is purely topological—since our charts are continuous maps, we have defined manifolds as spaces which look like $\mathbb{R}^n$ everywhere *as topological spaces*. To move from the world of topology to geometry, we have to equip our manifolds with additional structure. The two most important examples in this thesis will be *smooth structure* (Section 2.2.1)—allowing us to talk about differentiability of functions in addition to continuity—and *Riemannian structure* (Section 2.2.2)—allowing us to measure lengths and angles along the surface.

2.2.1 Smooth Structure

A smooth structure on a manifold M determines which functions $f : M \to \mathbb{R}$ are smooth (*i.e.* infinitely differentiable). Traditionally, this is expressed using a special atlas of charts which is declared to be smooth: then a function f is smooth on M if its expression in each chart is a smooth function on $\mathbb{R}^n$. As long as the charts satisfy some simple compatibility conditions on their overlaps, then this definition is independent of which particular charts we select from the atlas to check f's smoothness [Lee 2012, Chapter 1].



Tangent Spaces A smooth structure on M allows one to define *tangent spaces* associated to points on M. Conceptually, the tangent space T_pM represents the plane tangent to M at point p, giving a linear approximation of the domain around this point. Somewhat counterintuitively, these tangent spaces can be defined purely intrinsically, without any embedding to provide the position of M in $\mathbb{R}^n$. The key idea is that tangent vectors can be thought of as derivatives of curves lying on M. Since the smooth structure allows us to talk about derivatives of curves, this is enough to define tangent vectors, and hence tangent spaces. The *tangent bundle* TM is the collection of all tangent spaces:

$$TM := \bigsqcup_{p \in M} T_pM. \tag{2.1}$$

Any smooth mapping $f : M \to N$ between manifolds M and N has a linear approximation $df : TM \to TN$ which sends tangent vectors on M to tangent vectors on N. This map is often called the differential, or push-forward, since it pushes vectors from M to N.

It is important to note that these tangent spaces are defined only as abstract vector spaces, without any canonical choice of basis or inner product. So we can do arithmetic with tangent vectors—we can add them together or scale them up and down—but we cannot yet measure the lengths of vectors or angles between them. In the next section we will equip our surfaces with a Riemannian metric which will provide us with an inner product on our tangent spaces.

Similarly, since tangent vectors based at different points on the surface live in different tangent spaces, we cannot compare vectors which live at different points—*e.g.* we cannot ask whether two vectors point in the "same" direction, since they are elements of different vector spaces. Later on, we will see how a Riemannian metric also allows us to relate vectors in different tangent spaces through parallel transport.

Uniqueness of Smooth Structure Topological manifolds of dimension $n \leq 3$ have a unique smooth structure (up to diffeomorphism). Interestingly, the standard proof begins by showing that these manifolds can be triangulated piecewise-linearly [Moise 1952], and then proceeding to show that such triangulations can be smoothed to obtain a smooth structure [Hirsch & Mazur 1974]. Triangulations, which we use to encode polyhedral surfaces in Section 2.3.1 are also an essential tool in the continuous setting. On the other hand, higher-dimensional manifolds may have different smooth structures; for instance, Milnor [1956] famously constructed smooth structures on the 7-sphere inequivalent to the standard one.

2.2.2 Riemannian Structure

A Riemannian structure on M allows us to start doing *geometry*, measuring lengths and angles and so forth for curves running along M. Formally, this structure is usually encoded via a Riemannian metric g, which provides an inner product on each tangent space of M. That is to say, at each point p we have a symmetric, bilinear, positive-definite form $g_p : T_pM \times T_pM \to \mathbb{R}$. To emphasize that g_p is an inner product on T_pM, we will sometimes write $g_p(X, Y)$ as $\langle X, Y \rangle_g$ for vectors $X, Y \in T_pM$, and similarly, we will write $g_p(X, X)$ as $\|X\|_p^2$ for vectors $X \in T_pM$.

Isometries An isometry is a smooth mapping $f : M \to N$ between manifolds M and N which preserves the metric. So, for instance, if γ is a curve on M, then $f \circ \gamma$ is a curve of the same length on N. Similarly, if two curves γ_1 and γ_2 meet at an angle θ on M, then the curves $f \circ \gamma_1$ and $f \circ \gamma_1$ must meet at the same angle θ on N.

Geodesics To measure the length of a curve $\gamma : [0, T] \to M$, we add up the size of its velocity at all times from 0 to T. More formally, the length of γ can be expressed as the integral

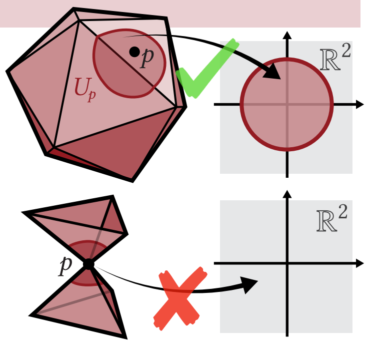
$$L(\gamma) := \int_0^T \|\dot{\gamma}(t)\|_g \, dt, \tag{2.2}$$

where $\dot{\gamma} : [0, T] \to TM$ is the derivative of γ. Now that we can measure the lengths of curves along M, we can also define distances between points in M: the distance between points x and y

on M is simply the length of the shortest path from x to y. If M is equal to $\mathbb{R}^n$ with the standard metric, then the shortest line between two points will be a straight line connecting them. On general Riemannian manifolds, then, we thing of shortest paths as generalizations of straight lines.

A *geodesic* is a (unit-speed) curve γ which is a locally shortest path, in the sense that for sufficiently close times s and t, γ follows the shortest path between $\gamma(s)$ and $\gamma(t)$. However, γ itself may not be the shortest path between its endpoints. A classic example of a geodesic which is not a shortest path is a curve wrapping around the equator of the sphere. This is locally shortest, since over any short time interval it follows a shortest path, but of course walking all of the way around the sphere is longer than not moving at all.

2.3 Polyhedral Surfaces



Informally speaking, a polyhedral surface is a collection of triangles which have been glued together to form a manifold. Just as in the smooth setting, a triangulated sphere is a manifold, whereas a double-knapped pyramid is not. And like in the smooth setting, we can consider different structures on a polyhedral surface: we can consider a triangulation, which yields topological information about the surface (Section 2.3.1), and we can consider a metric (Section 2.3.2), which yields geometric information about the surface.

2.3.1 Triangulations

In this thesis, we will represent the connectivity of a polyhedral surfaces using a triangulation. More precisely, we use a Δ-complex, which consists of a collection of disjoint triangles along with a prescribed gluing which attaches their vertices and edges together. Explicitly, this amounts to a collection of triangles $i_0 j_0 k_0, \ldots i_{|F|} j_{|F|} k_{|F|}$, alongside a list of vertex gluings $a \sim b$ and a list of edge gluings $(a, b) \sim (c, d)$, where a, b, c, d are vertices from the disjoint triangles. Figure 2.1 shows some examples; a more formal definition is provided by Hatcher [2002, Section 2.1]. Finally, throughout this thesis we consider only pure 2-complexes, *i.e.*, we require that every vertex and edge is contained in some triangle (and triangles are the cells of greatest dimension).

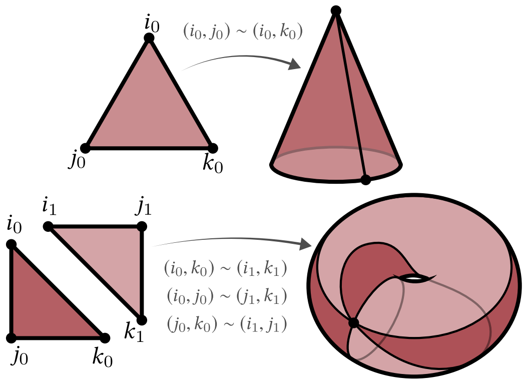

i_0
$(i_0, j_0) \sim (i_0, k_0)$
j_0
k_0
i_1
j_1
i_0
$(i_0, k_0) \sim (i_1, k_1)$
$(i_0, j_0) \sim (j_1, k_1)$
$(j_0, k_0) \sim (i_1, j_1)$
k_1
j_0
k_0

Figure 2.1: In a Δ-complex, the vertices of an edge or triangle are not required to be distinct. One can build a cone by gluing together two edges of the same triangle (top), or a torus out of two triangles and just a single vertex (bottom).

Existence Radó [1925] showed that every surface can be triangulated, which is to say that every topological 2-manifold is homeomorphic to some Δ-complex [Moise 2013, Chapter 8]. Perhaps the more surprising fact is that this theorem is not true in higher dimensions: Kirby &

Siebenmann [1969] showed that topological manifolds of dimension ≥ 6 are triangulable if and only if a certain cohomology class $\kappa(M) \in H^4(M; \mathbb{Z}/2\mathbb{Z})$ vanishes. However, Whitehead [1940] showed that every smooth (or even just C^1) manifold can be triangulated. In any case, this thesis only considers surfaces, where working with triangulations is much more straightforward.

Equivalence Two triangulations of a manifold are said to be combinatorially equivalent if they have a *common subdivision*, *i.e.* if there is a finer triangulation which can express all faces of both triangulations as unions of its finer faces. In 1908, Steinitz [1908] and Tietze [1908] conjectured that any two triangulations of a topological manifold are combinatorially equivalent, which came to be known as the *Hauptvermutung* (main conjecture) of geometric topology. This conjecture holds true in dimensions two [Radó 1925] and three [Moise 1952], but again fails in higher dimensions [Kirby & Siebenmann 1969].

Although any two triangulations of a given surface are combinatorially equivalent, the choice of triangulation can have dramatic impacts in practice.

Data Structures Perhaps the most common mesh data structure is the *vertex-face adjacency list*, which simply stores the three three vertices associated with each triangle. This representation is simple and easy to use, as it requires only an $|F| \times 3$ matrix. However, a vertex-face adjacency list alone does not provide enough information to encode a general Δ-complex: it provides a vertex gluing map, but leaves the edge gluing map implicit. When working with extrinsic triangle meshes in $\mathbb{R}^3$ this information is sufficient to recover the triangulation, but when working intrinsically one must store more information.

Fortunately, many of the other standard mesh data structures can be used out of the box to represent general Δ-complexes. For instance one can use winged-edge or halfedge structures [Baumgart 1975; Kettner 1999; Weiler 1985]; Botsch et al. [2010] provide an accessible introduction to these data structures. Alternatively, Sharp & Crane [2020a] observe that one can simply augment the vertex-face adjacency list with an additional array storing the edge gluing map to fully encode a general Δ-complex.

Tangent Spaces Away from vertices, tangent vectors on a manifold triangulation are straightforward to reason about, especially in our setting of interest where the triangles are given flat Euclidean metrics (Section 2.3.2). But even at vertices, there are still well-defined tangent spaces. After all, a manifold 2-dimensional Δ-complex is in particular a topological surface, which has a unique smooth structure. In the next section, we will use a metric on the triangulation to construct a convenient parameterization for these tangent spaces.

2.3.2 Polyhedral Geometry

A polyhedral cone metric on a surface M with vertex set V is a smooth Riemannian metric on the punctured surface $M \setminus V$ which is intrinsically flat everywhere. Such a metric can be encoded by a set of positive edge lengths $\ell : E \to \mathbb{R}_{>0}$ satisfying the triangle inequality $\ell_{ij} + \ell_{jk} > \ell_{ki}$ at each triangle corner; conversely, any such set of lengths determines a valid intrinsic metric. Under this metric, each triangle is isometric to a standard Euclidean triangle with the presecribed edge lengths. One typically obtains initial edge lengths $\ell_{ij} = \|p_i - p_j\|$ from input vertex positions $p : V \longrightarrow \mathbb{R}^3$, but in principle this could be any abstract metric (e.g., coming from a *cone flattening* [Bobenko & Springborn 2004]). As usual, this metric allows us to measure lengths and angles along the surface. For instance, triangle corner angles $\theta_i^{jk} \in (0, \pi)$ can be determined from the edge lengths via the law of cosines. Sharp et al. [2021, Appendix A] provide a detailed explanation of how to compute many geometric quantities of interest from edge lengths.

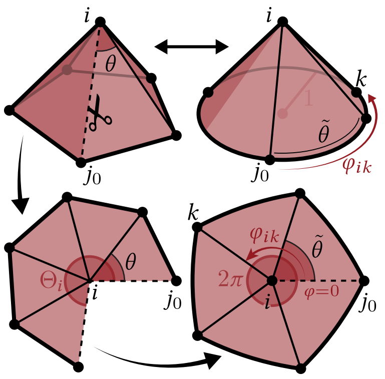
Curvature Although a polyhedral cone metric is flat almost everywhere, it still has a meaningful notion of curvature: each interior vertex i has an associated discrete Gaussian curvature

$$\Omega_i := 2\pi - \sum_{ijk \ni i} \theta_i^{jk}. \tag{2.3}$$

This angle defect measures the deviation of vertex i from being flat, and can be interpreted as the integral of Gaussian curvature over a small surface patch containing vertex i. Similarly, each boundary vertex has an associate discrete geodesic curvature

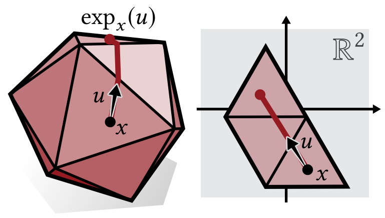
$$\kappa_i := \pi - \sum_{ijk \ni i} \theta_i^{jk}, \tag{2.4}$$

measuring the deviation of the boundary from a straight line around i.



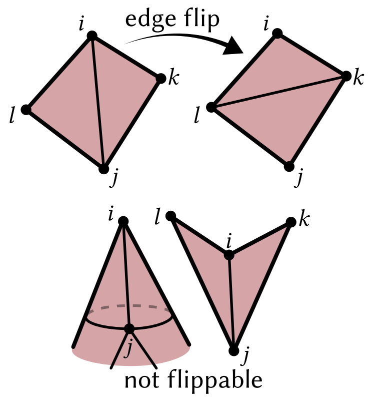
Vertex Tangent Spaces Even though the smooth polyhedral metric is not technically defined at a vertex i, it still provides us with a convenient parameterization of the tangent space T_iM. Any sufficiently small neighborhood of vertex i is isometric a cone of total angle Θ_i. Following Knöppel et al. [2013, Section 6], we express the direction of any tangent vector $v \in T_iM$ as a normalized angle $\varphi := 2\pi\theta/\Theta \in [0, 2\pi)$, where θ is the angle of v relative to an arbitrary but fixed reference edge ij_0, and Θ is the total angle sum at vertex i. The vector itself is then encoded as a complex number $re^{\mathbf{i}\varphi} \in \mathbb{C}$, where $\mathbf{i}$ is the imaginary unit and r is the vector's magnitude. Note that although we have used the metric to define a particular coordinate system on T_iM, the tangent space itself is well-defined independent of our choice of metric.

Parallel Transport The tangent spaces at adjacent vertices i and j are *a priori* just a pair of abstract vector spaces which are entirely unrelated to each other. However, once we have equipped our surface with a polyhedral metric we can use parallel transport to map vectors between the two tangent spaces. Concretely, we let the angular coordinate $\varphi_{ij} \in [0, 2\pi)$ encode the outgoing direction of an oriented edge ij from vertex i; we use $e_{ij} \in T_iM$ to denote the vector with direction φ_{ij} and magnitude ℓ_{ij}. The corresponding direction at vertex j is given by $\varphi_{ji} + \pi$. Hence, we can parallel transport vectors from T_iM to T_jM following edge ij by applying a rotation $R_{ij} := e^{\mathbf{i}((\varphi_{ji}+\pi)-\varphi_{ij})}$ (encoded as a unit complex number). See Sharp et al. [2019, §3.3 & §5.2] for further discussion.



Exponential and Logarithmic Map The exponential map $\exp_x(u)$ of a tangent vector u at point x computes the point p reached by starting at point x and walking straight (*i.e.*, along a geodesic) with initial direction u for a distance $\|u\|$ (inset, *left*). Concretely, this can be evaluated by laying out the relevant sequence of triangles in the plane and drawing a straight line (inset, *right*). Note that for any oriented edge ij we have $\exp_i(e_{ij}) = j$. Conversely, the logarithmic map $\log_x(p)$ of a given point $p \in M$ taken at point x gives the smallest tangent vector u at x such that $\exp_x(u) = p$. Hence, for any point p and veretx i, we have that $\exp_x(\log_x(p)) = p$. However, it is not necessarily the case that for any tangent vector v we have $\log_x(\exp_x(v)) = v$, since there may be a shorter path leading to the same destination. In particular, $\log_i(j)$ may not always yield the edge vector e_{ij}.

2.3.3 Retriangulation



Intrinsic Edge Flip Consider an edge ij contained in triangles ijk, jil. An edge flip replaces ij with the opposite diagonal kl. An edge ij is *flippable* if and only if

(i) $\deg i, \deg j \geq 2$ and

(ii) triangles ijk, jil form a convex quadrilateral,

i.e., if both $\theta_i^{jk} + \theta_i^{lj}$ and $\theta_j^{ki} + \theta_j^{il}$ are less than π [Sharp & Crane 2020b, §3.1.3]. Note that these conditions are considerably easier to check than in the extrinsic case [Liu et al. 2020, Appendix C].

Intrinsic Delaunay Triangulation A triangulation is *intrinsic Delaunay* if it satisfies the angle sum condition $\theta_k^{ij} + \theta_l^{ji} < \pi$ at all interior edges $ij \in E$. Such triangulations extend many useful properties of 2D Delaunay triangulations to surface meshes—[Sharp et al. 2021, §4.1.1] gives a detailed list. A triangulation can be made intrinsic Delaunay via a simple greedy algorithm: flip non-Delaunay edges until none remain [Bobenko & Springborn 2007].

2.4 Static Intrinsic Triangulations

2.4.1 Mapping & Correspondence

If one starts with an extrinsic triangle mesh and performs a sequence of intrinsic edge flips, one obtains an intrinsic triangulation which preserves the original geometry exactly—it is isometric to the extrinsic mesh. In practice, it is often essential to be able to evaluate this isometry explicitly, *i.e.* to map points back and forth between the original extrinsic triangulation and the new intrinsic triangulation. We call this mapping a *correspondence* between triangulations. Using the fact that the mapping is an isometry, one can build efficient data structures for evaluating the correspondence, using integer coordinates [Gillespie et al. 2021a] or signpost vectors [Sharp et al. 2019].

Chapter 3

Surface Simplification

Discarding is not the point; what matters is keeping those things that bring you joy. If you discard everything until you have nothing left but an empty house, I don't think you'll be happy living there. Our goal in tidying should be to create a living environment filled with the things we love.
Marie Kondō

The first instance of dynamic surfaces that we will consider is surface simplification. Whereas past simplification methods focus on visual appearance, our goal is to solve equations on the surface. Hence, rather than approximate the extrinsic geometry, we construct a coarse *intrinsic triangulation* which approximates the input domain. In the spirit of the *quadric error metric (QEM)* of Garland & Heckbert [1997], we perform greedy decimation while agglomerating global information about approximation error. In lieu of extrinsic quadrics, however, we store intrinsic tangent vectors that track how far curvature "drifts" as the surface evolves during simplification. This process also yields a bijective map between the fine and coarse mesh, and prolongation operators for both scalar- and vector-valued data. Moreover, we obtain hard guarantees on element quality via intrinsic retriangulation—a feature unique to the intrinsic setting. The overall payoff is a "black box" approach to geometry processing, which decouples mesh resolution from the size of matrices used to solve equations.

3.1 Intrinsic Vertex Removal

Extrinsic simplification methods reduce vertex count by making local changes to the mesh connectivity [Garland & Heckbert 1997; Hoppe 1996; Schroeder et al. 1992]. We extend local simplification to the intrinsic setting, using vertex removal as our atomic simplification operation. Intrinsic simplification provides strictly more possibilities than its extrinsic counterpart, since any extrinsic operation can be represented intrinsically.

Our method removes a vertex i in three steps:

1. Intrinsically flatten i (Section 3.1.1).
2. Remove i from the triangulation (Section 3.1.2).
3. Flip to an intrinsic Delaunay triangulation (Section 2.3.3).

The vertex removal step extends the scheme of Gillespie et al. [2021a, §3.5] to handle boundary vertices as well. Note that all changes to the geometry occur in the first step, redistributing the curvature at i to neighboring vertices j. The second step merely retriangulates a flat region, and the third step performs only intrinsic edge flips. Hence, when measuring distortion in Section 3.3 will need only consider the first (flattening) step to prioritize vertex removals. Maintaining a Delaunay triangulation at each step helps ensure numerical robustness throughout simplification.

Special cases To remove an ear vertex i, it is tempting to remove the triangle ijk containing i. However, doing so leaves points on the fine mesh that do not map anywhere on the coarse mesh. Instead, we turn any ear into a regular boundary vertex by flipping the opposite edge jk.

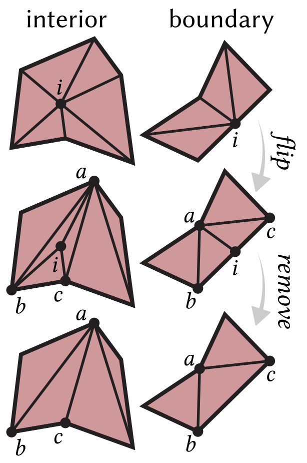
We cannot remove vertices i incident on a boundary self-edge, since every boundary loop must contain at least one vertex. Likewise, vertices i of self-faces (*i.e.*, triangles with only a single distinct vertex) can cause trouble for flipping, and are skipped.

3.1.1 Vertex Flattening

We first eliminate all curvature at vertex i. For this operation to remain local and valid we must bijectively flatten the one-ring of vertex i, while keeping edge lengths along the boundary of this region fixed. Extrinsic flattening schemes can fix boundary vertices, but it is unclear how to construct the least-distorting boundary polygon with prescribed lengths. In contrast, the CETM algorithm of Springborn et al. [2008] supports edge length constraints, and operates directly on an intrinsic triangulation. More details on CETM can be found in Chapter 4.

For some vertices, this procedure may fail to find a valid parameterization. In this case, we simply skip removing this vertex and move on to remove another vertex instead.

3.1.2 Flat Vertex Removal



To remove a flattened vertex i, we flip it to a degree-3 vertex and replace the three triangles iab, ibc, ica incident on i with the single triangle abc (inset, left). Since the vertex neighborhood is already flat, these operations preserve the geometry. Gillespie et al. [2021a, Appendix D.1] show that iteratively flipping any remaining flippable edge ij incident on i will yield a degree-3 vertex, so long as the neighborhood remains simplicial. Hence, at each step we first flip any self-edges ($i = j$); if there are none, we flip the edge ij with largest angle sum $\theta_k^{ij} + \theta_l^{ji}$ (since only convex triangle pairs can be flipped). In the rare case where $\deg i > 3$ and no flippable edges remain, we skip this vertex removal and revert the mesh to its previous state. If i is a boundary vertex, we again perform edge flips until $\deg i = 3$ and replace the two resulting triangles iab, ica with the single triangle abc (inset, right). Here again the geometry is unchanged, since i has no geodesic curvature. When i is an ear vertex we need only flip the opposite edge to give i degree-3, while for regular boundary vertices we use the same procedure as for interior vertices.

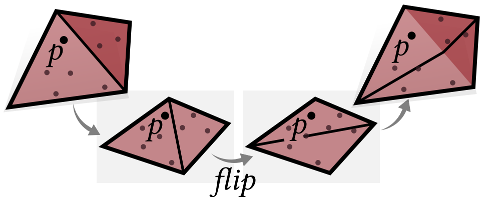
After removal, we must also update the angular coordinates φ_{jk} and corresponding edge vectors e_{jk} for each edge jk with endpoints adjacent to i (Section 2.3.2). We then flip the mesh to an intrinsic Delaunay triangulation, *à la* Section 2.3.3.

3.2 Correspondence Tracking

3.2.1 Mapping Points

To map any point p on the fine mesh to a point $\tilde{p}$ on the coarse mesh, we track its barycentric coordinates through local coarsening operations (namely: edge flips, vertex flattenings, and vertex removals). This map is trivially bijective, since at each step we simply re-write the given barycentric coordinates with respect to a different triangulation of the same planar region. The only way to violate bijectivity would be to perform a non-bijective vertex flattening—which we explicitly forbid. To evaluate this map on demand, one can record the list of local operations, and "re-play" these operations for each new query point, as done by Liu et al. [2020].

1. *Edge flips*: To track a point p through an intrinsic edge flip, we unfold the two adjacent triangles into the plane using the formulas provided by Sharp et al. [2021, Section 2.3.7], and compute the barycentric coordinates of p in the new triangle.

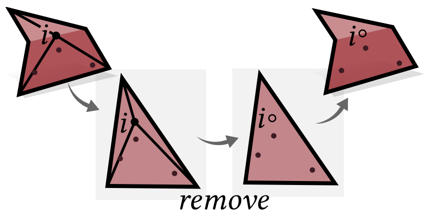


2. *Vertex flattening*: We must also compute new barycentric coordinates $\tilde{b}$ after each vertex flattening (Section 3.1.1). Here we use the projective interpolation scheme of Springborn et al. [2008, §3.4]. Since our parameterization is a discrete conformal equivalence, this scheme defines a continuous (C^0) bijective map. Let b_i, b_j, b_k be barycentric coordinates for a point in face ijk, and let u_i be the scale factor at i. Then

$$(\tilde{b}_i, \tilde{b}_j, \tilde{b}_k) = \frac{(e^{u_i} b_i, b_j, b_k)}{e^{u_i} b_i + b_j + b_k}, \tag{3.1}$$

where the denominator ensures our updated values still sum to 1.

3. *Vertex removal*: Once vertex i is flattened and flipped to degree three, its neighborhood can be laid out in the plane without distortion. Here we apply standard formulas to compute barycentric coordinates for vertex i in the new triangle, along with coordinates for any points located in the three removed triangles.



3.2.2 Mapping Edges

In addition to mapping points between the fine and coarse meshes, we can also map an edge on one mesh to the corresponding polygonal curve on the other. The key is to determine how each local operation modifies a polygonal curve. In our case, edge flips and vertex insertion/removal may add vertices to the polygonal curve, but leave its geometry the same, while vertex flattening distortions the curve geometry but leaves its combinatorics the same. Mapping edges allows us to draw the intrinsic triangulation sitting atop the extrinsic mesh, and to compute the common subdivision of the two triangulations.

3.3 Measuring Distortion

To prioritize vertex removals, we must quantify the cost of removing a vertex. Standard extrinsic metrics, such as QEM, are not appropriate: even if they could somehow be evaluated intrinsically, they would attempt to preserve irrelevant aspects of the geometry. Our method is however inspired by the remarkable effectiveness of greedy local error accumulation in QEM. Likewise, metrics that focus on finite element equality (*à la* [Shewchuk 2002]) are not appropriate, since the triangulation used to encode the intrinsic geometry is transient and subject to change.

Our ICE metric is instead based on two intrinsic and triangulation-independent concepts: *optimal transport* [Santambrogio 2015], and the *Karcher mean* [Karcher 2014]. Optimal transport helps quantify the effort of redistributing mass, providing the local cost for our metric. Karcher means encode the center of mass of all fine vertices contributing to a coarse vertex i, providing a way of accumulating information. These two pieces fit together in a natural way: after a single vertex removal, the mass-weighted norm of all error vectors t_i encoding Karcher means is exactly equal to the optimal transport cost. Hence, after many vertex removals this norm approximates the cost of transporting the initial fine mass distribution to the coarsened vertices. Vertex removals that keep cost small should hence be prioritized, since they better preserve the initial mass distribution. Just as in QEM, this information is captured by a fixed-size representation (masses and tangent vectors at each vertex) that is easily agglomerated during coarsening. For clarity of exposition we first define error metrics in 2D, before generalizing to surfaces and incorporating data like curvature or other attributes.

3.3.1 Flat Error Metric

Consider a mass distribution $m : V \to \mathbb{R}_{\geq 0}$ at mesh vertices, representing any nonnegative user-defined quantity (signed quantities will be addressed in Section 3.3.2). Suppose we remove vertex i, redistributing its mass m_i to its immediate neighbors j. In particular, let $\alpha_{ij} \in [0, 1]$ be the fraction of m_i sent to vertex j (hence $\sum_{j\sim i} \alpha_{ij} = 1$), so that the new mass at j is

$$\tilde{m}_j = m_j + \alpha_{ij} m_i. \tag{3.2}$$

To measure how mass is transported across the surface, we need to track not only the mass distribution, but also where mass came from. Hence, at each vertex i we store an *error vector* t_i (initially set to zero) pointing to the center of mass c_i of all vertices that contributed to the current value of m_i. Explicitly, after removing i, the center of mass at vertex j is

$$\tilde{c}_j = \frac{\alpha_{ij} m_i x_i + m_j x_j}{\alpha_{ij} m_i + m_j}, \tag{3.3}$$

where $x_i \in \mathbb{R}^2$ denotes the location of vertex i. The vector pointing from x_j to $\tilde{c}_j$ is thus

$$\tilde{t}_j = \tilde{c}_j - x_j = \frac{\alpha_{ij} m_i e_{ji}}{\alpha_{ij} m_i + m_j}, \tag{3.4}$$

where $e_{ji} = x_i - x_j$ is the vector along edge ji. The total cost of removing i can then be measured by summing up the mass-weighted norms of these vectors. Noting that $\|e_{ji}\| = \ell_{ij}$, we get a cost

$$C_i = \sum_{j \sim i} \tilde{m}_j \|\tilde{t}_j\| = \sum_{j \sim i} \alpha_{ij} m_i \ell_{ij}. \tag{3.5}$$

This cost also coincides with the so-called *1-Wasserstein distance* between the old and new mass distribution [Santambrogio 2015, Chapter 5]. Intuitively, this distance measures the total "effort" of moving mass from i to neighbors j, penalizing not only the amount of mass moved, but also the distance traveled.

Rather than assign a cost to each vertex removal in isolation, we can accumulate information about how mass has been redistributed across all prior removals. At each step, we update the mass distribution via Equation (3.2), but also update vectors encoding the centers of mass via

$$\tilde{t}_j = \frac{\alpha_{ij} m_i (t_i + e_{ji}) + m_j t_j}{\alpha_{ij} m_i + m_j}. \tag{3.6}$$

In other words, we re-express t_i relative to x_j by adding the edge vector e_{ji}, then take the mass-weighted average of the old error vector t_j with this new vector. The overall cost is still evaluated via Equation (3.5), but now approximates the effort of moving the initial mass distribution to the current one—rather than just penalizing the most recent change. This cost is only approximate since the 1-Wasserstein distance to the center of mass is not in general equal to the distance to the original fine distribution—but it is usually quite close. Thus, our error metric favors decimation sequences which keep each coarse vertex close to the center of all fine vertices that contribute to its mass.

3.3.2 Intrinsic Curvature Error Metric

Due to the Gauss-Bonnet theorem, flattening a vertex i conservatively redistributes curvature to neighboring vertices j, making curvature a natural "mass" distribution to guide simplification. A challenge here is that the old and new curvatures K and $\tilde{K}$ are not in general positive quantities. One possibility might be to use a transport cost for signed measures such as [Mainini 2012], but doing so would require us to solve a small optimal transport problem for each vertex removal. We instead adopt a cheap alternative. In particular, we define convex weights

$$\alpha_{ij} := \frac{|\tilde{K}_j - K_j|}{\sum_{l \sim i} |\tilde{K}_l - K_l|}. \tag{3.7}$$

For boundary vertices we use the same formula, but replace Gaussian curvature K with geodesic curvature κ. If vertex i is already flat prior to removal, then there is no change in curvature and we simply distribute mass equally to all neighbors. We then split the initial fine curvature function K (or κ) into two positive mass functions $K_i^+ := \max(K_i, 0)$ and $K_i^- := -\min(K_i, 0)$. Each of these quantities is tracked throughout simplification like m_i above, using two separate vectors t_i^+ and t_i^- (respectively), and weights α from Equation (3.7):

$$\tilde{t}_j^\pm = \frac{\alpha_{ij} m_i (R_{ij} t_i^\pm + e_{ji}) + m_j t_j^\pm}{\alpha_{ij} m_i + m_j}. \tag{3.8}$$

The parallel transport term R_{ij} to account for the surface curvature. The overall error, which defines the ICE metric, is then the sum of the errors in the two curvature functions (*à la* Equation (3.5)). Note that if a vertex i cannot be flattened or removed, we assign it an infinite cost (which may later get updated to a finite value when its neighbors are removed.

3.4 Results

3.4.1 Comparison with Extrinsic Methods

The flexibility gained by working in the larger space of intrinsic triangulations leads to lower geometric distortion than extrinsic meshes exhibit on meshes of equivalent size. In Figure 3.1 we coarsen a 28k bunny mesh down to 200 vertices with both the method of Liu et al. [2021] and our method. Even on this highly regular geometry we observe a modest reduction of both area distortion and anisotropic distortion. For more difficult triangulations, or surfaces with lower intrinsic curvature, we observe more significant gains.

As an extreme case, Figure 3.2 coarsens a developable surface from [Verhoeven et al. 2022] via both QEM and ICE. Since coarse extrinsic edges are shortest paths in $\mathbb{R}^n$, they underestimate intrinsic distances (hence areas); in contrast, intrinsic edges are essentially embedded in the original surface, providing better approximation of the original geometry.

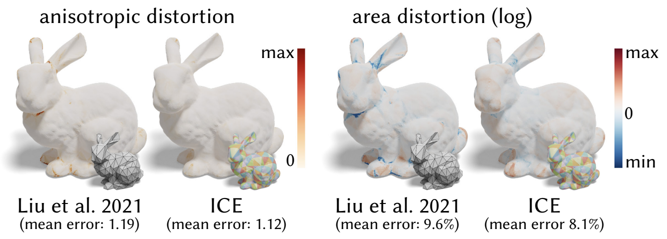


Figure 3.1: Even on an extremely nice triangulation of a highly regular surface we see a reduction in distortion relative to past methods—owing to the much larger space of intrinsic triangulations.

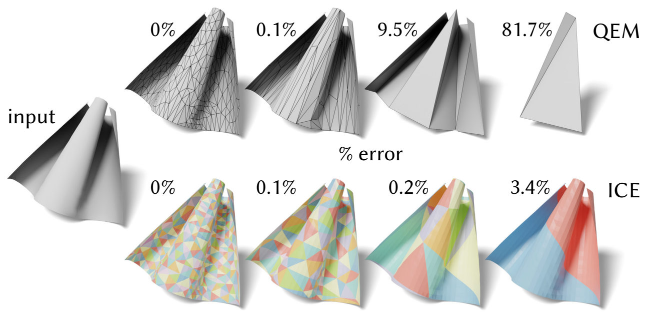


Figure 3.2: On surfaces with small extrinsic curvature, we achieve dramatically lower error in surface area compared to extrinsic methods like QEM.

3.4.2 Geometric Algorithms

Partial Differential Equations Better domain approximation in turn improves the quality of solutions computed on coarse meshes. For example, in Figure 3.3 we coarsen a cloth simulation mesh down to 500 vertices with an extrinsic method ([Liu et al. 2021] using QEM simplification) and our intrinsic method. We then solve a Poisson problem on the coarse meshes and apply prolongation, yielding more accurate results in the intrinsic case.

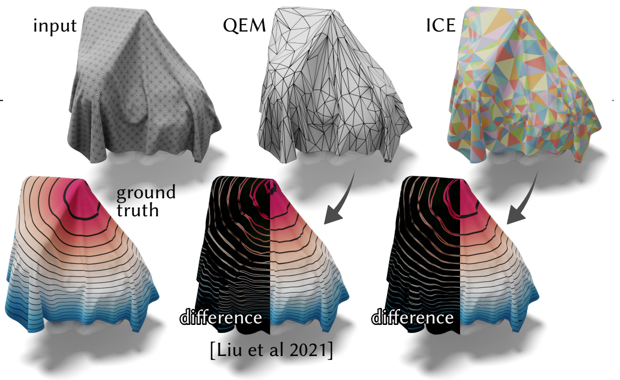


Figure 3.3: For the same vertex budget as extrinsic methods like QEM, ICE provides more accurate solutions for basic problems like solving a Poisson equation—seen here via smoother isolines that better approximate the ground truth.

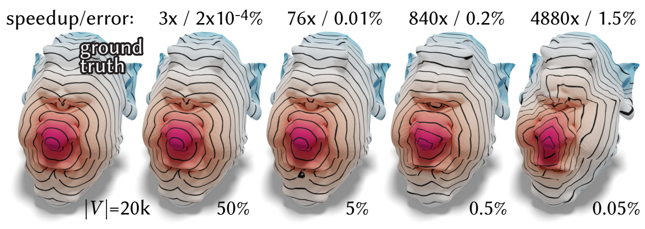


Figure 3.4: Intrinsic coarsening offers an attractive approach to approximating single-source geodesic distance, here providing a three orders of magnitude speedup for a fraction of a percent relative error.

Single-Source Geodesic Distance Geodesic distance is an intrinsic quantity, making it a natural fit for intrinsic coarsening. In Figure 3.5 we compare ICE to the extrinsic method of Lee et al. [1998] by measuring the difference between the exact distance on the fine input, and prolongated distances from the coarse meshes (both computed via [Mitchell et al. 1987]); here ICE achieves a roughly 4x reduction in relative error. Figure 3.4 illustrates the speed-accuracy trade off of using ICE, here reducing cost by three orders of magnitude while introducing only $\sim 1\%$ relative approximation error.

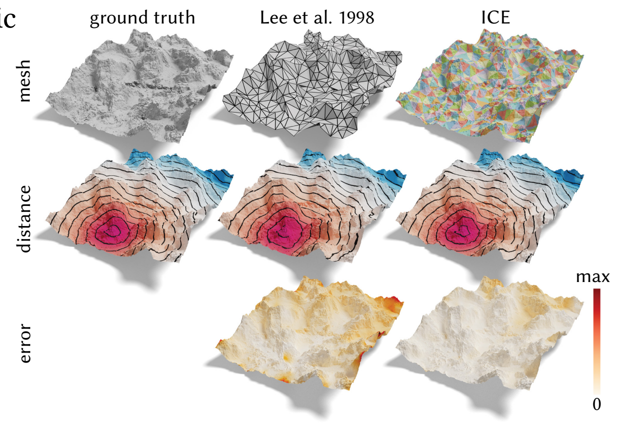


Figure 3.5: As geodesic distance is an intrinsic quantity, it is more accurately approximated via intrinsic coarsening—here providing a 4x reduction in relative error.

All-Pairs Geodesic Distance The benefits of an accurate intrinsic approximation become even more pronounced when approximating the dense matrix $D \in R^{|V|\times|V|}$ of all pairs of geodesic distances—a shape descriptor often used in correspondence and learning methods [Shamai & Kimmel 2017]. We can compute a low-rank approximation of D via

$$\widehat{D} := P\widetilde{D}P^\top,$$

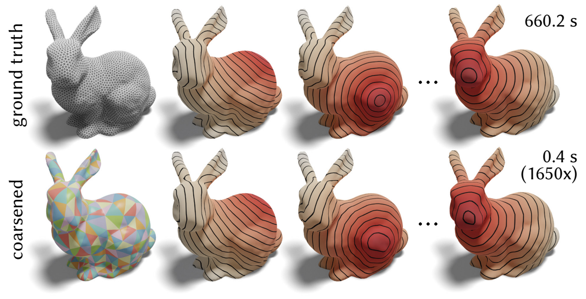


Figure 3.6: For a mesh with 6k vertices we obtain an all-pairs geodesic distance matrix 1650x faster, while incurring only 1.4% relative error.

where $\widetilde{D}$ is the coarse all-pairs matrix (computed again via [Mitchell et al. 1987]). See for instance Figure 3.6—here again we achieve several orders of magnitude speedup, with only 1.4% relative error.

Chapter 4

Surface Parameterization

He handed it to Harrow, who gently unfolded it in the way that only a bone magician could and in the way that always made Gideon's jaw hurt. She turned it into a long ribbon of enamel, an orange with the skin taken off and flattened, a three-dimensional object turned two-dimensional.
Tamsyn Muir, *Gideon the Ninth*

We now turn our attention to another instance of time-evolving surfaces: finding surface parameterizations. In particular, we describe a numerical method which computes maps that are locally injective and discretely conformal in an exact sense. Unlike previous methods for discrete conformal parameterization, the method is guaranteed to work for any manifold triangle mesh, with no restrictions on triangulation quality or cone singularities. In particular we consider maps from surfaces of any genus (with or without boundary) to the plane, and globally bijective maps from genus zero surfaces to the sphere. Recent theoretical developments have shown that each task can be formulated as a convex problem where the triangulation is allowed to change—we complete the picture by introducing the machinery needed to actually construct a discrete conformal map.

In the smooth setting, existence of conformal maps is guaranteed by the *uniformization theorem* [Abikoff 1981]. Very recently, Gu et al. [2018a,b] and Springborn [2019] established an analogous *discrete uniformization theorem* for triangle meshes. However, these theoretical results fall short of providing practical algorithms, since they do not describe how to construct the mapping between the input and target domain. We present the first end-to-end algorithm for computing and evaluating this map—in particular, we provide:

- a combinatorial data structure for correspondences between triangulations (Section 4.1),
- a scheme for evaluating discrete conformal maps based on the *light cone* (Section 4.2), and
- critical details needed to implement discrete uniformization including a careful treatment of numerics and boundary conditions (Section 4.3), and subtleties of the spherical case.

Our optimization procedure is a simple modification of the *CETM algorithm* (from Springborn et al. [2008], *Conformal Equivalence of Triangle Meshes*): we minimize the same energy, but evaluate it on a triangulation that changes according to the current scale factors. However, since

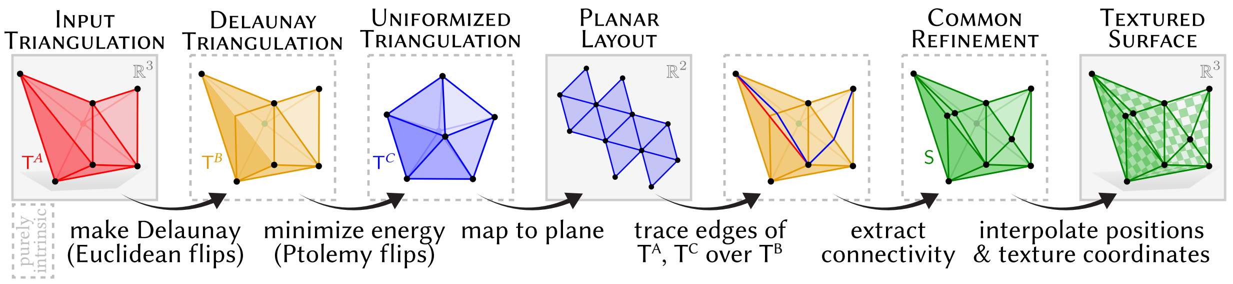


Figure 4.1: Steps of our algorithm. Throughout we color the input mesh T^A red, its intrinsic Delaunay triangulation T^B yellow, the uniformized triangulation T^C blue, and the common refinement S of all three green. (Note: triangulations in dashed boxes are purely intrinsic and never actually embedded in $\mathbb{R}^n$.)

the triangulation may now change, this procedure does not yield an explicit parameterization of the input. To improve the quality of the map, we also need to flip the input to an *intrinsic Delaunay triangulation* before starting the optimization. The main difficulty in developing a practical algorithm is therefore tracking and evaluating the correspondence between these three triangulations—Figure 4.1 gives an overview of the whole process. Importantly, even though the Euclidean geometry of our polyhedron changes during optimization, it can always be seen as different reflections of the same underlying hyperbolic polyhedron, which greatly simplifies the problem of correspondence.

4.1 Correspondence Data

We begin by describing our data structure for encoding the correspondence between different triangulations of the same polyhedron. In particular, we introduce an implicit, integer-based encoding that is easily updated via local formulas during each edge flip. An explicit geometric correspondence is later extracted from this information once all flips have been performed (*e.g.*, after uniformization)—see Section 4.2. Since this encoding uses only integer data, it avoids robustness issues that can arise with floating-point representations.

To encode the correspondence between triangulations T_1, T_2 sharing vertex set V, we store

(1) *normal coordinates*, which count how many times T_1 crosses each edge of T_2 (Section 4.1.1), and

(2) *roundabouts*, which give the circular ordering of edges from both T_1 and T_2 around each vertex (Section 4.1.2).

Normal coordinates enable us to later trace geodesic segments from each vertex i to all neighboring vertices j in T_1, yielding curves along T_2 (Section 4.2.1). Roundabouts provide the correspondence between these traced segments and logical edges of T_1. This latter data is needed because the two endpoints i, j of a traced segment may not uniquely determine an edge.

For our flattening procedure we use this scheme to track the correspondence both between T^A and T^B (as Euclidean polyhedra), and between T^B and T^C (as hyperbolic polyhedra). In the rest of this section we provide more detail about the data that we store, and in the next section we give an overview of how the correspondence can be evaluated using this data.

4.1.1 Normal Coordinates

Normal coordinates count the number of times a collection of curves cross each edge of a fixed triangulation (Figure 4.2). Our use of normal coordinates deviates from the standard treatment in two ways. First, rather than closed topological curves, we consider open geodesic segments that terminate at vertices. Second, we always assume that our normal coordinates encode the edges of another triangulation of the same vertex set. These assumptions enable us to develop a novel edge flip formula, given in Section 4.1.1. In particular, for each edge ij of T_2, we store the number of times $n_{ij} \in \mathbb{Z}_{\geq 0}$ that any edge of T_1 crosses ij transversely (Figure 4.2, *left*). Hence, for edges ij shared by both T_1 and T_2 we have $n_{ij} = 0$. From these numbers we can determine how many edges in T_1 emanate from corner k of a triangle ijk in T_2 (excluding those along edges of T_2) :

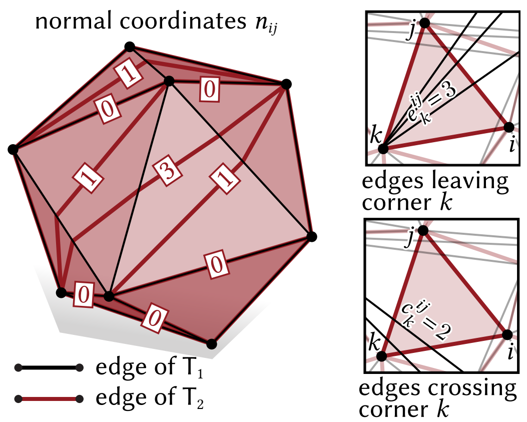


Figure 4.2: *Left:* normal coordinates n_{ij} count the number of times each edge ij in a triangulation T_2 crosses any edge of another triangulation T_1 transversely. *Right:* these coordinates can be used to determine other quantities, such as how many edges of T_1 cross or leave a corner of a triangle from T_2.

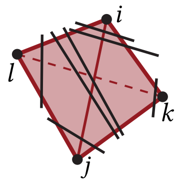
$$e_k^{ij} = \max\left(0, n_{ij} - n_{jk} - n_{ki}\right). \tag{4.1}$$

Likewise, the number of edges in T_1 that cross corner k of ijk is

$$c_k^{ij} = \tfrac{1}{2}\left(\max\left(0, n_{jk} + n_{ki} - n_{ij}\right) - e_i^{jk} - e_j^{ki}\right). \tag{4.2}$$

See Figure 4.2, *right* for examples.

Normal Coordinate Edge Flip Consider two triangles ijk, jil from T_2. In the simple case where no edge from T_1 terminates in a corner of either triangle (see inset), there is an edge flip update that resembles the Ptolemy relation [Mosher 1988]; [Thurston & Yuan 2012, Equation 1]:



$$n_{kl} = \max(n_{ki} + n_{lj}, n_{jk} + n_{li}) - n_{ij}. \tag{4.3}$$

In the general case, we must derive a more complicated formula:

$$n_{kl} = \max\left(0,\ c_l^{ji} + c_k^{ij} + \tfrac{1}{2}\left|c_j^{il} - c_j^{ki}\right| + \tfrac{1}{2}\left|c_i^{lj} - c_i^{jk}\right| - \tfrac{1}{2}e_l^{ji} - \tfrac{1}{2}e_k^{ij} + e_i^{lj} + e_i^{jk} + e_j^{il} + e_j^{ki} + \delta_{n_{ij}}\right). \tag{4.4}$$

Here δ_x is the *Kronecker delta*, equal to 1 for $x = 0$ and 0 otherwise.

4.1.2 Roundabouts

Although normal coordinates completely describe a triangulation sitting on top of T_2, they do not tell us how the edges of this triangulation correspond to the edges of T_1 since, as noted above, two endpoints may not uniquely identify an edge. We therefore augment our normal coordinates with what we call *roundabouts*, in analogy with roundabouts or traffic circles found on roadways. At each vertex $i \in \mathsf{V}$, these roundabouts describe how the outgoing halfedges of the two triangulations are interleaved.

More explicitly, for each halfedge $\overrightarrow{ij} \in \mathsf{H}_2$, the roundabout gives the first halfedge from T_1 following $\overrightarrow{ij}$, encoded as an index $r_{\overrightarrow{ij}} \in \mathbb{Z}_{\geq 0}$ (Figure 4.3). These indices start at zero, and enumerate the halfedges from T_1 in counter-clockwise order, starting at some arbitrary but fixed halfedge. Note that if a halfedge from T_2 coincides with a halfedge from T_1, the roundabout points to this halfedge, as indicated by self-arrows.

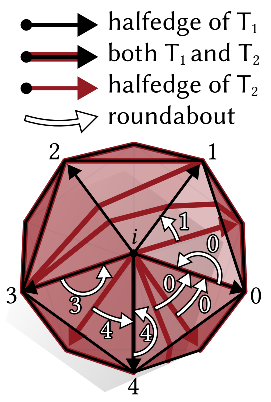
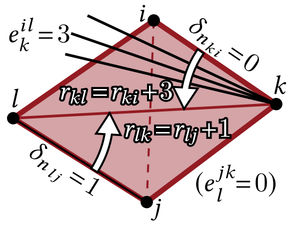


Figure 4.3: For each halfedge of T_2, the roundabout gives the next halfedge of T_1.

Roundabout Edge Flip Using per-vertex indices (instead of a map from H_2 to H_1) reduces the edge flip update to integer arithmetic. In particular, to update roundabouts after flipping an edge ij with opposite vertices k, l, we first update the normal coordinates as described in Section 4.1.1. We then have

$$\begin{aligned} r_{\overrightarrow{kl}} &= \operatorname{mod}(r_{\overrightarrow{ki}} + e^{il}_{k} + \delta_{n_{ki}}, \deg_1(k)), \\ r_{\overrightarrow{lk}} &= \operatorname{mod}(r_{\overrightarrow{lj}} + e^{jk}_{l} + \delta_{n_{lj}}, \deg_1(l)), \end{aligned} \tag{4.5}$$

where $\deg_1(i)$ is the degree of vertex i in the triangulation T_1. In other words, to find the first outgoing halfedge of T_1 following $\overrightarrow{kl} \in \mathsf{H}_2$, we start at $\overrightarrow{ki}$ and add the number of edges e^{il}_{k} of T_1 that emanate from corner k of triangle kil. Also, if $\overrightarrow{ki}$ is coincident with a halfedge from T_1, we add 1 to advance past this halfedge. The mod operation accounts for wraparound. See inset for an example. This update resembles a combinatorial version of the signpost update from Sharp et al. [2019, p. 3.2.1]: integer indices $r_{\overrightarrow{ij}}$ play the role of real-valued directions; the integer counts e^{jk}_{i} play the role of real-valued angles.



4.2 Mapping

Following uniformization (Section 4.3), we have three triangulations: the input T^A with vertex positions f, its intrinsic Delaunay triangulation T^B, and the flattened mesh T^C with texture coordinates z (Figure 4.1). For most tasks (*e.g.*, texture mapping or remeshing), we will need an explicit map between T^A and T^C, which we now construct. Using the correspondence data from Section 4.1 we first trace out geodesics to identify the points where edges of T^A and T^C intersect edges of T^B (Section 4.2.1). We use these points to construct the *common refinement* S, *i.e.*, a polygonal mesh that encompasses all three triangulations (Section 4.2.3). Finally, we interpolate functions f and z across S (Section 4.2.4), obtaining an extrinsic polygon mesh with vertex positions $f_i \in \mathbb{R}^3$ and texture coordinates z^{jk}_{i} at each triangle corner.

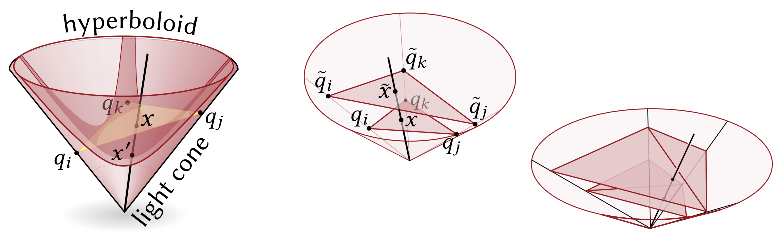


Figure 4.4: By drawing triangles in the light cone *(left)*, the map between surfaces can be found by drawing a straight line through the origin *(center)*, which also works for two different triangulations *(right)*.

Layout in the Light Cone As discussed in [Springborn et al. 2008, Section 3.4], conformally equivalent edge lengths naturally induce a *piecewise projective map*. However, when the triangulation is allowed to change, constructing this map becomes more difficult. A useful perspective, different from previous work [Bobenko et al. 2015; Springborn 2019; Sun et al. 2015], is to consider *chordal triangles* in the light cone—leading to simple interpolation formulas in homogeneous coordinates (*e.g.*, Equation (4.7)).

4.2.1 Tracing Edges

For the moment, consider just two triangulations $\mathsf{T}_1, \mathsf{T}_2$. We use the normal coordinates $n : \mathsf{E}_2 \to \mathbb{Z}_{\geq 0}$ to trace out the sequence of edges in T_2 crossed by each edge of T_1 (Section 4.2.1). The roundabouts $r : \mathsf{H}_2 \to \mathbb{Z}_{\geq 0}$ uniquely identify each traced sequence with the appropriate element of E_1. To get the curve geometry, we lay out a triangle strip in the Euclidean or hyperbolic plane, and draw a straight line between endpoints (Section 4.2.2). The final curve is encoded by 1D barycentric coordinates $s, t \in [0, 1]$ on each intersected edge.

Topological Tracing To identify the sequence of edges in T_2 crossed by some edge in T_1, we start at one crossing and repeatedly identify the next edge crossed until the edge of T_1 terminates at a vertex. We can determine the next edge crossed purely from the stored normal coordinates, by considering the three cases illustrated in Figure 4.5.

Note that the tracing procedure gives us each edge from T_1 as a sequence of edge crossings on T_2. To express the edges from T_2 as sequences of T_1 edge crossings, we allocate an array of size n_{ij} for each edge $ij \in \mathsf{E}_2$. Each time a traced edge $ab \in \mathsf{T}_1$ crosses ij, we store a reference to

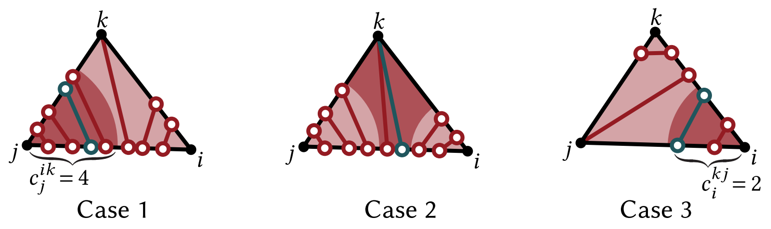
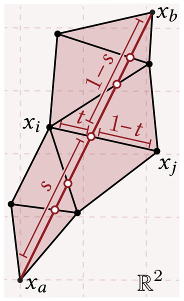


Figure 4.5: A curve entering triangle jik along edge ij can proceed in 3 ways: the left-most c_j^{ik} crossings go left (*left*), the rightmost c_i^{kj} crossings go right (*right*), and the remaining crossings terminate at vertex k (*center*).

ab in entry p of the array (using roundabouts to get the edge index).

4.2.2 Recovering Geodesics

To get the geometry of each traced edge $ab \in \mathsf{E}_2$, we use the crossing sequences computed in Section 4.2.1 and the edge lengths ℓ to incrementally lay out a triangle strip in the (Euclidean or hyperbolic) plane. We then intersect each interior edge ij of this strip with the line from a to b—by construction, this line will be contained entirely inside the strip. In particular, if $x_i \in \mathbb{R}^2$ are the vertices of a Euclidean triangle strip, we can solve the equation



$$(1-s)x_a + sx_b = (1-t)x_i + tx_j \tag{4.6}$$

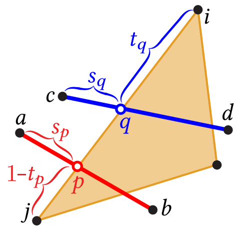
for the barycentric coordinates $s, t \in [0, 1]$ of the intersection point. The hyperbolic case is conceptually the same except that we work in the hyperboloid model, and and also compute a scale factor u at each intersection point.

4.2.3 Common Refinement

Tracing out the edges, allows us to construct the common refinement S of T^A, T^B, and T^C. To determine the connectivity of S we slice up each triangle $ijk \in \mathsf{F}^B$ independently, via a strategy similar to Sharp et al. [2019, Section 3.4]. To avoid computing segment-segment intersections directly (which is not numerically robust), we devise a strategy that takes advantage of combinatorial information. Floating-point values serve only to determine the ordering of intersection points along edges—and since neighboring triangles have identical barycentric coordinates along their shared edge, we always obtain a consistent tessellation.

4.2.4 Interpolation

The vertex coordinates f_i and texture coordinates z_i^{jk} define piecewise functions over the faces of T^A and T^C, *resp.*; we now sample these functions onto S. To do so, we will also need the scale factors u obtained while tracing hyperbolic geodesics. We again process each triangle $ijk \in \mathsf{T}^B$ independently. First, we interpolate data onto each edge ij of the triangle. For each edge point p along an edge $ab \in \mathsf{E}^A$, let s_p, t_p be the barycentric coordinates along ab and ij, *resp.*. Then $f_p = (1-s_p)f_a + s_p f_b$. Similarly, for an edge point q along $cd \in \mathsf{E}^C$ we have homogeneous texture coordinates



$$\hat{z}_q = e^{-u_q}\left((1-s_q)(z_c, 1) + s_q(z_d, 1)\right), \tag{4.7}$$

where $(z, 1)$ indicates that a 1 has been appended to z. The scale factors e^u arise from projective rather than linear interpolation—see supplement for details. To get values of f at edge points q, and values of $\hat{z}$ at edge points p, we linearly interpolate between adjacent known values along ij. Finally, to get the values at each face point, we write the endpoints of the two incident fragments in 2D barycentric coordinates relative to ijk, and compute the intersection point in homogeneous coordinates. The resulting s, t values are then used to linearly interpolate f and $\hat{z}$ from the segment endpoints. Note that since texture coordinates are discontinuous across cuts, we store $\hat{z}$ at corners rather than vertices. The final surface can be visualized by tessellating polygons into triangles; just as in [Springborn et al. 2008, Section 3.4] we perform a homogeneous divide on texture coordinates $\hat{z}$ at each sample point (*e.g.*, each pixel).

4.3 Planar Parameterization

Here we describe our procedure for planar parameterization (Figure 4.1)—see the paper for the spherical case. Given an input mesh T^A, we first flip to an intrinsic Delaunay triangulation T^B, which preserves the Euclidean geometry and defines the discrete conformal structure. We then solve an optimization problem for scale factors u that transform T^B into a triangulation T^C with the prescribed angle defects (Section 4.3.3). After optimization, we lay T^C out in the plane (Section 4.3.5), and pull this layout back to the input mesh as described in Section 4.2.

4.3.1 Variational Formulation

The input to our discrete uniformization procedure is the intrinsic Delaunay triangulation T^B, and target angle defects $\Omega^* : \mathsf{V} \longrightarrow \mathbb{R}$ which must satisfy a discrete Gauss-Bonnet condition:

$$\tfrac{1}{2\pi} \sum_{i \in \mathsf{V}} \Omega_i^* = |\mathsf{V}| - |\mathsf{E}^B| + |\mathsf{F}^B| \tag{4.8}$$

(see Section 4.3.4 for a generalization to surfaces with boundary). Note that target defects Ω_i^* must be smaller than 2π, since the sum of angles around a vertex is always positive. Minimizing a convex energy $\mathcal{E}$ then yields scale factors u relative to T^B. Unlike CETM we flip to Delaunay whenever we need to evaluate the energy or its derivatives (see Section 4.3.2). This process is completely hidden inside a callback routine—from the perspective of the optimizer, one simply has to solve an unconstrained problem that is convex and twice continuously differentiable (C^2).

4.3.2 Energy Evaluation

To evaluate our energy for any given u, we first compute the edge lengths $\tilde{\ell}_{ij} = e^{(u_i+u_j)/2}\ell_{ij}^B$, and flip to the corresponding ideal Delaunay triangulation $\widetilde{T} = (\mathsf{V}, \widetilde{\mathsf{E}}, \widetilde{\mathsf{F}})$ via Ptolemy flips. These flips change the Euclidean geometry but preserve the discrete conformal structure. We use $\tilde{\lambda}$, $\tilde{\theta}$, and $\widetilde{\Omega}$ to denote the corresponding Penner coordinates, interior angles, and angle defects, *resp.*

Energy The discrete conformal energy is then given by

$$\mathcal{E}(u) = \sum_{i \in \mathsf{V}} (2\pi - \Omega_i^*)\, u_i - \sum_{ij \in \widetilde{\mathsf{E}}} \pi \tilde{\lambda}_{ij} + \sum_{ijk \in \widetilde{\mathsf{F}}} 2f(\tilde{\lambda}_{ij}, \tilde{\lambda}_{jk}, \tilde{\lambda}_{ki}), \tag{4.9}$$

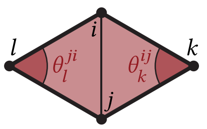
where $f(\tilde{\lambda}_{ij}, \tilde{\lambda}_{jk}, \tilde{\lambda}_{ki}) := \frac{1}{2}\left(\tilde{\theta}_i^{jk}\tilde{\lambda}_{jk} + \tilde{\theta}_j^{ki}\tilde{\lambda}_{ki} + \tilde{\theta}_k^{ij}\tilde{\lambda}_{ij}\right) + Л(\tilde{\theta}_i^{jk}) + Л(\tilde{\theta}_j^{ki}) + Л(\tilde{\theta}_k^{ij})$. Here Л denotes *Milnor's Lobachevsky function* $Л(\theta) := -\int_0^\theta \log|2\sin u|\, du$, and is related to *Clausen's integral* via $Л(\theta) = \frac{1}{2}\mathrm{Cl}_2(2\theta)$, which is implemented in standard numerical packages [Galassi et al. 1994].

Gradient At each vertex $i \in \mathsf{V}$, the gradient of the energy is

$$\partial_{u_i}\mathcal{E} = \widetilde{\Omega}_i - \Omega_i^* \tag{4.10}$$

Note, then, that any stationary point $\partial_u\mathcal{E} = 0$ achieves the desired angle defects $\widetilde{\Omega} = \Omega^*$.

Hessian The Hessian is given by the positive-semidefinite *cotan Laplacian* $L \in \mathbb{R}^{|\mathsf{V}|\times|\mathsf{V}|}$ [MacNeal 1949, Section 3.2; Crane et al. 2013, Chapter 6]. Since a Δ complex may contain more than one edge with the same endpoints, the off-diagonal entries L_{ij} and L_{ji} are obtained by summing the values $\frac{1}{2}(\cot\theta_k^{ij} + \cot\theta_l^{ji})$ over all edges $ij \in \widetilde{\mathsf{E}}$ with endpoints i and j, where k, l are the vertices opposite the edge. For each vertex $i \in \mathsf{V}$, we then have a diagonal entry $L_{ii} = -\sum_{ij\in\widetilde{\mathsf{E}}} L_{ij}$, where the sum is taken over all edges incident on i. Note that self-edges (where $i = j$) make no contribution.



4.3.3 Optimization

Since the energy $\mathcal{E}$ is convex and globally C^2, it can be minimized using any standard method for convex optimization. We use Newton's method with backtracking line search, as described in Algorithms 9.5 and 9.2 of Boyd & Vandenberghe [2004], *resp.* In particular, we use the descent direction $v \in \mathbb{R}^{|\mathsf{V}|}$ obtained by solving the linear system

$$Lv = \partial_u\mathcal{E}, \tag{4.11}$$

where $\partial_u\mathcal{E} \in \mathbb{R}^{|\mathsf{V}|}$ encodes the gradient defined in Section 4.3.2. Note that the matrix L has a one-dimensional kernel of constant vectors. We simply use the solution v that has no constant component (which corresponds to a global scaling). Although L is rank deficient, the system is solvable: Gauss-Bonnet ensures that the right-hand side sums to zero. We initialize Newton's method with $u = 0$, but since the energy is convex this choice will not affect the result (apart from a global scale).

4.3.4 Surfaces with Boundary

For a smooth surface M with boundary ∂M, the space of conformal maps to the plane is parameterized by a real-valued function along the boundary—geometrically, this function can be determined by prescribing either the scale factors u or the curvature density $\kappa\, ds$ along ∂M (see [Sawhney & Crane 2017, Section 4.2] for further discussion). We can specify such conditions by either a scale factor u_i or target exterior angle κ_i^* at each boundary vertex $i \in \partial\mathsf{V}$. To enforce these conditions, we glue together two copies of the input mesh along the boundary (as in Jin et al. [2004]), reducing the problem to the

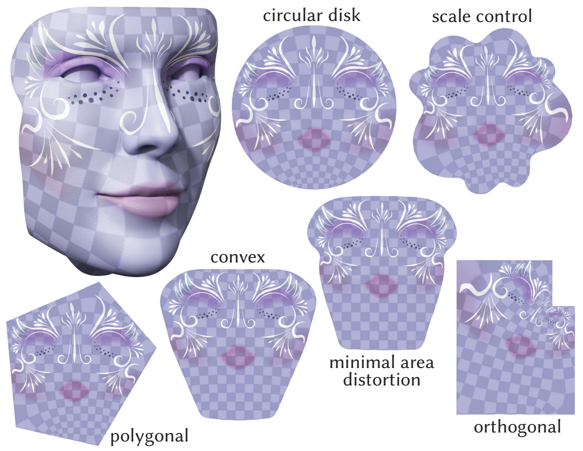
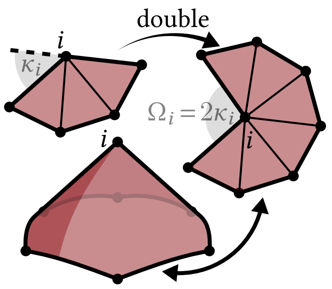


Figure 4.6: Our algorithm guarantees existence of a locally injective discrete conformal map for any prescribed boundary lengths or angles, which can be used to achieve a rich variety of behavior.

no-boundary case. Unlike CETM, we can hence always find a solution with the prescribed boundary data. Note that this construction extends Springborn [2019], which does not consider surfaces with boundary; Sun et al. [2015] describe a similar scheme in the case of prescribed boundary curvature. Maps to the circular disk are handled in a similar fashion, but using spherical uniformization.

Fixed Boundary Curvature Suppose we want our flattened domain to have an exterior angle κ_i^* at a boundary vertex i. The angle sum at i must then be equal to $\pi - \kappa_i^*$, hence on the doubled domain we prescribe an angle defect $\Omega_i^* = 2\pi - 2(\pi - \kappa^*) = 2\kappa_i^*$. Since the solution is unique, it must be symmetric across the two copies of the original mesh. Hence, if we cut the uniformized surface along the original boundary curve, each half will exhibit the desired angles κ^*. The only requirement is that the angle defects and exterior angles satisfy a Gauss-Bonnet condition $\sum_{i \in \mathsf{V}} \Omega_i^* + \sum_{i \in \partial \mathsf{V}} \kappa_i^* = |\mathsf{V}| - |\mathsf{E}| + |\mathsf{F}|$. In Figure 4.6 we assign target angles that yield convex ($\kappa_i^* > 0$), orthogonal ($\kappa_i^* \in \frac{\pi}{2}\mathbb{Z}$), or polygonal boundaries ($\kappa_i^* = 0$ almost everywhere).



Fixed Boundary Scale Factors To prescribe boundary scale factors, we fix the values u_i at vertices i of the doubled domain corresponding to the original boundary. For instance, setting $u_i = 0$ at all boundary vertices yields minimal area distortion [Chebyshev 1899, p. 242] in the sense that it minimizes the variation in scale factors [Springborn et al. 2008, Appendix E]—see Figure 4.6. Fixing these values restricts the convex energy $\mathcal{E}$ to a linear subspace; hence we are still solving a convex problem. To compute the descent direction, we now solve the same system (Equation (4.11)), except that we set zero Dirichlet boundary conditions at the boundary vertices, since we do not want these values to change. The minimizer will exhibit the target angle defects at interior vertices, since the gradient still only vanishes when $\widetilde{\Omega} = \Omega^*$.

4.3.5 Planar Layout

The final scale factors u provide an intrinsic description of the flattened surface, which we then lay out in the plane. Just as we do during optimization, we first scale the edge lengths and flip to Delaunay using Ptolemy edge flips to get a final triangulation (T^C, ℓ^C). Since the final edge lengths ℓ^C describe a triangulation that is flat away from cone singularities, we can simply lay the triangles out in the plane one at a time to get a parameterization with no flipped triangles. (The paper discusses numerically robust alternatives.) Since coordinates are discontinuous across cuts, we store values $z_i^{jk} \in \mathbb{R}^2$ at corners.

4.4 Results

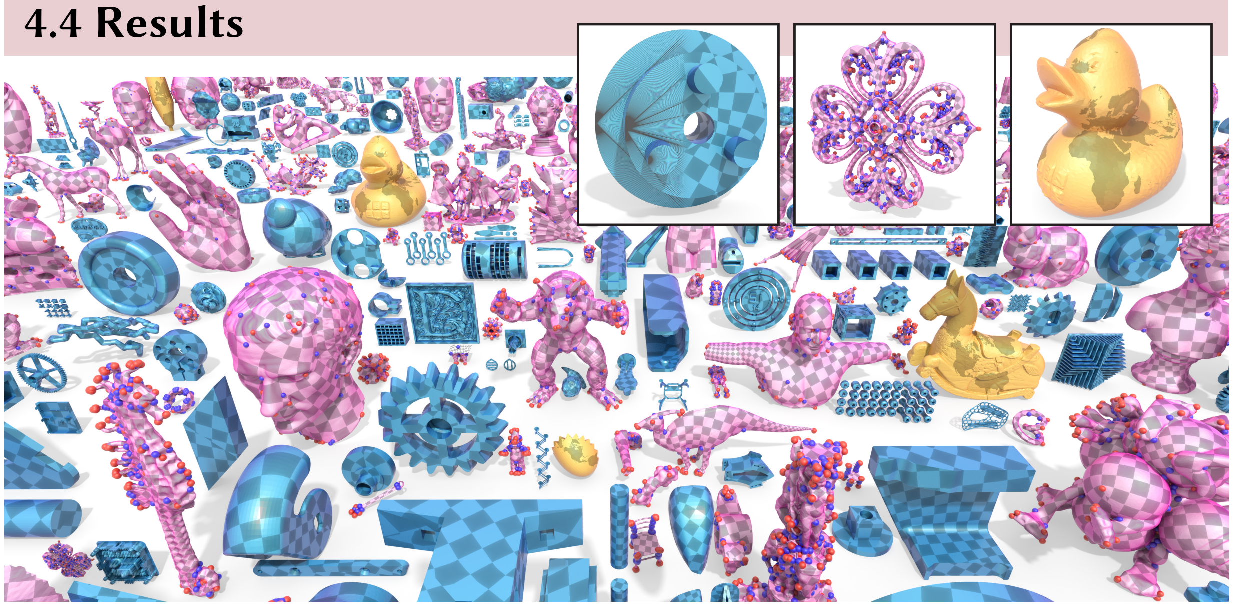


Figure 4.7: Our method computes locally injective, discretely conformal maps even for near-degenerate triangulations *(turquoise meshes)* and extremely difficult configurations of cone singularities *(magenta meshes)*. We also compute globally bijective conformal maps to the sphere *(yellow meshes)*.

This section evaluates the empirical behavior of our method, here referred to as *conformal equivalence of polyhedral surfaces (CEPS)*. Our main points of comparison is the CETM algorithm of Springborn et al. [2008], which does not use flips. All methods use identical code for tracking correspondence, *à la* Section 4.1. The overall observation is that CEPS succeeds on far more models than CETM. Even when CETM does succeed, it may not provide as good of an approximation of a smooth conformal map (Figure 4.8).

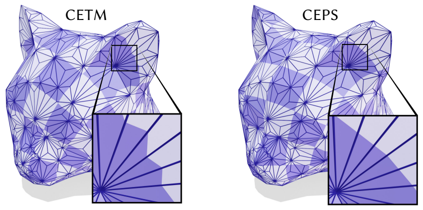


Figure 4.8: Even when CETM succeeds, the quality of the map may be lower since it uses a different notion of conformal equivalence (based on the input rather than Delaunay triangulation).

Difficult Cone Configurations. We ran our method on the standard benchmark of Myles et al. [2014], referred to as *MPZ*, which contains challenging cone configurations. CEPS succeeds on

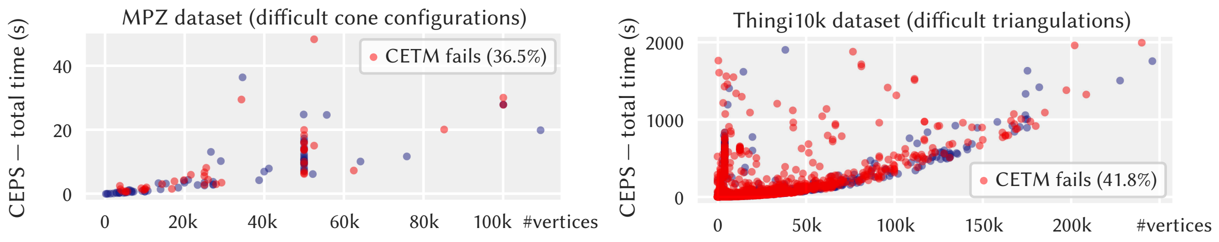


Figure 4.9: Timings for our method (CEPS) on two datasets. Note that CETM fails on a large percentage of models where we succeed (highlighted in red).

all 114 models, including extraction of the common refinement. Maps were discretely conformal up to floating point error, with an average length cross ratio error of about 10^{-9}, and no worse than about 10^{-4}. In contrast, CETM succeeded on only 73 models (Figure 4.9, *top*). Moreover, the tracing and refinement steps of CEPS could be trivially parallelized over edges and faces, *resp.*.

Many injective but non-conformal methods do not do as well on this difficult benchmark: as reported by Bright et al. [2017, Section 8.1], their method and the methods of Chien et al. [2016], Aigerman et al. [2014], Levi & Zorin [2014], and Lipman [2012] succeed on 104, 102, 97, 93, and 90 models, *resp.* Many of these methods have running times on the order of minutes or (on the most difficult examples) hours, versus seconds for our method. On the other hand, we must change/refine the triangulation, whereas these methods keep the triangulation fixed. Like CEPS, the combinatorial method of Zhou et al. [2020] succeeds on all MPZ models, but can yield highly distorted maps that are expensive to optimize; cost is again on the order of minutes to hours.

Difficult Triangulations. As a stress test of floating-point behavior, we parameterized all manifold meshes from Thingi10k, splitting disconnected meshes into their connected components (32,744 examples in total), and using a time out of 2000 seconds. Note that previous work on cone parameterization does not even attempt this benchmark, which has dramatically worse element quality than MPZ. For these examples we apply the greedy cone placement strategy from Springborn et al. [2008, Section 5.1], stopping when all log scale factors u_i are in the range $[-5, 5]$ (*i.e.*, a max scale factor of about 150). Here CEPS successfully computes a parameterized mesh S for 98.6% of models, yielding an injective map on 97.7%. Examples where we fail are quite pathological (*e.g.*, Figure 4.11, *right*). Overall about 68% and 15% of failures were due to failure of iterative straightening or optimization (*resp.*) to converge within the time limit, and for about 13% Delaunay flipping failed due to floating point error. The worst cross ratio error was typically around 10^{-5}. CETM fails on almost half of these examples (Figure 4.9).

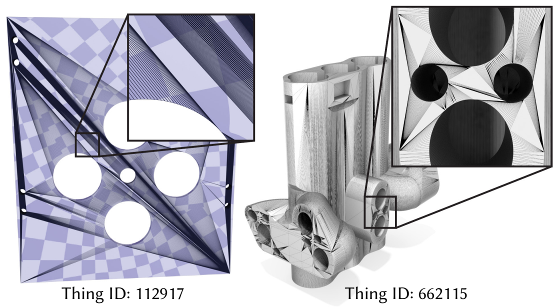


Figure 4.11: Our implementation robustly handles extremely poor triangulations (left) failing only on the most pathological inputs (right).

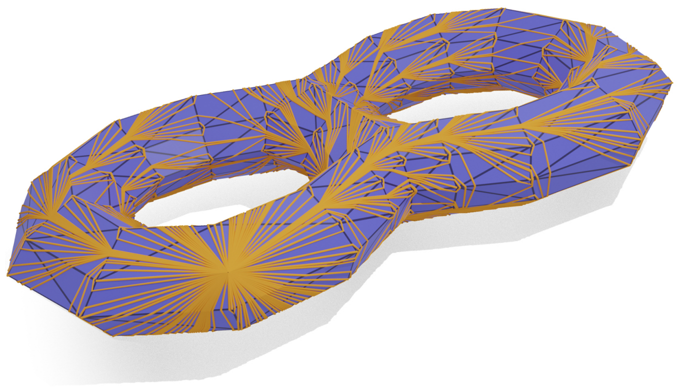


Figure 4.10: Since we allow edge flips, we need not worry how coarse the mesh is near large cones. Here we set all but one angle defect to almost 2π—the remaining vertex has an angle defect of -1032.79.

Chapter 5

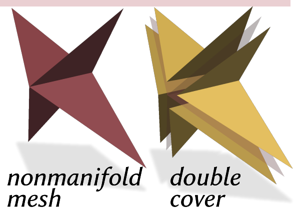
Proposed Work

Time is a tool you can put on the wall, or wear it on your wrist.
The past is far behind us, the future doesn't exist
Joseph Pelling and Becky Sloan, *Don't Hug Me I'm Scared*

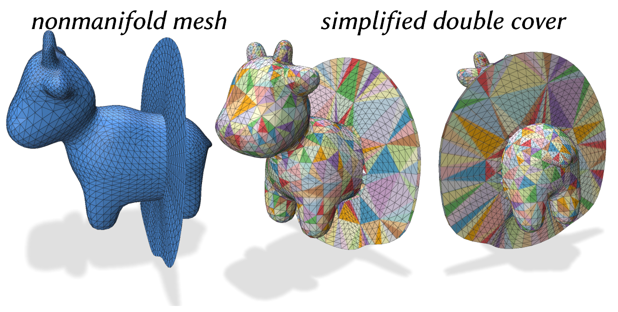
ONE key limitation of the intrinsic techniques presented so far is that they are limited to manifold triangle meshes. Nonmanifold meshes pose a challenge when working intrinsically: we often lay neighborhoods of our mesh out in the plane, but nonmanifold meshes are precisely the meshes which cannot always be mapped injectively to the plane. And worse, edge flips—a fundamental operation of intrinsic geometry processing—do not make sense for nonmanifold edges.

In Section 5.1, I propose a method for intrinsically simplifying nonmanifold meshes following the approach of Sharp & Crane [2020a]. In Section 5.2, I present my planned timeline to finish this work and other ongoing work in time to graduate in the spring.

5.1 Intrinsic Simplification of Nonmanifold Meshes



We can address these problems by following and considering the orientable double cover of a nonmanifold mesh (inset). The key idea is that given any mesh, we can construct a new manifold mesh by making two copies of each triangle and attaching them together in an appropriate way. If you think of the original mesh as describing a slightly-thickened volume, rather than an infinitesimally-thin surface, then this double cover models the boundary of the volume.



Importantly, functions on the original mesh correspond bijectively symmetric functions on the double cover, so if you need to compute a function with given properties on the original mesh, you can equivalently find a symmetric function on the double cover instead. Hence, to simplify a nonmanifold mesh intrinsically, we can first pass to its manifold double cover, and then simplify the double cover. However, doing so directly may produce different triangulations on the two sides of the double cover—especially in flat regions of the mesh or on models with a high degree of symmetry—which makes it difficult to specify what it means for a function to by "symmetric" on this simplified double cover. It is probably beneficial to force the two sides to have the same triangulation by always removing corresponding vertices on both sides at the same time. On the other hand, this will require some tricky implementation to allow backtracking if one vertex removal succeeds and the subsequent removal on the other side fails, and modifying the intrinsic simplification algorithm eliminates some of the conceptual simplicity of the double cover approach. So it will also be interesting to measure how well it works to construct the double cover and then run the existing simplification algorithm with no modification.

While investigating nonmanifold intrinsic simplification, I also plan to do a more thorough evaluation of the existing intrinsic simplification code and add some quality of life features to make it easier to use.

5.2 Timeline

1. Dec. 2023 – Jan. 2024: finish up ongoing work (Harnack tracing); submit to Siggraph
2. Jan. – Apr. 2024: investigate nonmanifold intrinsic simplification. And finish up an unrelated project on "circular arc triangulations" (meshes whose edges follow circular arcs rather than straight lines).
3. Apr. – Jun. 2024: write thesis
4. Jun. 2024: thesis defense

Bibliography

Abikoff, W. (Oct. 1981). “The uniformization theorem”. *The American Mathematical Monthly* 88.8, pp. 574–492. DOI: 10.2307/2320507.

Aigerman, N., Poranne, R., and Lipman, Y. (July 2014). “Lifted bijections for low distortion surface mappings”. *ACM Transactions on Graphics* 33.4, pp. 1–12. DOI: 10.1145/2601097.2601158.

Alekseevskij, D., Vinberg, E. B., and Solodovnikov, A. (June 1993). “Geometry of spaces of constant curvature”. *Geometry II*. Vol. 29. Encyclopaedia of Mathematical Sciences. Springer. DOI: 10.1007/978-3-662-02901-5_1.

Alexandrov, A. D. (1942). “Existence of a convex polyhedron and of a convex surface with a given metric”. *Matematicheskii Sbornik* 53.11, pp. 15–65.

Alexandrov, A. D. (1948). *Intrinsic Geometry of Convex Surfaces*. Vol. 2. OGIZ, Moscow-Leningrad. DOI: 10.1201/9780203643846.

Baumgart, B. G. (May 1975). “A polyhedron representation for computer vision”. *Proceedings of the May 19-22, 1975, National Computer Conference and Exposition*, pp. 589–596. DOI: 10.1145/1499949.1500071.

Bobenko, A. and Springborn, B. (2004). “Variational principles for circle patterns and Koebe’s theorem”. *Transactions of the American Mathematical Society* 356.2, pp. 659–689. DOI: 10.1090/S0002-9947-03-03239-2.

Bobenko, A. I. and Izmestiev, I. (2008). “Alexandrov’s theorem, weighted Delaunay triangulations, and mixed volumes”. *Annales de l’Institut Fourier*. Vol. 58. 2, pp. 447–505. DOI: 10.5802/aif.2358.

Bobenko, A. I., Pinkall, U., and Springborn, B. A. (2015). “Discrete conformal maps and ideal hyperbolic polyhedra”. *Geometry & Topology* 19.4, pp. 2155–2215. DOI: 10.2140/gt.2015.19.2155.

Bobenko, A. I. and Springborn, B. A. (Sept. 2007). “A discrete Laplace–Beltrami operator for simplicial surfaces”. *Discrete & Computational Geometry* 38.4, pp. 740–756. DOI: 10.1007/s00454-007-9006-1.

Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., and Lévy, B. (2010). *Polygon Mesh Processing*. DOI: 10.1201/b10688.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press. DOI: 10.1017/CBO9780511804441.

Bright, A., Chien, E., and Weber, O. (July 2017). “Harmonic global parametrization with rational holonomy”. *ACM Transactions on Graphics* 36.4, pp. 1–15. DOI: 10.1145/3072959.3073646.

Cannon, J. W., Floyd, W. J., Kenyon, R., Parry, W. R., et al. (1997). *Hyperbolic Geometry*. Vol. 31. MSRI Publications. ISBN: 0-521-62048-1.

Chebyshev, P. L. (1899). *Œuvres de P.L. Tchebychef*. Vol. 1. Commissionaires de l'Académie Impériale des Sciences.

Chien, E., Levi, Z., and Weber, O. (Dec. 2016). "Bounded distortion parametrization in the space of metrics". *ACM Transactions on Graphics* 35.6, pp. 1–16. DOI: 10.1145/2980179.2982426.

Crane, K., de Goes, F., Desbrun, M., and Schröder, P. (2013). "Digital geometry processing with discrete exterior calculus". *ACM SIGGRAPH 2013 Courses*. ACM. DOI: 10.1145/2504435.2504442.

de Goes, F., Memari, P., Mullen, P., and Desbrun, M. (June 2014). "Weighted triangulations for geometry processing". *ACM Transactions on Graphics* 33.3, pp. 1–13. DOI: 10.1145/2602143.

Finnendahl, U., Schwartz, M., and Alexa, M. (Apr. 2023). "Arap revisited discretizing the elastic energy using intrinsic voronoi cells". *Computer Graphics Forum (SGP)*. DOI: 10.1111/cgf.14790.

Fisher, M., Springborn, B., Schröder, P., and Bobenko, A. (Aug. 2007). "An algorithm for the construction of intrinsic delaunay triangulations with applications to digital geometry processing". *Computing* 81.2, pp. 199–213. DOI: 10.1007/s00607-007-0249-8.

Fumero, M., Möller, M., and Rodolà, E. (Nov. 2020). "Nonlinear spectral geometry processing via the tv transform". *ACM Transactions on Graphics* 39.6, pp. 1–16. DOI: 10.1145/3414685.3417849.

Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Alken, P., Booth, M., Rossi, F., and Ulerich, R. (1994). *GNU Scientific Library*. Vol. 20. ACM.

Garland, M. and Heckbert, P. S. (Aug. 1997). "Surface simplification using quadric error metrics". *SIGGRAPH 1997*. ACM, pp. 209–216. DOI: 10.1145/258734.258849.

Gauß, C. F. (1825). *General Investigations of Curved Surfaces*.

Gillespie, M., Sharp, N., and Crane, K. (Dec. 2021a). "Integer coordinates for intrinsic geometry processing". *ACM Transactions on Graphics* 40.6, pp. 1–13. DOI: 10.1145/3478513.3480522.

Gillespie, M., Springborn, B., and Crane, K. (July 2021b). "Discrete conformal equivalence of polyhedral surfaces". *ACM Transactions on Graphics* 40.4, pp. 1–20. DOI: 10.1145/3592401.

Glickenstein, D. (2005). "Geometric triangulations and discrete Laplacians on manifolds". *arXiv preprint*.

Glickenstein, D. (2023). "Geometric triangulations and discrete Laplacians on manifolds: an update". *Computational Geometry*. DOI: 10.1016/j.comgeo.2023.102063.

Gu, X. D., Guo, R., Luo, F., Sun, J., and Wu, T. (July 2018a). "A discrete uniformization theorem for polyhedral surfaces II". *Journal of Differential Geometry* 109.3, pp. 431–466. DOI: 10.4310/jdg/1531188190.

Gu, X. D., Luo, F., Sun, J., and Wu, T. (June 2018b). "A discrete uniformization theorem for polyhedral surfaces". *Journal of Differential Geometry* 109.2, pp. 223–256. DOI: 10.4310/jdg/1527040872.

Hatcher, A. (2002). *Algebraic Topology*. ISBN: 978-0-521-79540-1.

Hilbert, D. (Jan. 1901). "Ueber flächen von constanter Gaussscher krümmung". *Transactions of the American Mathematical Society* 2.1, pp. 87–99. DOI: 10.2307/1986308.

Hirsch, M. W. and Mazur, B. (1974). *Smoothings of Piecewise Linear Manifolds*. Annals of Mathematics Studies 80. Princeton University Press. DOI: 10.1515/9781400881680.

Hoppe, H. (Aug. 1996). “Progressive meshes”. *SIGGRAPH 1996*. ACM, pp. 99–108. DOI: 10.1145/237170.237216.

Indermitte, C., Liebling, T. M., Troyanov, M., and Clémençon, H. (2001). “Voronoi diagrams on piecewise flat surfaces and an application to biological growth”. *Theoretical Computer Science* 263.1, pp. 263–274. ISSN: 0304-3975. DOI: 10.1016/S0304-3975(00)00248-6.

Jin, M., Wang, Y., Yau, S.-T., and Gu, X. D. (2004). “Optimal global conformal surface parameterization”. *IEEE Visualization 2004*, pp. 267–274. DOI: 10.1109/VISUAL.2004.75.

Karcher, H. (2014). “Riemannian center of mass and so called Karcher mean”. *arXiv preprint arXiv:1407.2087*.

Kettner, L. (May 1999). “Using generic programming for designing a data structure for polyhedral surfaces”. *Computational Geometry* 13.1. DOI: 10.1016/S0925-7721(99)00007-3.

Kharevych, L., Springborn, B., and Schröder, P. (Apr. 2006). “Discrete conformal mappings via circle patterns”. *ACM Transactions on Graphics* 25.2, pp. 412–438. DOI: 10.1145/1138450.1138461.

Kirby, R. C. and Siebenmann, L. C. (July 1969). “On the triangulation of manifolds and the hauptvermutung”. *Bulletin of the American Mathematical Society* 75.4, pp. 742–749. DOI: 10.1090/S0002-9904-1969-12271-8.

Knöppel, F., Crane, K., Pinkall, U., and Schröder, P. (July 2013). “Globally optimal direction fields”. *ACM Transactions on Graphics* 32.4. DOI: 10.1145/2461912.2462005.

Lee, A. W. F., Sweldens, W., Schröder, P., Cowsar, L. C., and Dobkin, D. P. (July 1998). “MAPS: multiresolution adaptive parameterization of surfaces”. *SIGGRAPH 1998*. ACM, pp. 95–104. DOI: 10.1145/280814.280828.

Lee, J. M. (2012). *Introduction to Smooth Manifolds*. 2nd ed. Vol. 218. Graduate Texts in Mathematics. Springer. ISBN: 978-1-4419-9981-8. DOI: 10.1007/978-1-4419-9982-5.

Levi, Z. and Zorin, D. (Nov. 2014). “Strict minimizers for geometric optimization”. *ACM Transactions on Graphics* 33.6, pp. 1–14. DOI: 10.1145/2661229.2661258.

Lipman, Y. (July 2012). “Bounded distortion mapping spaces for triangular meshes”. *ACM Transactions on Graphics* 31.4, pp. 1–13. DOI: 10.1145/2185520.2185604.

Liu, H.-T. D., Gillespie, M., Chislett, B., Sharp, N., Jacobson, A., and Crane, K. (July 2023). “Surface simplification using intrinsic error metrics”. *ACM Transactions on Graphics* 42.4, pp. 1–17. ISSN: 0730-0301. DOI: 10.1145/3592403.

Liu, H. D., Kim, V. G., Chaudhuri, S., Aigerman, N., and Jacobson, A. (Aug. 2020). “Neural subdivision”. *ACM Transactions on Graphics* 39.4, pp. 1–16. DOI: 10.1145/3386569.3392418.

Liu, H. D., Zhang, J. E., Ben-Chen, M., and Jacobson, A. (July 2021). “Surface multigrid via intrinsic prolongation”. *ACM Transactions on Graphics* 40.4, pp. 1–13. DOI: 10.1145/3450626.3459768.

Luo, F. (2004). “Combinatorial Yamabe flow on surfaces”. *Communications in Contemporary Mathematics* 6.5, pp. 765–780. DOI: 10.1142/S0219199704001501.

MacNeal, R. H. (1949). “The Solution of Partial Differential Equations by Means of Electrical Networks”. PhD thesis. California Institute of Technology. DOI: 10.7907/PZ04-5290.

Mainini, E. (2012). “A description of transport cost for signed measures”. *Journal of Mathematical Sciences* 181, pp. 837–855. DOI: 10.1007/s10958-012-0718-2.

Milnor, J. (Sept. 1956). “On manifolds homeomorphic to the 7-sphere”. *Annals of Mathematics* 64.2, pp. 399–405. DOI: 10.2307/1969983.

Mitchell, J. S., Mount, D. M., and Papadimitriou, C. H. (1987). “The discrete geodesic problem”. *SIAM Journal on Computing* 16.4, pp. 647–668. doi: 10.1137/0216045.

Moise, E. E. (July 1952). “Affine structures in 3-manifolds: V. the triangulation theorem and hauptvermutung”. *Annals of Mathematics* 56.1, pp. 96–114. doi: 10.2307/1969769.

Moise, E. E. (2013). *Geometric Topology in Dimensions 2 and 3*. 1st ed. Vol. 47. Springer. doi: 10.1007/978-1-4612-9906-6.

Mosher, L. (Mar. 1988). “Tiling the projective foliation space of a punctured surface”. *Transactions of the American Mathematical Society* 306.1, pp. 1–70. doi: 10.2307/2000830.

Myles, A., Pietroni, N., and Zorin, D. (July 2014). “Robust field-aligned global parametrization”. *ACM Transactions on Graphics* 33.4, pp. 1–14. doi: 10.1145/2601097.2601154.

Penner, R. C. (Jan. 2012). *Decorated Teichmüller Theory*. QGM Master Class Series. European Mathematical Society, Zürich. doi: 10.4171/075.

Radó, T. (1925). “Über den Begriff der Riemannschen Fläche”. *Acta Litt. Sci. Univ. Szeged* 2, pp. 101–121.

Regge, T. (1961). “General relativity without coordinates”. *Il Nuovo Cimento (1955-1965)* 19.3. doi: 10.1007/BF02733251.

Riemann, B. (1854). *On the Hypotheses which lie at the Bases of Geometry*. doi: 10.1007/978-3-319-26042-6.

Rivin, I. (May 1994). “Euclidean structures on simplicial surfaces and hyperbolic volume”. *Annals of Mathematics* 139.3, pp. 553–580. doi: 10.2307/2118572.

Roček, M. and Williams, R. M. (1984). “The quantization of Regge calculus”. *Zeitschrift für Physik C Particles and Fields* 21.4, pp. 371–381. doi: 10.1007/BF01581603.

Santambrogio, F. (2015). *Optimal Transport for Applied Mathematicians*. Vol. 87. Progress in Nonlinear Differential Equations and Their Applications. Birkäuser Cham. doi: 10.1007/978-3-319-20828-2.

Sawhney, R. and Crane, K. (2017). “Boundary first flattening”. *ACM Transactions on Graphics* 37.5, pp. 1–14. doi: 10.1145/3132705.

Schroeder, W. J., Zarge, J. A., and Lorensen, W. E. (July 1992). “Decimation of triangle meshes”. *SIGGRAPH 1992*. ACM, pp. 65–70. doi: 10.1145/142920.134010.

Shamai, G. and Kimmel, R. (July 2017). “Geodesic distance descriptors”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. doi: 10.1109/CVPR.2017.386.

Sharp, N. and Crane, K. (Aug. 2020a). “A Laplacian for nonmanifold triangle meshes”. *Computer Graphics Forum (SGP)* 39.5. doi: 10.1111/cgf.14069.

Sharp, N. and Crane, K. (Nov. 2020b). “You can find geodesic paths in triangle meshes by just flipping edges”. *ACM Transactions on Graphics* 39.6. doi: 10.1145/3414685.3417839.

Sharp, N., Gillespie, M., and Crane, K. (July 2021). “Geometry processing with intrinsic triangulations”. ACM SIGGRAPH 2021 Courses. doi: 10.1145/3450508.3464592.

Sharp, N., Soliman, Y., and Crane, K. (July 2019). “Navigating intrinsic triangulations”. *ACM Transactions on Graphics* 38.4. doi: 10.1145/3306346.3322979.

Shewchuk, J. R. (2002). “What Is a Good Linear Finite Element? Interpolation, Conditioning, Anisotropy, and Quality Measures”.

Springborn, B., Schröder, P., and Pinkall, U. (Aug. 2008). “Conformal equivalence of triangle meshes”. *ACM Transactions on Graphics* 27.3, pp. 1–11. doi: 10.1145/1360612.1360676.

Springborn, B. A. (Sept. 2019). “Ideal hyperbolic polyhedra and discrete uniformization”. *Discrete & Computational Geometry* 64, pp. 63–108. DOI: 10.1007/s00454-019-00132-8.

Steinitz, E. (1908). “Beiträge zur Analysis situs”. *Sitzungsberichte der Berliner Mathematische Gesellschaft* 7, pp. 29–49.

Sun, J., Wu, T., Gu, X. D., and Luo, F. (2015). “Discrete conformal deformation: algorithm and experiments”. *SIAM Journal on Imaging Sciences* 8.3. DOI: 10.1137/141001986.

Thurston, D. and Yuan, Q. (2012). “Notes on Curves on Surfaces”.

Tietze, H. (Dec. 1908). “Über die topologischen invarianten mehrdimensionaler mannigfaltigkeiten”. *Monatshefte für Mathematik und Physik* 19, pp. 1–118. DOI: 10.1007/BF01736688.

Troyanov, M. (1986). “Les surfaces Euclidiennes à singularités coniques”. *L'Enseignement Mathématique* 32, pp. 79–94. DOI: 10.5169/seals-55079.

Verhoeven, F., Vaxman, A., Hoffmann, T., and Sorkine-Hornung, O. (Mar. 2022). “Dev2pq: planar quadrilateral strip remeshing of developable surfaces”. *ACM Transactions on Graphics* 41.3, pp. 1–18. DOI: 10.1145/3510002.

Weiler, K. (Jan. 1985). “Edge-based data structures for solid modeling in curved-surface environments”. *IEEE Computer Graphics and Applications* 5.1, pp. 21–40. DOI: 10.1109/MCG.1985.276271.

Whitehead, J. H. C. (Oct. 1940). “On C^1-complexes”. *Annals of Mathematics* 41.4, pp. 809–824. DOI: 10.2307/1968861.

Zhou, J., Tu, C., Zorin, D., and Campen, M. (2020). “Combinatorial construction of seamless parameter domains”. *Computer Graphics Forum*. Vol. 39, pp. 179–190. DOI: 10.1111/cgf.13922.

Appendix A

A Brief Introduction to Hyperbolic Geometry

A.1 Models of Hyperbolic Geometry

Just as the sphere S^2 is a surface of constant curvature $K = +1$, the *hyperbolic plane* H^2 is a surface of constant negative curvature $K = -1$. Unlike S^2, there is no way to smoothly embed H^2 in Euclidean $\mathbb{R}^3$ *isometrically*, *i.e.*, without distorting its geometry [Hilbert 1901]. Instead, we must visualize it through one of several *models*, each of which faithfully represents only some of its geometric features. A good analogy is the Mercator projection of the globe, which preserves angles but distorts the size of land masses. Figure A.1 depicts three models that are useful for our purposes. For further background on hyperbolic geometry, see Alekseevskij et al. [1993] and Cannon et al. [1997].

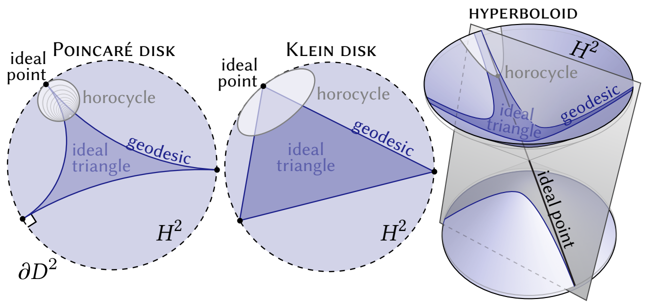


Figure A.1: Since the hyperbolic plane H^2 cannot be isometrically embedded in $\mathbb{R}^3$, it must be understood through the use of several "models"—here we illustrate how several key quantities are realized in each model.

In the *Poincaré disk model*, points in H^2 are identified with points in the open unit disk $D^2 := \{p \in \mathbb{R}^2 : |p| < 1\}$. Although this disk looks like a finite piece of the Euclidean plane, lengths at a point $p \in D^2$ get scaled by $2/(1-|p|^2)$ so that short distances near the boundary ∂D^2 represent large distances in H^2. One can hence travel any distance along a straightest curve or *geodesic* without ever reaching the boundary—limit points on ∂D^2 are called *ideal points*. Though geodesics are straight in H^2, in the Poincaré model they appear as circular arcs orthogonal to ∂D^2. The Poincaré model is conformal: angles between circular arcs give the true angle between geodesics in H^2. Finally, just as a straight line in $\mathbb{R}^2$ can be viewed as a circle of "infinite radius," a *horocycle* is the limit of a family of increasingly large circles tangent at a common point—drawn in the Poincaré model as a circle tangent to the boundary.

The *Beltrami-Klein model* is much like the Poincaré model, but with a different metric. Geodesics appear as straight lines, but Euclidean angles no longer give the true angles in H^2, *i.e.*, the Beltrami-Klein model is not conformal. Horocycles in the Beltrami-Klein model appear as

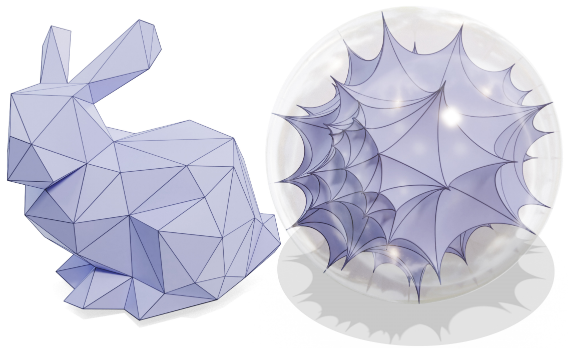
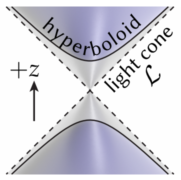


Figure A.2: An ordinary triangle mesh *(left)* can always be viewed as an *ideal hyperbolic polyhedron (right)*, *i.e.*, surface made from triangles of constant negative curvature and all three vertices at infinity.

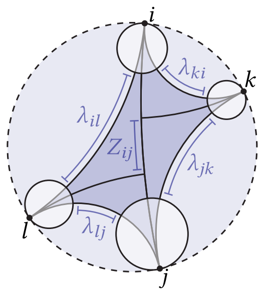
ellipses. This model helps explain the relationship between Euclidean and hyperbolic polyhedra (Appendix A.2.1).



The *hyperboloid model* represents H^2 as the upper sheet of the two-sheeted hyperboloid. Just as the sphere is the set of all points $p \in \mathbb{R}^3$ such that $\langle p, p\rangle = 1$, this hyperboloid is the set of all points satisfying $\langle p, p\rangle_{2,1} = -1$, where $\langle p, q\rangle_{2,1} := p_x q_x + p_y q_y - p_z q_z$ is the *Lorentz inner product*; this inner product is also used to measure the angles and lengths of vectors tangent to the hyperboloid. Geodesics in H^2 correspond to intersections of the hyperboloid with planes through the origin, and ideal points are identified with lines in the *light cone* $\mathcal{L} := \{p \in \mathbb{R}^3 : \langle p, p\rangle_{2,1} = 0\}$. Horocycles are obtained by taking a plane tangent to $\mathcal{L}$, shifting it in the positive z-direction, and intersecting with the hyperboloid. Thus, we can identify horocycles with points in the *positive light cone* $\mathcal{L}^+ := \{p \in \mathcal{L} : p_z > 0\}$; each point $p \in \mathcal{L}^+$ also corresponds to the plane $\{q \in \mathbb{R}^3 : \langle p, q\rangle_{2,1} = -1\}$. The hyperboloid model is essential for developing our interpolation scheme—see Section 4.2.4.

A.2 Ideal Polyhedra

An *ideal hyperbolic polyhedron* is a surface of constant negative curvature, and a finite collection of *cusps* analogous to Euclidean cone points (Figure A.2, *right*). We can construct ideal polyhedra by gluing together *ideal triangles*: regions of H^2 bounded by three geodesics approaching three ideal points at infinity (Figure A.1). A strange fact about ideal triangles is that they are all congruent, *i.e.*, they are identical up to isometries of H^2. Hence, the geometry of an ideal polyhedron is determined entirely by how neighboring triangles ijk, jil are glued together—namely, how far we slide them along the shared geodesic ij. One way to quantify gluings is to use *shear coordinates*, which for each edge ij give the distance $Z_{ij} \in \mathbb{R}$ between the altitudes dropped from opposite vertices k and l (see inset). Alternatively, we can pick an arbitrary horocycle at each vertex, yielding a *decorated ideal polyhedron*. Though edges of an ideal triangle do not have finite length, there is now a finite distance $\lambda_{ij} \in \mathbb{R}$ between the horocycles at i and j—these values are called the *Penner coordinates*. Shear and Penner coordinates are related by



$$Z_{ij} = \tfrac{1}{2}(\lambda_{il} - \lambda_{lj} + \lambda_{jk} - \lambda_{ki}) \quad \text{(A.1)}$$

(see [Penner 2012, Corollary 4.16, p. 40]). Note that if the horocycles at i and j overlap, λ_{ij} will be negative. Yet unlike negative Euclidean lengths, negative Penner coordinates will cause no trouble for discrete uniformization. Likewise, whereas Euclidean lengths must satisfy the triangle inequality, any three Penner coordinates $\lambda_{ij}, \lambda_{jk}, \lambda_{ki} \in \mathbb{R}$ (whether positive or negative) can be realized by some choice of horocycles.

A.2.1 Euclidean-Ideal Correspondence

Every Euclidean polyhedron gives rise to an ideal polyhedron, in the following way. Any triangle $ijk \in \mathsf{F}$ drawn in its Euclidean circumdisk can be interpreted as an ideal triangle in the Beltrami-Klein model. To glue two ideal triangles ijk, jil together along an edge ij, we simply identify the same points as in the Euclidean polyhedron. An ideal polyhedron constructed this way will have shear coordinates $Z_{ij} = \log \mathfrak{c}_{ij}$, and if we assign Penner coordinates

$$\lambda_{ij} = 2 \log \ell_{ij} \tag{A.2}$$

we get a decorated version of the same polyhedron. In general, we can move from Euclidean to hyperbolic polyhedra by "taking a logarithm"—for example, Equation (A.1) now just becomes the logarithm of the length cross ratio. More importantly, for a *fixed* triangulation, a conformal scaling of edge lengths corresponds to a shift in horocycles of the form

$$\tilde{\lambda}_{ij} = \lambda_{ij} + u_i + u_j. \tag{A.3}$$

In other words, conformally equivalent edge lengths $\ell, \tilde{\ell}$ describe the same ideal polyhedron, just decorated with different horocycles.

A.2.2 Ptolemy Flip

Penner coordinates are easily updated during edge flips via *Ptolemy's relation* [Penner 2012, Corollary 4.16, p. 40]. Letting $\ell_{ij} = e^{\lambda_{ij}/2}$ for each edge in Figure A.3 *(top right)*, we compute

$$\ell_{kl} = (\ell_{ki}\ell_{lj} + \ell_{jk}\ell_{li})/\ell_{ij}. \tag{A.4}$$

The new Penner coordinate is then $\lambda_{kl} = 2\log(\ell_{kl})$ (Figure A.3, *top right*). Since Equation (A.4) is a rational expression in ℓ, it is often simplest to just store and manipulate the edge lengths ℓ rather than the Penner coordinates λ. See the paper for further discussion of numerics.

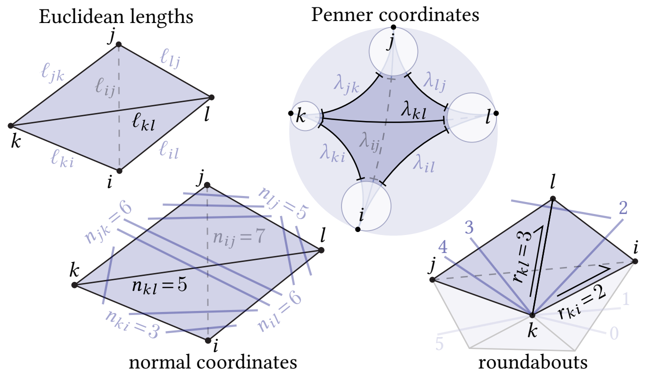


Figure A.3: For each edge flip, we need to update any data stored on edges. Here we indicate quantities involved in updating Euclidean edge lengths *(top left)*, Penner coordinates *(top right)*, normal coordinates *(bottom left)* and roundabouts *(bottom right)*.

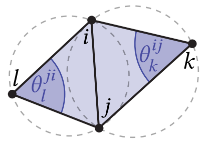
Importantly, this so-called *Ptolemy flip* is the same as a Euclidean edge flip if and only if the two Euclidean triangles are concyclic. In general, Euclidean flips may distort the discrete conformal structure even though they preserve the Euclidean geometry, whereas Ptolemy flips always preserve the hyperbolic metric, hence the conformal structure. Moreover, Euclidean flips are well-defined only when the triangle inequalities are satisfied, whereas Ptolemy flips are always well-defined.

A.2.3 Delaunay Triangulations

For polyhedral surfaces, discrete conformal equivalence is defined in terms of *Delaunay triangulations*—not because they are "nice" in a numerical sense, but because they are key to establishing the discrete uniformization theorem. Delaunay triangulations have similar but distinct definitions in the Euclidean and ideal hyperbolic settings.

A.2.4 Intrinsic Delaunay Triangulations

A planar triangulation is Delaunay if there are no vertices inside any triangle circumcircle. Equivalently, we can ask that every interior edge ij satisfy the *local Delaunay condition*



$$\theta_k^{ij} + \theta_l^{ji} \leq \pi. \tag{A.5}$$

This characterization generalizes to Euclidean polyhedra, since the edge lengths ℓ are sufficient to determine the angles θ. Such *intrinsic Delaunay triangulations* can be found using a simple greedy algorithm: while any edge fails to satisfy Equation (A.5), perform a Euclidean flip. This algorithm terminates after finitely many flips [Bobenko & Springborn 2007; Indermitte et al. 2001], and in practice takes about $|E|$ flips on real-world meshes [Sharp et al. 2019, Figure 10]. Note if two triangles are inscribed in a common circle, then either diagonal satisfies Equation (A.5).

A.2.5 Ideal Delaunay Triangulations

A hyperbolic analogue is an *ideal Delaunay triangulation* [Springborn 2019, Section 4]: if $\ell = e^{\lambda/2}$ are edge lengths associated with given Penner coordinates λ, then every edge must satisfy the *local ideal Delaunay condition*

$$\ell_{ij}^2(\ell_{jk}\ell_{ki} + \ell_{il}\ell_{lj}) < (\ell_{il}\ell_{ki} + \ell_{jk}\ell_{lj})(\ell_{il}\ell_{jk} + \ell_{ki}\ell_{lj}), \tag{A.6}$$

which we obtain by combining Equations 3 and 10 from Springborn [2019]. We can again find such a triangulation by greedily flipping edges, but this time using Ptolemy flips. Remarkably, if Equation (A.6) is satisfied globally, then the lengths ℓ always describe a valid *Euclidean* intrinsic Delaunay triangulation [Springborn 2019, p. 4.14]. Yet working in the ideal setting enables us to start with lengths that do not describe a valid Euclidean metric and flip to a valid one.