Discrete Conformal Equivalence of Polyhedral Surfaces

MARK GILLESPIE, Carnegie Mellon University, USA
BORIS SPRINGBORN, Technische Universität Berlin, Germany
KEENAN CRANE, Carnegie Mellon University, USA

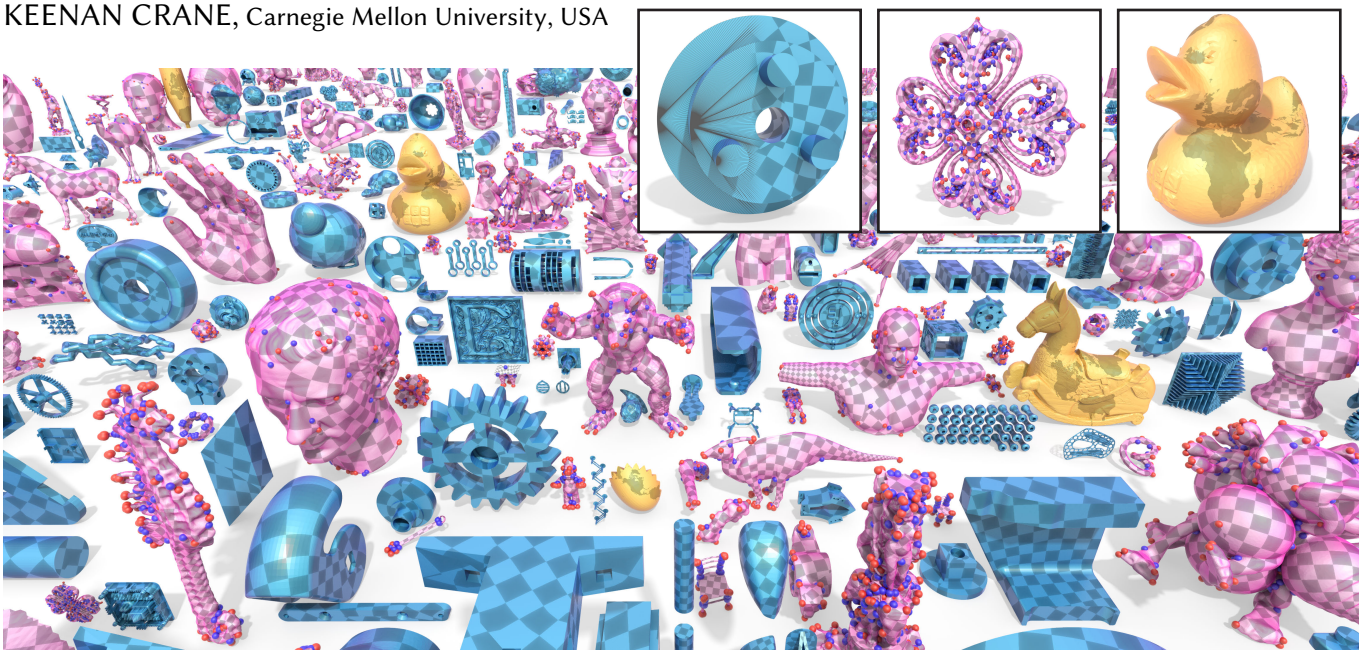


Fig. 1. Our method computes locally injective, discretely conformal maps even for near-degenerate triangulations *(turquoise meshes)* and extremely difficult configurations of cone singularities *(magenta meshes)*. We also compute globally bijective conformal maps to the sphere *(yellow meshes)*.

This paper describes a numerical method for surface parameterization, yielding 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, or globally bijective maps from genus zero surfaces to the sphere. Recent theoretical developments show 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 particular, we introduce a new scheme for tracking correspondence between triangulations based on *normal coordinates*, and a new interpolation procedure based on layout in the *light cone*. Stress tests involving difficult cone configurations and near-degenerate triangulations indicate that the method is extremely robust in practice, and provides high-quality interpolation even on meshes with poor elements.

Authors' addresses: Mark Gillespie, Carnegie Mellon University, USA; Boris Springborn, Technische Universität Berlin, Germany; Keenan Crane, Carnegie Mellon University, USA.

0730-0301/2021/8-ART103
https://doi.org/10.1145/3450626.3459763

CCS Concepts: • **Mathematics of computing → Mesh generation**.

Additional Key Words and Phrases: conformal parameterization, injective mapping, discrete differential geometry

ACM Reference Format:
Mark Gillespie, Boris Springborn, and Keenan Crane. 2021. Discrete Conformal Equivalence of Polyhedral Surfaces. *ACM Trans. Graph.* 40, 4, Article 103 (August 2021), 20 pages. https://doi.org/10.1145/3450626.3459763

1 INTRODUCTION

Angle-preserving or *conformal* maps play a key role in applied geometry [Gu et al. 2020], ranging from biological shape analysis to digital fabrication to machine learning [Koehl and Hass 2015; Konaković et al. 2016; Maron et al. 2017]. Conformal maps are desirable for many reasons: they provide canonical mappings between shapes [Lipman and Daubechies 2011; Baden et al. 2018]; typically involve only sparse linear systems or easy convex problems (hence scale to very large meshes); and provide both low angle *and* area distortion when enriched with well-placed cuts or *cone singularities* [Kharevych et al. 2006; Ben-Chen et al. 2008; Sharp and Crane 2018; Soliman et al. 2018].

To date, however, conformal mapping algorithms fail to guarantee that a valid map is always found: if the input surface is too

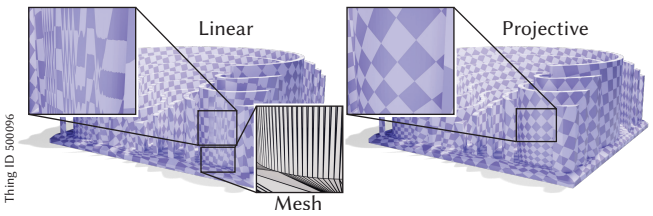


Fig. 3. For meshes with low-quality elements, standard linear interpolation yields a poor conformal map *(left)*. We describe how to perform projective interpolation across triangulations, yielding a much nicer map *(right)*.

poorly triangulated, or the target curvatures are too extreme, existing methods find a map that is not locally injective, or simply fail to find any map at all. Such failures hinder the reliability of broader geometry processing algorithms that depend on conformal maps.

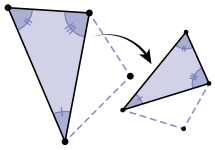
In the smooth setting, existence of conformal maps is guaranteed by the *uniformization theorem* [Abikoff 1981]. Very recently, Gu et al. [2018a,b] and Springborn [2019] established an analogous *discrete uniformization theorem* for triangle meshes. However, these theoretical results fall short of providing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

- a novel combinatorial data structure for tracking correspondence between different triangulations (Section 5),
- a new interpolation scheme for evaluating the discrete conformal map based on the *light cone* (Section 6), and
- critical details needed to implement discrete uniformization including a careful treatment of numerics, boundary conditions, 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 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 flip the input to an *intrinsic Delaunay triangulation*. The main difficulty in developing a practical algorithm is therefore tracking and evaluating the correspondence between three triangulations—Figure 2 gives an overview of the whole process.

2 RELATED WORK

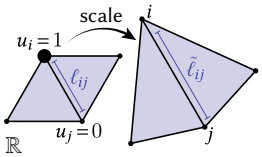
2.1 Discrete Conformal Equivalence



In the smooth setting, conformal maps preserve angles—naïvely, one might therefore require that for triangle meshes, discrete conformal maps preserve the angles at all corners. However, this condition is far too rigid: since each triangle can only scale and rotate, its neighbors—and in turn, the entire surface—may only scale by a constant amount. As a result, many other notions of discrete conformal maps have been explored; Crane [2020] gives a detailed account.

A particularly successful approach is the notion of *discrete conformal equivalence*. In the smooth setting, two Riemannian metrics $g, \tilde{g}$ (which determine angles) are conformally equivalent if they are related by a positive scaling $\tilde{g} = e^{2u}g$ for some real-valued function u. On a triangle mesh, the Riemannian metric is captured by the lengths ℓ_{ij} of all edges ij, and two sets of lengths $\ell, \tilde{\ell}$ are called discretely conformally equivalent if

$$\tilde{\ell}_{ij} = e^{(u_i+u_j)/2}\ell_{ij} \qquad (1)$$



for some assignment of *scale factors* $u_i \in \mathbb{R}$ to vertices i [Roček and Williams 1984; Luo 2004]. This innocent-looking definition leads to a rich discrete theory which is just as flexible as the smooth one [Bobenko et al. 2015]. Bücking [2016, 2018] and Gu et al. [2019] consider convergence under refinement.

2.1.1 *Discrete Uniformization.*

Conformal equivalence offers an appealing strategy for parameterization: rather than solve directly for a map to the plane, first find scale factors that describe a discretely conformally equivalent flat surface—perhaps with target angle defects Ω_i^* prescribed at just a few isolated *cone points* (Figure 4, *top right*). This new surface is then cut open and unfolded into the plane (Figure 4, *bottom*). In the smooth setting, existence of such scale factors is guaranteed by the *uniformization theorem* [Abikoff 1981]

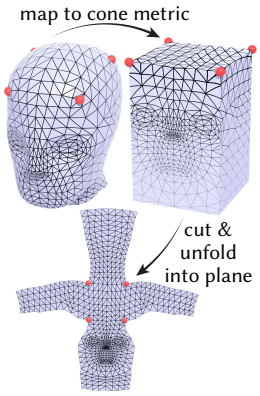


Fig. 4. Conformal parameterization with cone singularities.

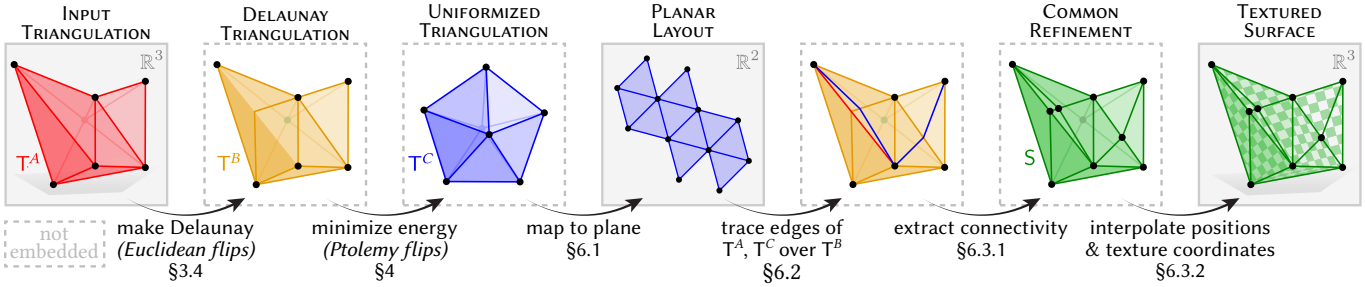


Fig. 2. Steps of our algorithm. Throughout we color the input mesh T^A red, its intrinsic Delaunay triangulation T^B yellow, the uniformized triangulation T^C blue, and the common refinement S of all three green. (Note: triangulations in dashed boxes are purely intrinsic and never actually embedded in $\mathbb{R}^n$.)

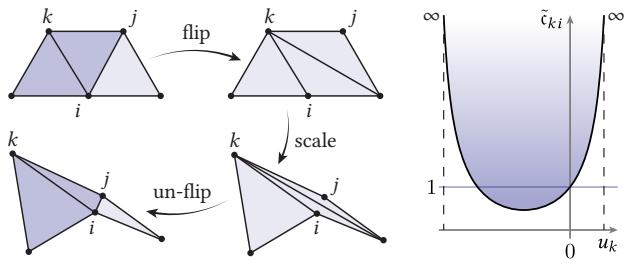


Fig. 5. Performing Euclidean edge flips at arbitrary moments in the flow can badly distort the conformal structure. Here, we flip edge ij, scale edges incident on k by a factor $e^{u_k/2}$, and undo the flip. The cross ratio $\tilde{c}_{ki}$ of edge ki (Equation 4) is not preserved, and in fact can take almost any value.

and its generalization to cone metrics [Troyanov 1991]. In the discrete setting, however, there is a critical problem: for a fixed triangulation, there may be no scale factors that achieve the target angle defects. One must therefore adopt an expanded notion of discrete conformal equivalence that allows the triangulation to change (Section 2.1.2).

To actually compute the scale factors, Luo [2004] proposed the *discrete Yamabe flow*

$$\frac{d}{dt}u_i(t) = \Omega_i^* - \widetilde{\Omega}_i(t). \tag{2}$$

Here $\widetilde{\Omega}_i(t)$ are the angle defects induced by the scale factors $u(t)$. However, since there may be no scale factors that achieve the target angle defects, this flow can fail to reach a critical point $\frac{d}{dt}u_i = 0$, where $\widetilde{\Omega}_i = \Omega_i^*$. In this case, the scaled edge lengths $\tilde{\ell}$ will eventually violate the triangle inequality—at which point the flow becomes ill-defined and cannot continue. Springborn et al. [2008] and Bobenko et al. [2015] describe this flow as gradient descent on an explicit convex energy $\mathcal{E}$, leading to the more efficient, 2nd-order *CETM algorithm*. CETM extends $\mathcal{E}$ to be well-defined even for invalid edge lengths—but if the minimizer is found in this extended region, it fails to describe a valid parameterization (Figure 25).

Flipping Edges. Luo [2004] conjectured that degenerate triangles might be avoided by applying Euclidean edge flips at the exact moment when triangles degenerate, as implemented by Campen and Zorin [2017b, Section 7.3.1], but this idea has two fatal flaws. First, mixing flips with vertex scaling can yield lengths that are not conformally equivalent to the original ones (Figure 5). Second, it can cause discontinuities in the value of $\mathcal{E}$, voiding any guarantee that the flow will converge (Figure 7). This lack of guarantees is a problem even for methods that care only about injectivity, and not conformal maps [Chien et al. 2016; Campen and Zorin 2017b,a; Campen et al. 2019]. Likewise, the generalized method of Chen et al. [2016, Algorithm 1] takes a step of arbitrary size before performing power Delaunay flips, and [Yu et al. 2017, Algorithm 1] takes an arbitrary step before performing Euclidean flips. Both algorithms can hence distort conformal structure, or worse, produce edge lengths that violate the triangle inequality—at which point the flow is undefined and cannot continue. Our use of *Ptolemy flips* ensures the flow is always well-defined and exactly preserves the conformal structure (see Section 3.3.4, and the use of Algorithm 11 within Algorithm 4).

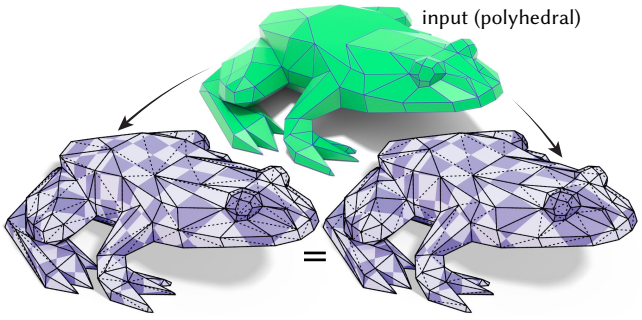


Fig. 6. We adopt a notion of conformal equivalence that yields the same discrete conformal map, no matter how the input polyhedral surface is triangulated. Here a mesh with planar faces is triangulated two different ways, yielding identical results.

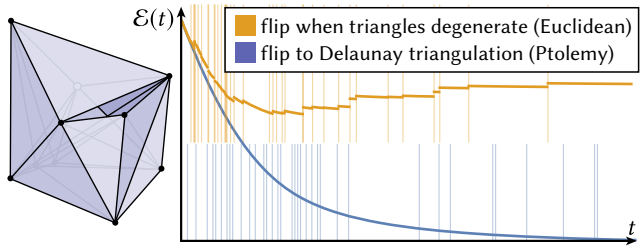


Fig. 7. Flipping edges when triangles degenerate causes the energy $\mathcal{E}$ to jump discontinuously—voiding any guarantee of convergence *(top)*. In contrast, flipping to Delaunay via Ptolemy flips before evaluating the energy ensures that we always reach the correct solution *(bottom)*. Here we consider a coarse double torus with target angle defects $+3\pi/4$ at all but one vertex, which has large negative curvature. We take small steps to clearly plot the energy; vertical lines indicate flip times.

2.1.2 Variable Triangulations. A recent theoretical breakthrough is a notion of discrete conformal equivalence that does not depend on how a polyhedral surface is triangulated (Figure 6), along with associated *discrete uniformization theorems* for the Euclidean [Gu et al. 2018a], hyperbolic [Gu et al. 2018b], and spherical [Springborn 2019] cases. This work is intimately linked to realization results for ideal hyperbolic polyhedra [Rivin 1994; Fillastre 2008; Prosanov 2020]. The theorems guarantee one can *always* find a conformally equivalent triangulation with prescribed angle defects Ω^*, so long as they satisfy Gauss-Bonnet. This solution is unique up to scale (Euclidean case) or Möbius transformations (spherical case).

There are two equivalent definitions of discrete conformal equivalence—a key idea introduced by Gu et al. [2018a] is to consider an *intrinsic Delaunay triangulation* of the input (Section 3.4).

One definition is that two Delaunay triangulations are conformally equivalent if they are related by an alternating sequence of vertex scalings (Equation 1) and concyclic Euclidean edge flips (Figure 9), which maintain the Delaunay property [Gu et al. 2018a,

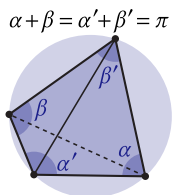


Fig. 9. Either triangulation of a circular quad satisfies the local Delaunay property $\alpha+\beta \leq \pi$.

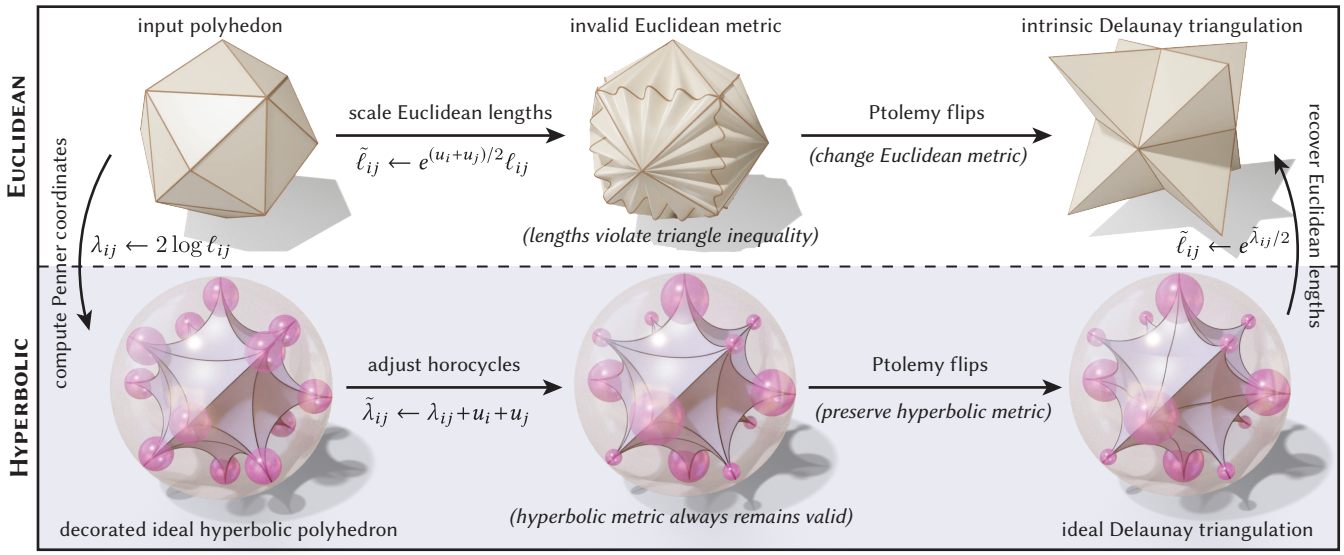


Fig. 8. *Top:* Triangle meshes with different connectivity (but the same vertices) are considered discretely conformally equivalent if they are the same up to a conformal rescaling of edge lengths, followed by Ptolemy edge flips to a Delaunay triangulation. *Bottom:* This definition, and the use of Ptolemy (rather than Euclidean) edge flips, arises from a hyperbolic perspective, where we simply retriangulate a hyperbolic polyhedron without changing its geometry.

Definition 1.1]. Algorithms that adopt this definition must stop and flip whenever two triangles become concyclic. Wu [2014] shows that only finitely many flips are needed, ensuring that computation terminates. Sun et al. [2015] present an implementation of such a scheme, but do not evaluate the pointwise map between the domain and target (as needed for, *e.g.*, texture mapping or remeshing).

We adopt an alternative definition which is theoretically equivalent—though this is far from obvious: the two triangulations are discretely conformally equivalent if they describe the same *ideal hyperbolic polyhedron* [Bobenko et al. 2015, Definition 5.1.4]. As observed by Springborn [2019], a discretely conformally equivalent triangulation can be obtained by applying an arbitrary vertex scaling, then flipping to a Delanuay triangulation via *Ptolemy flips* (Section 3.3.4), rather than ordinary Euclidean flips—see Figure 8, *top*. Since Ptolemy flips are well-defined even when edge lengths fail to satisfy the triangle inequality, one need not worry about maintaining a valid Euclidean metric, nor about triangles being concyclic: at any moment, one can simply scale to an *invalid* metric, then flip to a valid one. This procedure always works, because it corresponds to retriangulating the associated *ideal hyperbolic polyhedron* (Figure 8, *bottom*). Concurrent work by Campen et al. [2021] also takes this approach.

By adopting this definition, we cast discrete conformal parameterization as an unconstrained convex optimization problem where the only variables are the scale factors u_i. The optimizer need not worry about edge flips, which appear only within a black-box callback routine that evaluates the energy and its derivatives. Moreover, we can use a 2nd-order Newton method to achieve fast convergence, since the energy we minimize is twice continuously differentiable (C^2) even across different triangulations, and the Hessian is easy to compute (just the cotan-Laplacian). Overall this approach is generally faster than stopping to perform flips (see Figure 10 and Section 8.3), and also accommodates the more difficult spherical case, which involves additional bounds constraints (Section 7).

2.2 Discrete Conformal Mapping

The triangulation produced by uniformization cannot be used in most applications unless we know how to map data back to the input mesh. Two basic strategies have been developed for this purpose. Fisher et al. [2007] maintain an explicit mesh of the common refinement of two triangulations, guaranteeing correct connectivity (Sun et al. [2015] adopt a similar approach). Sharp et al. [2019b] observe that explicit encodings incur significant cost, and instead implicitly encode correspondence via *signposts* at vertices. This floating-point encoding can however fail to describe correct connectivity in extreme situations (such as Figure 27). We provide the best of both worlds: an implicit, integer-based encoding that can be updated without resolving intersections, yet guarantees the right connectivity (Section 5). This encoding is based on *normal coordinates*, a tool

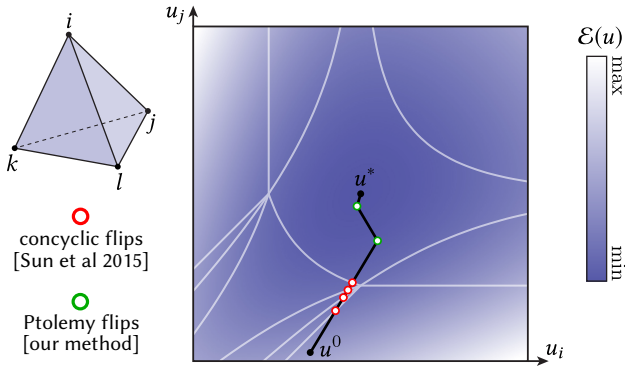


Fig. 10. A slice of the energy landscape for a tetrahedron. Each conformal scaling u induces a Delaunay triangulation—white curves delineate regions with a common triangulation. Previous algorithms must stop and flip at each region boundary (where triangles become concyclic), whereas we can flip at any moment—since Ptolemy flips commute with scaling.

from 3-manifold geometry [Kneser 1929; Haken 1961] and computational topology [Schaefer et al. 2002; Agol et al. 2006; Erickson and Nayyeri 2013]. We enrich this construction with a combinatorial analogue of signposts, which we call *roundabouts* (Section 5).

The other question is how to interpolate data across triangulations, such as vertex or texture coordinates. The natural choice for discrete conformal maps is to use piecewise projective interpolation [Bobenko et al. 2015], which can be implemented via standard homogeneous coordinates [Springborn et al. 2008, Section 3.4]. We extend this idea to variable triangulations by laying out triangles in the *light cone* rather than the Euclidean plane (see Section 6.0.1).

Importantly, our approach to discrete conformal mapping depends critically on the hyperbolic picture. Without this picture, one could not use the implicit connectivity encoding (which depends on hyperbolic straightening), and would be forced to explicitly maintain the full connectivity of the common refinement, as done by Sun et al. [2015]. Likewise, our high-quality interpolation scheme (shown in Figure 3) relies on calculations in the light cone model of the hyperbolic plane.

2.3 Other Methods

Conformal Mapping. Other methods for conformal parameterization do not provide a general solution. For instance, early methods based on linear finite elements [Lévy et al. 2002; Desbrun et al. 2002; Mullen et al. 2008] do not guarantee injectivity, nor do they handle cone singularities. More recent linear methods support cones [Ben-Chen et al. 2008; Vintescu et al. 2017; Sawhney and Crane 2017], but injectivity is still missing. *Orbifold* methods (*e.g.*, [Aigerman and Lipman 2016]) provide injectivity, but support only a restricted set of cone configurations where cone angles cannot be prescribed. Angle-based methods [Sheffer et al. 2005] rely on nonconvex optimization, with no general convergence guarantees. Finally, Bobenko et al. [2015] and later Zhang et al. [2014] provide connections between discrete conformal equivalence and *circle patterns*.

Injective Mapping. Discrete uniformization has a special relationship to methods for locally injective mapping, since CETM is often used for initialization [Chien et al. 2016; Campen and Zorin 2017b; Campen et al. 2019]; we provide even stronger guarantees. Unlike [Mandad and Campen 2019; Shen et al. 2019] we do not claim to guarantee injectivity in floating point—yet still achieve injectivity in extremely challenging scenarios (Section 8.3.2). Overall we observe that the freedom to modify the triangulation during optimization leads to significantly improved robustness—see Section 8.3.

3 PRELIMINARIES

This section provides essential definitions needed to motivate and derive our algorithms; some readers may wish to skip ahead to Section 4, and return here for reference. The most important concept is illustrated in Figure 14: any triangle mesh can be interpreted as both a *Euclidean polyhedron* (Section 3.2) and a *decorated ideal polyhedron* (Section 3.3), leading to a definition of conformal equivalence across different triangulations (Section 3.5). For further background, see Bobenko et al. [2015] and Springborn [2019].

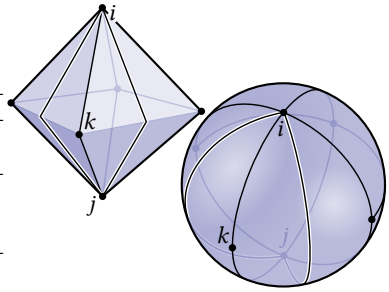
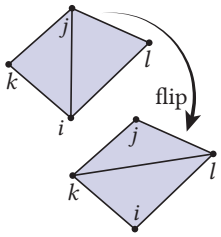


Fig. 11. An edge or triangle in a Δ-complex might not be uniquely determined by its vertices. Here, performing intrinsic edge flips on an octahedron yields two distinct edges between the same pair of vertices i and j, and two triangles with the same vertices i, j, and k. The sphere depicts the abstract connectivity.

3.1 Combinatorial Polyhedra

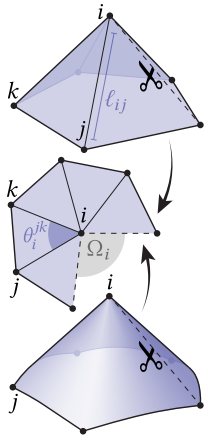
Throughout we use $\mathsf{T} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$ to denote the connectivity of a manifold triangulation with vertices V, edges E, and faces F; we assume T is orientable purely to simplify exposition. Even when the input is an ordinary (simplicial) triangulation, we may need to construct triangulations where, *e.g.*, multiple edges connect the same two vertices, or two triangles share the same three vertices (Figure 11). Formally, we use *triangulation* to mean a Δ-complex in the sense of Hatcher [2002, Section 2.1], which we encode via a *halfedge data structure* [Botsch et al. 2010, Section 2.3]. Though edges and faces are not uniquely determined by their vertices, for brevity we will still denote them by vertex pairs $ij \in \mathsf{E}$ and triples $ijk \in \mathsf{F}$, *resp.*, where i, j, and k need not be distinct. The notation ϕ_i^{jk} indicates a quantity ϕ at corner i of a triangle ijk.

3.1.1 Combinatorial Edge Flip. For two triangles sharing a common edge, an *edge flip* replaces this edge with the opposite diagonal—we will need this operation in order to construct intrinsic Delaunay triangulations (Section 3.4). If we locally index the vertices of these two triangles as depicted in the inset figure, then the edge flip replaces the original triangles ijk and jil with jkl and lki. Any other data stored on the triangulation must also be updated, as depicted in Figure 12 and discussed in Sections 3.2.1, 3.3.4, 5.1.1 and 5.2.1.



3.2 Euclidean Polyhedra

A *Euclidean polyhedron* is a surface that looks like the flat Euclidean plane everywhere except at a finite collection of *cone points*. The canonical example is an ordinary triangle mesh in $\mathbb{R}^3$, where the neighborhood around each vertex is isometric to a piece of a circular cone (see inset). For uniformization, however, we do not need to keep track of how the surface is embedded in space. Instead, we can store a purely *intrinsic* description of the geometry, given by the edge lengths $\ell : \mathsf{E} \to \mathbb{R}_{>0}$ of a triangulation $\mathsf{T} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$. If these lengths satisfy the triangle inequalities in each triangle $ijk \in \mathsf{F}$, then we call ℓ a *discrete metric*. Other quantities such as corner



angles θ_i^{jk} can be recovered from edge lengths via standard trigonometric formulas. In particular, each vertex $i \in \mathsf{V}$ has an *angle defect* $\Omega_i := 2\pi - \sum_{ijk \in \mathsf{F}} \theta_i^{jk}$, which characterizes the flatness of the vertex, and is equal to the integral of Gaussian curvature over a small neighborhood around the cone point.

3.2.1 Euclidean Edge Flip. This description also enables us to change the triangulation of a Euclidean polyhedron without changing its intrinsic geometry. In particular, given only the edge lengths, the new edge length ℓ_{kl} resulting from an edge flip can be determined by laying out the known triangles ijk and jil in the Euclidean plane, and measuring the distance between vertices k and l. More robust numerical strategies are discussed by Fisher et al. [2007] and Sharp et al. [2019b].

3.2.2 Conformally Equivalent Edge Lengths. We say that two discrete metrics $\ell, \tilde{\ell} : \mathsf{E} \to \mathbb{R}_{>0}$ on the same triangulation $\mathsf{T} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$ are *discretely conformally equivalent* if at all edges $ij \in \mathsf{E}$

$$\tilde{\ell}_{ij} = e^{(u_i + u_j)/2} \, \ell_{ij} \tag{3}$$

for some assignment of vertex scale factors $u : \mathsf{V} \to \mathbb{R}$. These metrics are conformally equivalent if and only if they induce the same *length cross ratios* [Springborn et al. 2008, Section 2]

$$\mathrm{c}_{ij} = \frac{\ell_{il}\ell_{jk}}{\ell_{lj}\ell_{ki}}. \tag{4}$$

We will give a definition of conformal equivalence for Euclidean polyhedra with *different* connectivity in Section 3.5.

3.3 Hyperbolic Polyhedra

3.3.1 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 13 depicts three models that are useful for our purposes. For further background on hyperbolic geometry, see Cannon et al. [1997]; Alekseevskij et al. [1993].

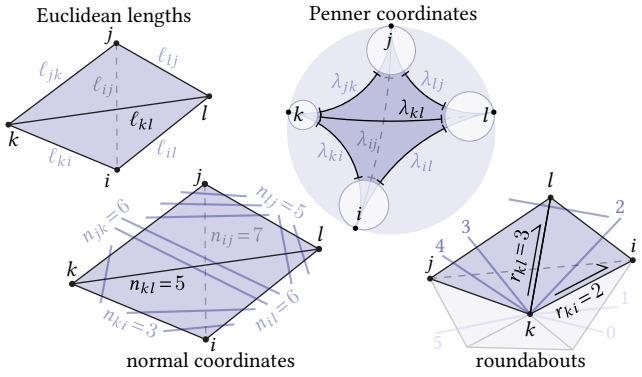


Fig. 12. For each edge flip, we need to update any data stored on edges. Here we indicate quantities involved in updating Euclidean edge lengths *(top left)*, Penner coordinates *(top right)*, normal coordinates *(bottom left)* and roundabouts *(bottom right)*.

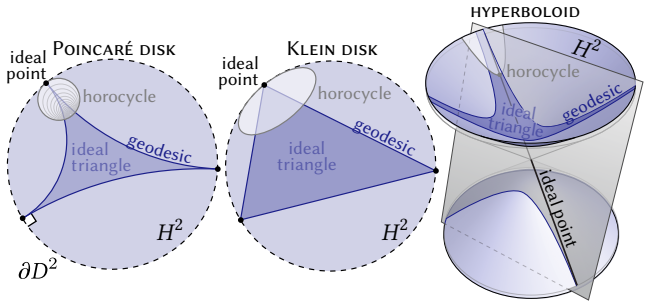
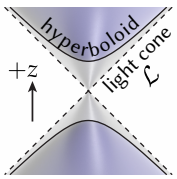


Fig. 13. Since the hyperbolic plane H^2 cannot be isometrically embedded in $\mathbb{R}^3$, it must be understood through the use of several "models"—here we illustrate how several key quantities are realized in each model.

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 ellipses. This model helps explain the relationship between Euclidean and hyperbolic polyhedra (Section 3.3.3).

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 6.4.



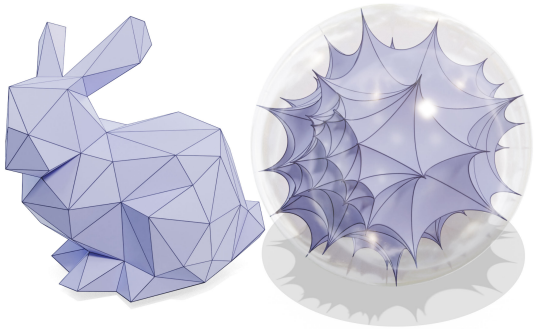
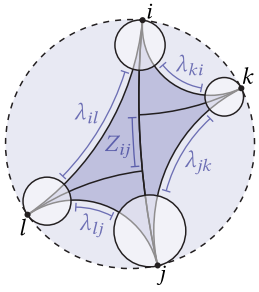


Fig. 14. An ordinary triangle mesh *(left)* can always be viewed as an *ideal hyperbolic polyhedron (right)*, *i.e.*, surface made from triangles of constant negative curvature and all three vertices at infinity.

3.3.2 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 14, *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 13). 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



$$Z_{ij} = \tfrac{1}{2}(\lambda_{il} - \lambda_{lj} + \lambda_{jk} - \lambda_{ki}) \tag{5}$$

(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.

3.3.3 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

$$\lambda_{ij} = 2 \log \ell_{ij} \tag{6}$$

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 5 now just becomes the logarithm of Equation 4. More importantly, for a *fixed* triangulation, a conformal scaling of edge lengths *à la* Equation 3 corresponds to a shift in horocycles of the form

$$\tilde{\lambda}_{ij} = \lambda_{ij} + u_i + u_j. \tag{7}$$

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

3.3.4 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 12 *(top right)*, we compute

$$\ell_{kl} = (\ell_{ki}\ell_{lj} + \ell_{jk}\ell_{li})/\ell_{ij}. \tag{8}$$

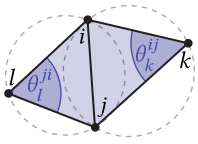
The new Penner coordinate is then $\lambda_{kl} = 2\log(\ell_{kl})$ (Figure 12, *top right*). Since Equation 8 is a rational expression in ℓ, it is often simplest to just store and manipulate the edge lengths ℓ rather than the Penner coordinates λ. See Section 8.1 for further discussion of numerics.

Importantly, this so-called *Ptolemy flip* is the same as a Euclidean edge flip if and only if the two Euclidean triangles are concyclic (Figure 9). In general, Euclidean flips may distort the discrete conformal structure even though they preserve the Euclidean geometry (Figure 15), 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.

3.4 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 mentioned in Section 1. Delaunay triangulations have similar but distinct definitions in the Euclidean and ideal hyperbolic settings.

3.4.1 Intrinsic Delaunay Triangulations. A planar triangulation is Delaunay if there are no vertices on the interior of any triangle circumcircle. Equivalently, we can ask that every interior edge ij contained in triangles ijk, jil satisfy the *local Delaunay condition*



$$\theta_k^{ij} + \theta_l^{ji} \leq \pi. \tag{9}$$

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 9, perform a Euclidean flip (*à la* Section 3.2.1). This algorithm terminates after finitely many flips [Indermitte et al. 2001; Bobenko and Springborn 2007], and in practice takes about $|\mathsf{E}|$ flips on real-world meshes [Sharp et al. 2019b, Figure 10]. Note if two triangles are inscribed in a common circle, then either diagonal satisfies Equation 9 (Figure 9).

3.4.2 Ideal Delaunay Triangulations. A hyperbolic analogue is an *ideal Delaunay triangulation* [Springborn 2019, Section 4]: if $\ell = e^{\lambda/2}$

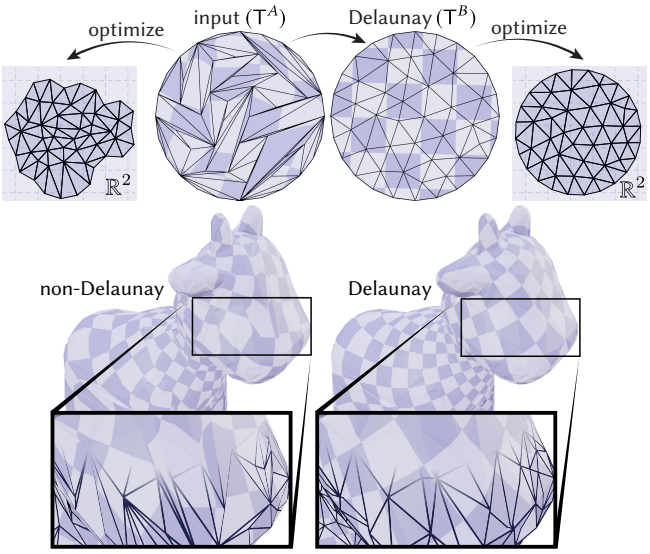


Fig. 15. *Top:* uniformization should leave a flat region unchanged, but unless one first flips to an intrinsic Delaunay triangulation, Ptolemy flips performed during optimization will distort the given shape. *Bottom:* in general, flipping to intrinsic Delaunay first tends to yield a better map.

are edge lengths associated with given Penner coordinates λ, then every edge must satisfy the *local ideal Delaunay condition*

$$\ell_{ij}^2(\ell_{jk}\ell_{ki} + \ell_{il}\ell_{lj}) < (\ell_{il}\ell_{ki} + \ell_{jk}\ell_{lj})(\ell_{il}\ell_{jk} + \ell_{ki}\ell_{lj}), \qquad (10)$$

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 10 is satisfied globally, then the lengths ℓ always describe a valid *Euclidean* intrinsic Delaunay triangulation [Springborn 2019, 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 (Figure 8).

3.5 Discrete Conformal Equivalence

We can now state what it means for polyhedra with different triangulations to be discretely conformally equivalent. Consider in particular two Euclidean polyhedra with the same vertex set V, encoded as intrinsic Delaunay triangulations (T, ℓ) and $(\tilde{\mathsf{T}}, \tilde{\ell})$. Two mathematically equivalent definitions provide not only different geometric perspectives, but also lead to different algorithms.

Euclidean perspective. One definition of discrete conformal equivalence is that there must exist a sequence of Euclidean intrinsic Delaunay triangulations

$$(\mathsf{T}, \ell) = (\mathsf{T}_1, \ell_1), \ \ldots, \ (\mathsf{T}_n, \ell_n) = (\tilde{\mathsf{T}}, \tilde{\ell})$$

where each consecutive pair $(\mathsf{T}_i, \ell_i), (\mathsf{T}_{i+1}, \ell_{i+1})$ is related by either (i) a conformal scaling of edge lengths, *à la* Equation 3, or (ii) Euclidean edge flips of concyclic triangle pairs, *à la* Section 3.2.1.

Hyperbolic perspective. The other definition says that (T, ℓ) and $(\tilde{\mathsf{T}}, \tilde{\ell})$ are discretely conformally equivalent if the associated ideal hyperbolic polyhedra (as defined in Section 3.3.3) are isometric, *i.e.*, if they simply describe different triangulations of the same negatively-curved surface. Concretely, any modification of the initial Penner coordinates via Equation 7 followed by Ptolemy flips to an ideal Delaunay triangulation will yield a discretely conformally equivalent surface. This perspective is illustrated in Figure 8.

An important difference between these two perspectives is that in the Euclidean case one must stop to perform edge flips whenever the triangulation becomes non-Delaunay, whereas in the hyperbolic case scaling and flipping are decoupled: one can adjust Penner coordinates freely, and need not stop to perform flips.

4 UNIFORMIZATION

Here we describe our procedure for planar parameterization—see Section 7 for the spherical case. This procedure is outlined in Figure 2; detailed pseudocode can be found in the supplement.

Given an input mesh T^A, we first flip to an intrinsic Delaunay triangulation T^B (*à la* Section 3.4.1), 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). After optimization, we lay T^C out in the plane (Section 4.5). However, this layout does not yet provide a mapping of the input mesh to the plane—Sections 5 and 6 describe how to construct such a map. Note that if we skip the first step (*i.e.*, do not flip to intrinsic Delaunay) then we could work with just two triangulations, and get a map that is still locally injective, but may exhibit conformal distortion (see Figures 15 and 24).

4.1 Variational Formulation

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

$$\tfrac{1}{2\pi} \sum_{i \in \mathsf{V}} \Omega_i^* = |\mathsf{V}| - |\mathsf{E}^B| + |\mathsf{F}^B| \qquad (11)$$

(see Section 4.4 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.

Note that unlike CETM we flip to Delaunay whenever we need to evaluate the energy or its derivatives (as detailed in Section 4.2). 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).

4.2 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 $\widetilde{T} = (\mathsf{V}, \widetilde{\mathsf{E}}, \widetilde{\mathsf{F}})$ using Ptolemy flips. These flips change the Euclidean geometry but preserve the discrete conformal structure. We will use $\tilde{\lambda}$, $\tilde{\theta}$, and $\widetilde{\Omega}$ to denote the corresponding Penner coordinates, interior angles, and angle defects, *resp.*

4.2.1 Energy. The discrete conformal energy is then given by

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

where $f(\tilde{\lambda}_{ij}, \tilde{\lambda}_{jk}, \tilde{\lambda}_{ki})$ is given by

$$\tfrac{1}{2}\left(\tilde{\theta}_i^{jk}\tilde{\lambda}_{jk} + \tilde{\theta}_j^{ki}\tilde{\lambda}_{ki} + \tilde{\theta}_k^{ij}\tilde{\lambda}_{ij}\right) + \text{Л}(\tilde{\theta}_i^{jk}) + \text{Л}(\tilde{\theta}_j^{ki}) + \text{Л}(\tilde{\theta}_k^{ij}).$$

Here Л denotes *Milnor's Lobachevsky function*

$$\text{Л}(\theta) := -\int_0^\theta \log|2\sin u|\, du,$$

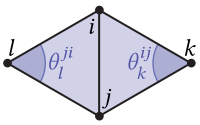
which is related to *Clausen's integral* via $\text{Л}(\theta) = \frac{1}{2}\text{Cl}_2(2\theta)$; the latter is implemented in standard numerical packages [Galassi et al. 1994]. Unlike CETM, which extends the energy linearly to handle lengths that violate the triangle inequality, we always evaluate this energy on the intrinsic Delaunay triangulation implied by the current scale factors. Constant shifts relative to [Springborn et al. 2008, Equation 7] ensure that, when evaluated this way, the energy, its gradient, and its Hessian vary continuously with the log scale factors u—even though different scale factors can induce different triangulations [Springborn 2019, Proposition 7.12]. Note also that a *Euclidean* edge flip preserves this energy $\mathcal{E}$ if and only if the two participating triangles are concyclic—again motivating the use of Delaunay triangulations.

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

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

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

4.2.3 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. 2013a, Chapter 6]. Since a Δ complex may contain more than one edge with the same endpoints (see for example Figure 11), 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 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 and Vandenberghe [2004], *resp.* In particular, we use the descent direction $v \in \mathbb{R}^{|\mathsf{V}|}$ obtained by solving the linear system

$$Lv = \partial_u \mathcal{E}, \tag{12}$$

where $\partial_u\mathcal{E} \in \mathbb{R}^{|\mathsf{V}|}$ encodes the gradient defined in Section 4.2.2. 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).

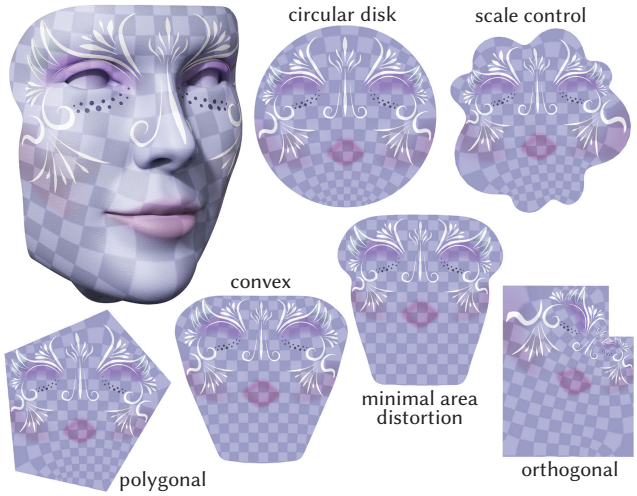
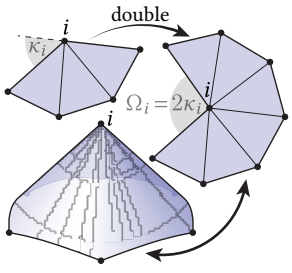


Fig. 16. Our algorithm guarantees existence of a locally injective discrete conformal map for any prescribed boundary lengths or angles, which can be used to achieve a rich variety of behavior. Spherical uniformization also provides a globally injective conformal map to the unit disk.

4.4 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 and 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 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 the spherical uniformization from Section 7.

4.4.1 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 $\sum_{i\in V}\Omega_i^* + \sum_{i\in\partial V}\kappa_i^* = |V| - |E| + |F|$. In Figure 16 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).

4.4.2 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 16. 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 12), 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 $\widetilde{\Omega} = \Omega^*$.

4.5 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 (*à la* Equation 3) 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 (Figure 4), we can simply lay the triangles out in the plane one at a time to get a parameterization with no flipped triangles. (Section 8.2 discusses a numerically robust alternative.) Since coordinates are discontinuous across cuts, we store values $z_i^{jk} \in \mathbb{R}^2$ at corners.

5 CORRESPONDENCE

We now describe a data structure for tracking 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 6. Since this encoding uses only integer data, it avoids robustness issues that can arise with floating-point representations (*e.g.*, Figure 27).

Explicitly, to encode the correspondence between any two triangulations $\mathsf{T}_1, \mathsf{T}_2$ with the same vertex set V, we store

- *normal coordinates*, which count the number of times T_1 crosses each edge of T_2 (Section 5.1), and
- *roundabouts*, which give the circular ordering of edges from both T_1 and T_2 around each vertex (Section 5.2).

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 6.1). 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 (Figure 11).

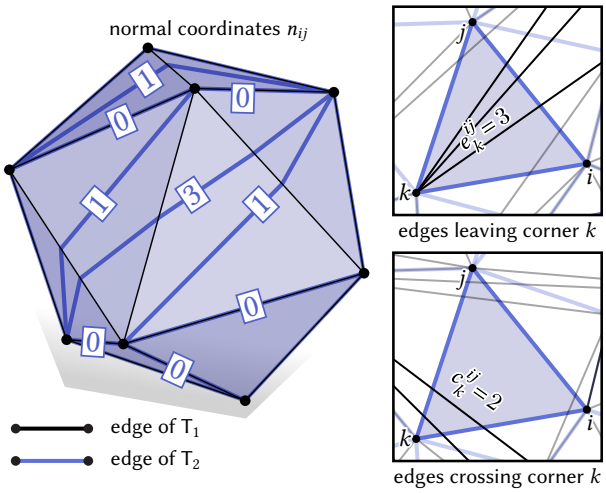


Fig. 17. *Left:* normal coordinates n_{ij} count the number of times each edge ij in a triangulation T_2 crosses any edge of another triangulation T_1 transversely. *Right:* these coordinates can be used to determine other quantities, such as how many edges of T_1 cross or leave a corner of a triangle from T_2.

For our flattening procedure we use this scheme to track the correspondence both between T^A and T^B, and between T^B and T^C (see Figure 2). Note that in the remaining sections we use H to denote the *halfedges* associated with edges E, *i.e.*, the two possible orientations $\overrightarrow{ij} \neq \overrightarrow{ji}$ of each edge ij in E.

5.1 Normal Coordinates

Normal coordinates count the number of times a collection of curves cross each edge of a fixed triangulation (Figure 17). 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 5.1.1. 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 17, *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):

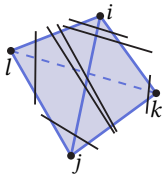
$$e_k^{ij} = \max\left(0, n_{ij} - n_{jk} - n_{ki}\right). \tag{13}$$

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

$$c_k^{ij} = \tfrac{1}{2}\left(\max\left(0, n_{jk} + n_{ki} - n_{ij}\right) - e_i^{jk} - e_j^{ki}\right). \tag{14}$$

See Figure 17, *right* for examples.

5.1.1 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 and Yuan 2012, Equation 1]:

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

In the general case, we must derive a more complicated formula (see supplement):

$$n_{kl} = \max\Big(0,\ c_l^{ji} + c_k^{ij} + \tfrac{1}{2}\left|c_j^{il} - c_j^{ki}\right| + \tfrac{1}{2}\left|c_i^{lj} - c_i^{jk}\right| - \tfrac{1}{2}e_l^{ji} - \tfrac{1}{2}e_k^{ij} + e_i^{lj} + e_i^{jk} + e_j^{il} + e_j^{ki} + \delta_{n_{ij}}\Big). \quad (15)$$

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

5.2 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 (Figure 11). 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.

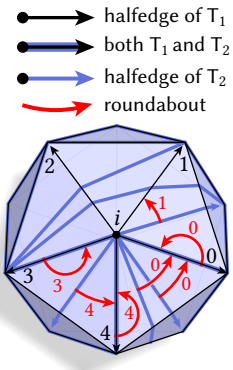


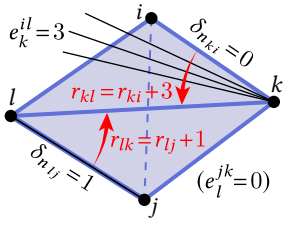
Fig. 18. For each halfedge of T_2, the roundabout gives the next halfedge of T_1.

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 18). 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.

5.2.1 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 5.1.1. We then have

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

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_k^{il} 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. [2019b, 3.2.1]: integer indices $r_{\overrightarrow{ij}}$ play the role of real-valued directions; the integer counts e_i^{jk} play the role of real-valued angles.



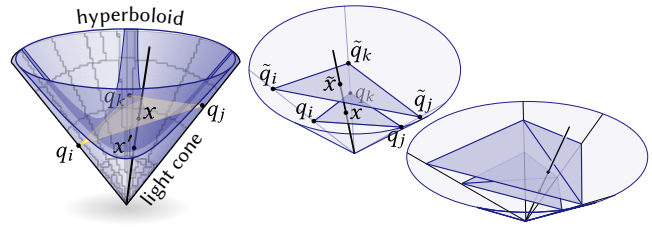


Fig. 19. By drawing triangles in the light cone *(left)*, the map between surfaces can be found by drawing a straight line through the origin *(center)*, which also works for two different triangulations *(right)*.

6 MAPPING

Following uniformization (Section 4), 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 2). 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 5 we first trace out geodesics to identify the points where edges of T^A and T^C intersect edges of T^B (Section 6.1). We then use these points to construct a *common refinement* S, *i.e.*, the smallest polygonal tessellation that contains all three triangulations (Section 6.3). Finally, we interpolate the functions f and z across S (Section 6.4). The result is an ordinary polygon mesh with vertex coordinates $f_i \in \mathbb{R}^3$ and texture coordinates z_i^{jk} at each triangle corner; these texture coordinates can be used for either piecewise projective or standard piecewise linear interpolation.

6.0.1 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; Sun et al. 2015; Springborn 2019], is to consider *chordal triangles* in the light cone—leading to simple interpolation formulas in homogeneous coordinates (*e.g.*, Equation 16). We here give a brief sketch, which is made more precise in the supplement. In particular, take any Euclidean triangle and place it in $\mathbb{R}^3$ so that its vertices sit at points q_i, q_j, q_k on the light cone (Figure 19, *left*). As discussed in Section 3.3, these points also define the vertices of a decorated ideal triangle. Hence, central projection from any point x on the Euclidean triangle to the hyperboloid provides an explicit mapping between the Euclidean and ideal triangle. Moreover, if we

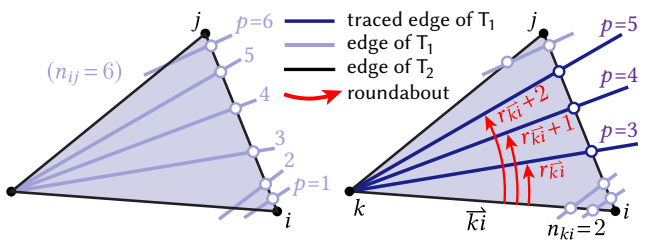


Fig. 20. *Left:* we index crossings along each halfedge ij by an integer p. *Right:* for each halfedge we trace out curves leaving the opposite corner.

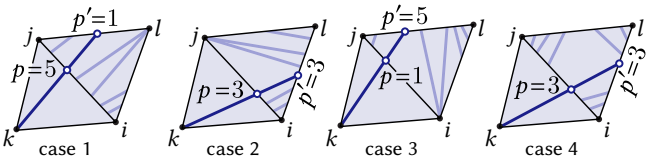
apply a scaling $\tilde{q}_i = e^{u_i} q_i$ to each vertex, we get another chordal triangle; central projection between points x and $\tilde{x}$ then gives the circumcircle-preserving projective map used for interpolation in Springborn et al. [2008] (Figure 19, *center*). The real power of this construction is that it enables us to map between different triangulations of the same vertices. Consider for instance two chordal triangles ijk, jil—if we apply a flip and then scale the vertices q as before, we get two chordal triangles lki, klj sitting "above" the other ones (Figure 19, *right*). The map between these triangle pairs is still given by central projection, and can be expressed as a piecewise projective map on the four triangles of their common subdivision. This idea is extended in the supplement to general pairs of triangulations.

6.1 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 6.1.1). 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 6.2). The final curve is encoded by 1D barycentric coordinates $s, t \in [0, 1]$ on each intersected edge. We enumerate points where edges of T_1 cross a halfedge $\overrightarrow{ij}$ of T_2 by a *crossing index* $p \in \{1, \ldots, n_{ij}\}$ (see Figure 20, *left*).

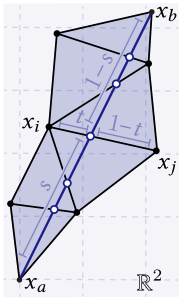
6.1.1 Topological Tracing. To trace out all the edges of T_1 over T_2, we iterate over the halfedges $\overrightarrow{ij} \in \mathsf{H}_2$ and trace edges emanating from the opposite corner k (Figure 20, *right*), namely, the edges with indices $p = 1 + n_{ki}, \ldots, 1 + n_{ki} + e_{\overrightarrow{ij}}$. This procedure is detailed in Algorithm 1. We identify the edge of T_1 corresponding to each traced curve p by incrementing the roundabout $r_{\overrightarrow{ki}}$ by $p - n_{ki} - 1$. (By marking traced edges in T_1, we avoid tracing edges twice.) Note that roundabouts must be used even for curves shared by both triangulations, since after a sequence of edge flips they may no longer correspond to the same logical edge.

The procedure NextEdge (Algorithm 2) uses the normal coordinates n to determine the next halfedge $\overrightarrow{ij}$ and crossing index p along the curve. The image below depicts the four possible cases—this pattern of edge crossings can be determined solely using the normal coordinates for the triangle jil, via the formulas given in Section 5.1. See Appendix A.2 for further details.



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

6.2 Recovering Geodesics

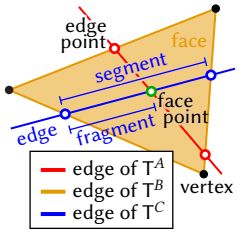


To get the geometry of each traced edge $ab \in \mathsf{E}_2$, we use the crossing sequences computed in Section 6.1 and the edge lengths ℓ to incrementally lay out a triangle strip in the (Euclidean or hyperbolic) plane. We then intersect each interior edge ij of this strip with the line from a to b—by 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

$$(1 - s)x_a + sx_b = (1 - t)x_i + tx_j$$

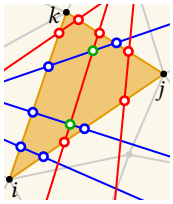
for the barycentric coordinates $s, t \in [0, 1]$ of the intersection point. The hyperbolic case is conceptually the same except that we work in the hyperboloid model, and and also compute a scale factor u at each intersection point—see Appendix A.3 for details.

6.3 Common Refinement

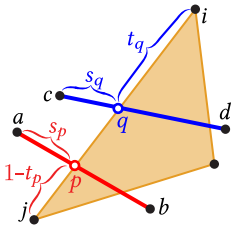


We now construct the common refinement S of T^A, T^B, and T^C. Here and in Appendix B we will refer to points along edges of T^B (resulting from tracing) as *edge points*, and any new vertices inserted into polygons as *face points*, reserving *vertex* for elements of V. Likewise, an *edge* is the complete edge of some triangulation, a *segment* is the restriction of an edge to a triangle, and a *fragment* is a piece of a segment (see inset).

6.3.1 Connectivity. First, for each edge $ij \in \mathsf{E}^B$, we use the procedures from Sections 6.1 and 6.2 to trace out (i) a Euclidean geodesic over T^A to obtain edge sequences and barycentric coordinates (s, t), and (ii) a hyperbolic geodesic over T^C to obtain edge sequences, barycentric coordinates, and scale factors u. 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. [2019b, 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. See Appendix B for details.



6.4 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

$$\hat{z}_q = e^{-u_q}\left((1 - s_q)(z_c, 1) + s_q(z_d, 1)\right), \tag{16}$$

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 via Equation 19. 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).

7 SPHERICAL UNIFORMIZATION

We now consider conformal maps to the sphere S^2. Given a genus-0 Delaunay triangulation $\mathsf{T} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$ with edge lengths $\ell : \mathsf{E}^B \to \mathbb{R}_{>0}$, we give an algorithm that computes vertex positions $f : \mathsf{V} \to S^2 \subset \mathbb{R}^3$ that describe a discretely conformally equivalent convex sphere-inscribed polyhedron. The solution is guaranteed to exist, and is unique up to a Möbius transformation of the sphere.

The strategy used by CETM is essentially to delete the neighborhood of a special vertex i^*, conformally map this modified surface to a flat disk, and apply stereographic projection to the sphere, where the removed vertex i^* is re-inserted. For a fixed triangulation, there are several problems. First, as discussed previously, a discretely conformally equivalent flat disk may not exist. Even if we allow the triangulation to vary, it is not immediately clear what to do about boundary edges (which cannot be flipped). Second, the final polyhedron may not be convex. In fact, many combinatorial triangulations do not admit *any* convex embedding in the sphere—conformal or otherwise [Rivin 1996]. Third, the map may become non-injective when vertex i^* is re-inserted.

Imagine that we instead start with the object we want: a convex sphere-inscribed polyhedron $\mathcal{P}$ conformally equivalent to the input surface. If we stereographically project this polyhedron to the plane through any vertex i^*, we get a planar disk where all boundary vertices j are connected to the same vertex i^* at infinity (Figure 21, *left*). Stereographic projection preserves discrete conformal equivalence with the input, and since the polyhedron is convex, its stereographic projection will be a planar Delaunay triangulation [Brown 1979]—and has hence a convex boundary. Hence, if we can construct such a triangulation, we can obtain the desired spherical conformal map (via stereographic projection).

To solve this problem, Springborn [2019] reformulates it in the hyperbolic setting where one can freely flip edges without invalidating the hyperbolic metric. Here, the Penner coordinates $\lambda_{i^*j} = 2 \log \ell_{i^*j}$ incident on the special vertex i^* are now infinite—effectively pushing the horocycle at i^* off to infinity (Figure 21, *right*). This decorated

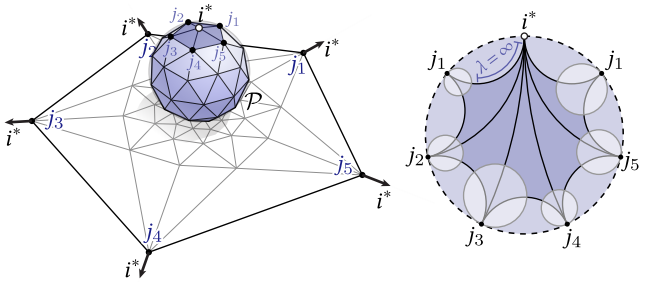
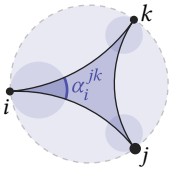


Fig. 21. *Left:* a convex polyhedron inscribed in the sphere can also be viewed, via stereographic projection, as a planar Delaunay triangulation with all boundary vertices connected to a vertex i^* at infinity. *Right:* in the Poincaré model, the horocycle at i^* shrinks to a point, and the incident Penner coordinates λ_{i^*j} go to infinity.

polyhedron can be found via the same uniformization procedure as in Section 4, but with a few important modifications. For one, it uses a modified Delaunay flipping procedure which accounts for edges of infinite length (Section 7.1), and a modified energy which accounts for the boundary vertices j adjacent to i^* (Section 7.2). Linear inequality constraints on u ensure that the edges i^*j are convex and have the right cross ratios (Section 7.3). Solving a bounds-constrained optimization problem (Section 4.3) yields scale factors u that describe the desired planar disk, which we can then stereographically project back onto the sphere.

7.1 Modified Delaunay Flips



Since some Penner coordinates are now infinite (Figure 21, *right*), we can no longer check the Delaunay condition using Equation 10. However, just as the Euclidean Delaunay condition is expressed in terms of angles (Equation 9), we can still express the ideal Delaunay condition in terms of the arc length α_i^{jk} of the horocycle at vertex i within triangle ijk (see inset). Initially, all edge lengths ℓ are well-defined and we have

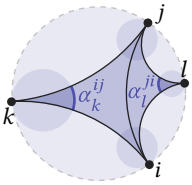
$$\alpha_i^{jk} = \frac{\ell_{jk}}{\ell_{ki}\ell_{ij}}. \tag{17}$$

Scaling lengths *à la* Equation 1 then gives new arc lengths

$$\tilde{\alpha}_i^{jk} = e^{-u_i}\alpha_i^{jk}.$$

At the special vertex i^*, where $u_{i^*} = \infty$, we hence get $\tilde{\alpha}_i^{jk} = 0$ as expected. An edge ij then satisfies the ideal Delaunay condition if

$$\tilde{\alpha}_k^{ji} + \tilde{\alpha}_l^{ij} < \tilde{\alpha}_i^{jk} + \tilde{\alpha}_i^{lj} + \tilde{\alpha}_j^{ik} + \tilde{\alpha}_j^{li}. \tag{18}$$



If this condition is not satisfied, we perform a Ptolemy flip (Equation 8). However, rather than compute $\tilde{\ell}_{kl}$ directly (which may be infinite), we first compute ℓ_{kl} via the Ptolemy relation and then scale to get $\tilde{\ell}_{kl}$. Importantly, if Equation 18 is satisfied with equality for an edge kl opposite the special vertex i^*, we must pick the flip that connects kl to i^*—since for any sequence

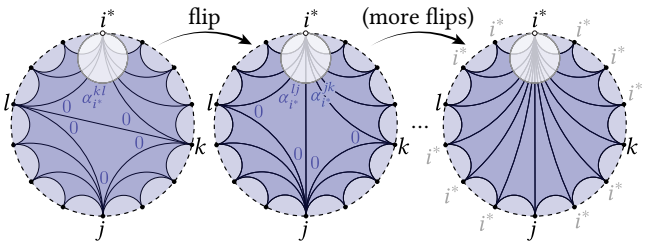


Fig. 23. To find a triangulation connecting the special vertex i^* to all other vertices j, we put a finite horocycle at i^* and send all other horocycles to infinity. Modified Delaunay flips then yield the desired triangulation.

of finite horocycles around i^* approaching infinity, this is the edge that would belong to an ideal Delaunay triangulation.

7.2 Spherical Variational Principle

As in the Euclidean case, the energy and its derivatives are always evaluated on the triangulation $\widetilde{T}$ obtained by updating the Penner coordinates of T^B (*à la* Equation 7) and flipping to an ideal Delaunay triangulation (*à la* Section 7.1). In particular, let $\mathsf{T}^\circ := (\mathsf{V}^\circ, \mathsf{E}^\circ, \mathsf{F}^\circ)$ be the mesh obtained by removing the special vertex i^* together with its incident edges and faces from $\widetilde{\mathsf{T}}$. The energy for spherical uniformization is then

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

(see Springborn [2019, Equation 56 and Theorem 7.18], which differs by a constant that does not affect minimizers). For each vertex $i \in \mathsf{V}^\circ$, its gradient is

$$\partial_{u_j} \mathcal{E}_{S^2} = \widetilde{\Omega}_j + \pi(\deg_{\mathsf{F}^\circ}(j) - \deg_{\mathsf{E}^\circ}(j)),$$

where $\deg_{\mathsf{E}^\circ}(j)$ and $\deg_{\mathsf{F}^\circ}(j)$ are the number of edges and faces of T° containing j, *resp.*; this degree difference will be -1 for vertices adjacent to V° (and zero otherwise). Ω^* does not appear because we do not consider cone singularities in the spherical case. The Hessian is again cotan-Laplace, where cotangents from any removed face are set to zero.

7.3 Constraints

In the fixed triangulation case, Bobenko et al. [2015, Proposition 3.2.1] observe that setting $u_j = -\lambda_{i^*j}$ ensures that the boundary edges i^*j exhibit the right length cross ratio. However, in the variable triangulation case we do not know *a priori* which vertices j will ultimately be adjacent to the removed vertex i^* (since this set may change due to edge flips). Instead, as proposed by Springborn [2019], we impose the inequality constraint $u_j \geq -\lambda_{i^*j}$ for *all* vertices $j \in \mathsf{V}^\circ$, where λ_{i^*j} is the geodesic distance between horocycles in the input triangulation. At a minimizer, these inequalities will be satisfied with equality for vertices j adjacent to i^*.

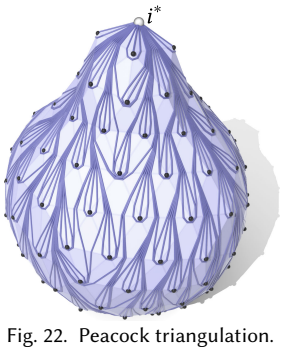


Fig. 22. Peacock triangulation.

To compute the geodesic distances, we first construct a triangulation that connects i^* to all other vertices $j \in \mathsf{V}^\circ$ by minimal geodesics. To do so, we send all the horocycles *except* the one at i^* to infinity—in the Poincaré model, the representative circles shrink down to points (Figure 23, *left*). In general, an edge connecting two vertices $j_1, j_2 \neq i^*$ cannot be Delaunay, since the horocyclic arc length α at both vertices will be zero—hence smaller than the arc length of the complementary vertices (see Figure 23 and Equation 18). By flipping to a Delaunay triangulation, we ensure that any edge leaving a vertex $j \neq i^*$ connects only to i^* (Figure 23, *right*). Moreover one can show that, due to the global Delaunay property, every such edge is a minimal geodesic [Springborn 2019, Proposition 5.16]. All other edges go from i^* back to i^*, resulting in what we call a *peacock triangulation* (Figure 22). To get the values λ_{i^*j}, we then read off the distances between the original horocycles (rather than those that have been scaled down to points).

7.4 Optimization

Once we know λ_{i^*j}, we can solve the convex optimization problem

$$\begin{aligned} \min_{u:\mathsf{V}^\circ \to \mathbb{R}} \quad & \mathcal{E}_{S^2}(u) \\ \text{s.t.} \quad & u_j \geq -\lambda_{i^*j}, \qquad \forall j \in \mathsf{V}^\circ. \end{aligned}$$

This problem can be solved via a bounds-constrained Newton method; see Section 8.1 for further discussion.

7.5 Spherical Layout

After optimization, we have scale factors u at vertices that describe a flat metric on the topological disk T°. We lay this disk out in the plane using the same procedure as described in Section 4.5, then stereographically project to get coordinates z on the unit sphere $S^2 \subset \mathbb{R}^3$ (re-inserting the special vertex i^* at the center of stereographic projection). This final map is unique only up to Möbius transformations of the sphere; we compute a canonical Möbius transformation via Baden et al. [2018, Algorithm 1], using vertex rather than face areas to express the center of mass.

7.6 Spherical Interpolation

Interpolation is done as in Section 6, except we now lift coordinates $z \in \mathbb{R}^3$ to homogeneous coordinates $\hat{z} \in \mathbb{R}^4$, and scale factors must now account for both uniformization and stereographic projection. Let $\tilde{\ell}_{ij}$ be the edge lengths of $\mathcal{P}$, and ℓ_{ij} be the lengths from T^B after applying the same sequence of Ptolemy flips used for uniformization. Then solving Equation 3 within each triangle ijk yields

$$u_i = \log\left(\frac{\tilde{\ell}_{ij}}{\ell_{ij}} \frac{\ell_{jk}}{\tilde{\ell}_{jk}} \frac{\tilde{\ell}_{ki}}{\ell_{ki}}\right)$$

(and similarly for u_j, u_k). Since stereographic projection preserves discrete conformal equivalence, these values agree across triangles. Also, since $\mathcal{P}$ is convex, normalizing interpolated coordinates gives an injective map to the unit sphere (for, *e.g.*, texture mapping).

8 EVALUATION

This section evaluates the empirical behavior of our method, here referred to as *conformal equivalence of polyhedral surfaces (CEPS)*. Our main points of comparison are the CETM algorithm of Springborn

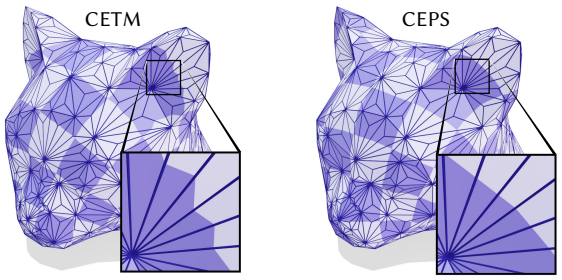


Fig. 24. Even when CETM succeeds, the quality of the map may be lower since it effectively considers a different notion of conformal equivalence (based on the input rather than Delaunay triangulation).

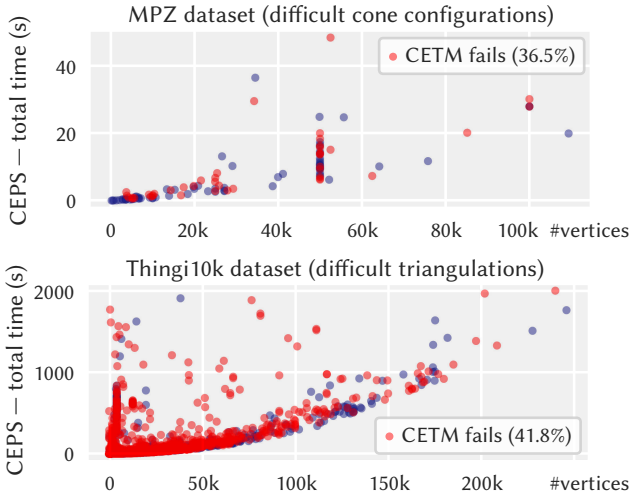


Fig. 25. Timings for our method (CEPS) on two datasets. Note that CETM fails on a large percentage of models where we succeed (highlighted in red).

et al. [2008], which does not use flips, and what we call *uniformization with Euclidean edge flips (UEF)* [Sun et al. 2015], which stops to flip concylic triangle pairs, as described in Section 2.1.2. All methods use identical code for tracking correspondence, *à la* Section 5. We also briefly consider other methods for spherical conformal mapping (Section 8.3.3) and non-conformal injective mapping (Section 8.3.1).

The overall observation is that CEPS succeeds on far more models than CETM, and exhibits better scaling than UEF. Even when CETM does succeed, it may not provide as good of an approximation of a smooth conformal map (Figure 24). Moreover, our UEF implementation is more efficient than the one suggested by Sun et al. [2015] since it uses the implicit tracking scheme from Section 5 (which depends critically on the hyperbolic perspective), rather than an explicit overlay *à la* Fisher et al. [2007].

8.1 Implementation

Algorithms were implemented in double precision in C++ using the halfedge implementation in *GeometryCentral* [Sharp et al. 2019a]. For cone flattening, we used Newton's method with backtracking line search [Boyd and Vandenberghe 2004, Algorithms 9.2 and 9.5],

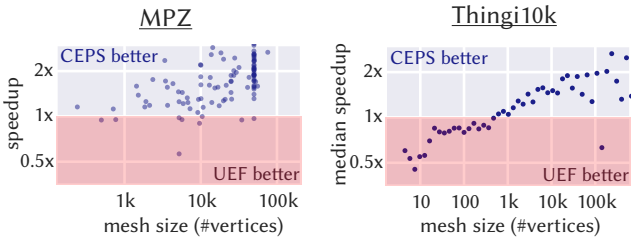


Fig. 26. Speedup of our method over UEF (total wall-clock time for computing scale factors). Beyond about 1k vertices, CEPS is typically faster.

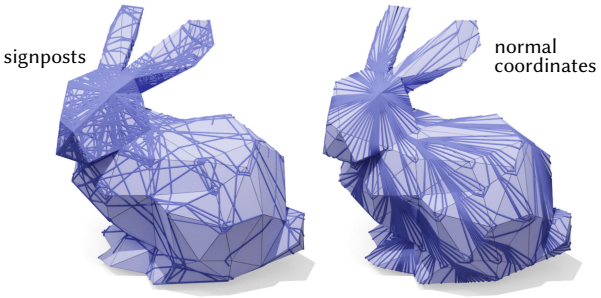


Fig. 27. *Left:* The signpost data structure fails to be numerically robust in extreme situations, such as when tracing the "peacock triangulation." *Right:* our integer-based encoding ensures we get the right connectivity.

using CHOLMOD to solve linear systems [Chen et al. 2008]. For spherical uniformization, we used bounds-constrained Newton's method with backtracking line search [Balay et al. 2019, 1997; Munson et al. 2014]. In practice, we use the implementation found in the *PETsc/TAO* library—specifically, we use the `TAOBNLS` solver [Benson et al. 2003, Section 4.2.1]. Timings were measured on an Intel i9-9980XE with 32 GB of RAM, using a single thread.

8.2 Numerics

Our algorithms are guaranteed to work in exact arithmetic for any valid input—in a real implementation we must also be careful about floating point error. Here we describe several useful techniques, though of course other improvements may be possible. Note that we mollify input edge lengths as described by Sharp and Crane [2020] (using $\delta = 10^{-12}$) which helps with near-degenerate models and otherwise leaves the input untouched.

8.2.1 Euclidean Uniformization. One way to evaluate the intrinsic Delaunay condition (Equation 9) is to use the angle cotangents; Fisher et al. [2007] provide details. We instead check the hyperbolic condition (Equation 10)—even when constructing the Euclidean Delaunay triangulation—since it yields the same triangulation, but only involves rational arithmetic on edge lengths. When triangles are nearly concyclic, this condition may be violated (or satisfied) both before *and* after the edge flip, due to floating point error. Hence, we check Equation 10 for all edges, and perform a flip only if it increases the difference between right- and left-hand side.

8.2.2 Spherical Uniformization. For most models the choice of special vertex i^* makes no difference, but on models with long, thin

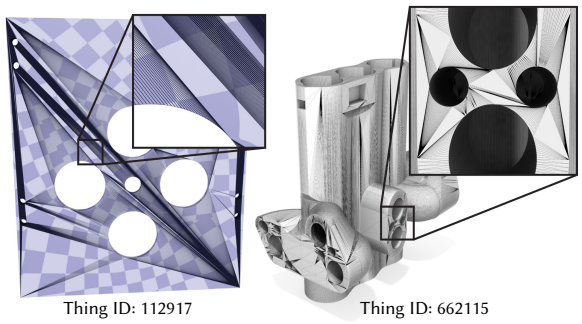


Fig. 28. Our implementation robustly handles extremely poor triangulations (left) failing only on the most pathological inputs (right). See Figure 1 for more examples.

features a useful heuristic is to put the vertex i^* at the *intrinsic median* Sharp et al. [2019c, 8.3]; in practice we use the `findCenter` method from *GeometryCentral* [Sharp et al. 2019a]. When constructing the peacock triangulation in Section 7.3, values of ℓ can become large enough to result in floating point overflow. We therefore work instead with the values $\lambda = 2\log\ell$, which occupy a much smaller range. In particular, to compute the values α_i^{jk}, we evaluate the log of Equation 17: $\log\alpha_i^{jk} = \frac{1}{2}(\lambda_{jk} - \lambda_{ki} - \lambda_{ij})$, then exponentiate, and multiply by the scale factor e^{-u_i} (which equals zero for all vertices but i^*). As before, Equation 18 gives the Delaunay condition in terms of α. We also use the *log-sum-exp trick* [Blanchard et al. 2019, Equation 1.3] to help with numerical precision when applying the Ptolemy relation $\ell_{kl} = (\ell_{ki}\ell_{lj} + \ell_{jk}\ell_{li})/\ell_{ij}$. This means we write the log of the numerator as

$$\tfrac{1}{2}\lambda_{ki} + \tfrac{1}{2}\lambda_{lj} + \log(1 + e^{(\lambda_{jk}/2+\lambda_{li}/2)-(\lambda_{ki}/2+\lambda_{lj}/2)}),$$

where, without loss of generality, we label the vertices so that $\lambda_{ki} + \lambda_{lj} > \lambda_{jk} + \lambda_{li}$. We then subtract $\frac{1}{2}\lambda_{ij}$ to get the final log length for the new edge kl. (When building the peacock triangulation, we never explicitly compute the values of ℓ.)

Layout. One could lay out triangles incrementally, as in [Springborn et al. 2008, Section 3.3]. We found it more robust to use the spectral layout of Mullen et al. [2008], which we use in all examples. This algorithm requires only the cotan-Laplace and mass matrices, which can be built directly from the final edge lengths ℓ^C. Since lengths describe a flat metric, spectral layout incurs no further distortion.

Mapping. When laying out triangle strips (Section 6.2), we found that it improves floating point robustness to first incrementally compute the angles for each halfedge, and then use these angles to recover final vertex positions $x \in \mathbb{R}^2$. In meshes with near-degenerate triangles, we find that the hyperbolic layout procedure can sometimes fail to place points on the light cone due to floating-point errors. In particular, the new vertex coordinates q_l might be at the origin, or contain NaNs. In this case, we replace the global strip layout with a local iterative straightening procedure (akin to nonlinear Gauss-Seidel). In particular, we consider two consecutive triangles at a time and update the location where the geodesic intersects the common edge—see supplement for details.

8.3 Experiments

8.3.1 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 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 25, *top*) and was less than 2x faster (Figure 29, *top*). Moreover, the tracing and refinement steps of CEPS could be trivially parallelized over edges and faces, *resp.* UEF also succeeds on these models, but is generally slower than CEPS (Figure 26, *left*).

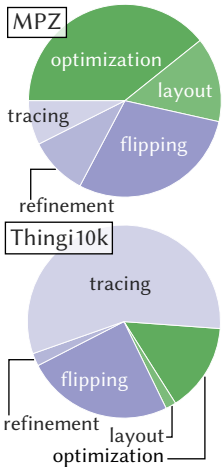


Fig. 29. Average breakdown of costs in CEPS; green tasks are shared by CETM.

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 and 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.

8.3.2 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 28, *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

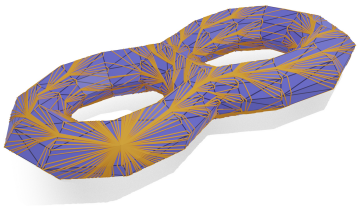


Fig. 30. Since we allow edge flips, we need not worry how coarse the mesh is near large cones. Here we set all but one angle defect to almost 2π—the remaining vertex has an angle defect of -1032.79.

Fig. 32. In the genus-0 case, our method guarantees a bijective discrete conformal map to a convex polyhedron with vertices on the sphere.

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 25), and performance of UEF suffers as models get larger (Figure 26, *right*).

8.3.3 Spherical Conformal Parameterization. We ran our spherical algorithm on two other datasets: the *Spherical Demon* brain scan dataset of Yeo et al. [2009], and the anatomical surface dataset of Boyer et al. [2011] (Figure 32). On the brain dataset, where each model has about 230k faces, we obtained injective discrete conformal maps to the sphere on all 78 brain hemispheres, taking an average of 493 seconds per hemisphere. The anatomical surface models are topological disks, so we compute conformal maps to a hemisphere. We succeed in computing these maps on all 187 of the manifold meshes without handles in the dataset. One of our maps contains degenerate triangles, but none have flipped triangles. The models take an average of 14.4 seconds to uniformize.

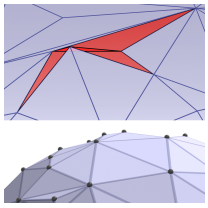


Fig. 31. Past spherical methods often exhibit foldover, and cannot guarantee convexity.

Past methods for spherical conformal parameterization largely compute maps to the sphere that are harmonic with respect to piecewise linear Dirichlet energy [Gu et al. 2004]. However, unless the input mesh is already Delaunay, such maps can have flipped triangles (Figure 31, *right*). More fundamentally, it is impossible for any method that uses a fixed triangulation to guarantee convexity—no matter what algorithm is used, or where the vertices are placed—since not all combinatorial triangulations admit a convex embedding in the sphere [Rivin 1996]. In practice, flipped triangles and nonconvex edges are quite common in discrete harmonic maps: on the brain dataset we observed, on average, foldover at about 100 edges and nonconvexity at about 20k edges when using the method of Kazhdan et al. [2012] (see inset). Other techniques for spherical conformal mapping gave very similar results [Crane et al. 2013b; Dym et al. 2019].

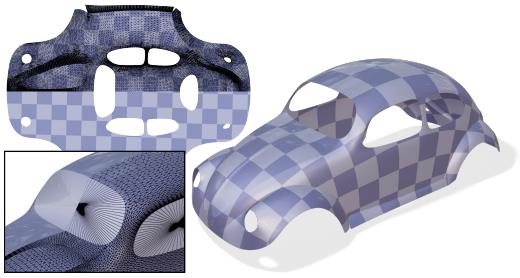


Fig. 33. We can handle multiply-connected domains by simply triangulating holes prior to flattening, then removing these triangles after flattening.

8.4 Multiply-Connected Surfaces

Many surfaces encountered in practice will have multiple boundary components. Though uniformization can be used to find a flat metric on such surfaces, this metric cannot always be laid out in the plane without additional cuts, due to nontrivial monodromy around boundary cycles (see for instance Hefetz et al. [2019, Figure 6]). Extension of discrete uniformization to multiply-connected *circle domains* and *slit domains* has been considered by Bobenko et al. [2016], but there is an even simpler alternative appropriate for practical texture mapping: just triangulate each of the holes, then remove these triangles after parameterization. Figure 33 shows one such example. A natural question is how to more directly control boundary behavior by setting the intrinsic lengths of inserted edges.

9 LIMITATIONS AND FUTURE WORK

The ability to modify the input triangulation is ultimately what enables one to establish a discrete uniformization theorem where existence is guaranteed. From a practical point of view, it comes at a small cost: the output mesh has different connectivity than the input. Importantly, however, the common refinement S can be stored as a standard mesh with ordinary vertex and texture coordinates that can be used downstream. Our implementation also outputs a sparse $|V^S| \times |V^A|$ matrix that interpolates values from the input mesh to the larger set of vertices in S. The refinement is around 1.5–3x bigger than the input on common models (*e.g.*, those in MPZ), and around 5x on most of Thingi10k—though on pathological models even initial Delaunay flipping can increase size by 45x or more. To reduce the final mesh size, it may be possible to "un-flip" many edges of the planar layout (*à la* Kharevych et al. [2006, Section 5]), which incurs some conformal distortion but preserves injectivity. If one cares purely about injectivity, the initial Delaunay retriangulation step can also be skipped. The method is guaranteed to work only in exact arithmetic—it is natural to consider numerical improvements, or ways to further reduce dependence on floating-point arithmetic (*e.g.*, during mesh extraction). We did not consider uniformization over hyperbolic domains, though this case is well-supported by the same theory—see Springborn [2019] for further discussion.

ACKNOWLEDGMENTS

Thanks to Saul Schleimer for useful discussions about normal coordinates. This work was supported by a Packard Fellowship, NSF Award 1717320, DFG TRR 109, an NSF Graduate Research Fellowship, and gifts from Autodesk, Adobe, and Facebook.

REFERENCES

William Abikoff. 1981. The Uniformization Theorem. *The American Mathematical Monthly* 88, 8 (1981).

Ian Agol, Joel Hass, and William Thurston. 2006. The Computational Complexity of Knot Genus and Spanning Area. *Trans. Amer. Math. Soc.* 358, 9 (2006).

Noam Aigerman and Yaron Lipman. 2016. Hyperbolic Orbifold Tutte Embeddings. *ACM Transactions on Graphics* 35, 6 (2016).

Noam Aigerman, Roi Poranne, and Yaron Lipman. 2014. Lifted Bijections for Low Distortion Surface Mappings. *ACM Transactions on Graphics* 33, 4 (2014).

DV Alekseevskij, Eh B Vinberg, and AS Solodovnikov. 1993. Geometry of Spaces of Constant Curvature. In *Geometry II*. Springer.

Alex Baden, Keenan Crane, and Misha Kazhdan. 2018. Möbius Registration. *Computer Graphics Forum* 37, 5 (2018).

Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dmitry Karpeyev, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. 2019. *PETSc Users Manual*. Technical Report ANL-95/11 - Revision 3.11. Argonne National Laboratory.

Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. 1997. Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. In *Modern Software Tools in Scientific Computing*, E. Arge, A. M. Bruaset, and H. P. Langtangen (Eds.). Birkhäuser Press.

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

Steve Benson, Lois Curfman McInnes, Jorge J More, and Jason Sarich. 2003. *TAO Users Manual*. Technical Report. Argonne National Lab., IL (US).

Pierre Blanchard, Desmond J Higham, and Nicholas J Higham. 2019. Accurate Computation of the Log-Sum-Exp and Softmax Functions. *arXiv preprint* (2019).

Alexander I Bobenko, Ulrich Pinkall, and Boris A Springborn. 2015. Discrete Conformal Maps and Ideal Hyperbolic Polyhedra. *Geometry & Topology* 19, 4 (2015).

Alexander I Bobenko, Stefan Sechelmann, and Boris Springborn. 2016. Discrete Conformal Maps: Boundary Value Problems, Circle Domains, Fuchsian and Schottky Uniformization. In *Advances in Discrete Differential Geometry*. Springer.

Alexander I Bobenko and Boris A Springborn. 2007. A Discrete Laplace–Beltrami Operator for Simplicial Surfaces. *Disc. & Comp. Geom.* 38, 4 (2007).

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

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge university press.

Doug M Boyer, Yaron Lipman, Elizabeth St Clair, Jesus Puente, Biren A Patel, Thomas Funkhouser, Jukka Jernvall, and Ingrid Daubechies. 2011. Algorithms to Automatically Quantify the Geometric Similarity of Anatomical Surfaces. *Proceedings of the National Academy of Sciences* 108, 45 (2011).

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

Kevin Q Brown. 1979. Voronoi Diagrams from Convex Hulls. *Information processing letters* 9, 5 (1979).

Ulrike Bücking. 2016. Approximation of Conformal Mappings Using Conformally Equivalent Triangular Lattices. In *Adv. in Disc. Diff. Geom.* Springer.

Ulrike Bücking. 2018. C^∞-Convergence of Conformal Mappings for Conformally Equivalent Triangular Lattices. *Results in Mathematics* 73, 2 (2018).

Marcel Campen, Ryan Capouellez, Hanxiao Shen, Leyi Zhu, Daniele Panozzo, and Denis Zorin. 2021. Efficient and Robust Discrete Conformal Equivalence with Boundary. (2021). arXiv:cs.CG/2104.04614

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

Marcel Campen and Denis Zorin. 2017a. On Discrete Conformal Seamless Similarity Maps. *arXiv preprint* (2017).

Marcel Campen and Denis Zorin. 2017b. Similarity Maps and Field-Guided T-Splines: A Perfect Couple. *ACM Transactions on Graphics* 36, 4 (2017).

James W Cannon, William J Floyd, Richard Kenyon, Walter R Parry, et al. 1997. Hyperbolic Geometry. *Flavors of geometry* 31 (1997).

Pafnutiĭ L'vovich Chebyshev. 1899. *Œuvres de P.L. Tchebychef*. Vol. 1. Commissionaires de l'Académie Impériale des Sciences.

Wei Chen, Min Zhang, Na Lei, and Xianfeng David Gu. 2016. Dynamic Unified Surface Ricci Flow. *Geom., Img. and Com.* 3, 1 (2016).

Yanqing Chen, Timothy A Davis, William W Hager, and Sivasankaran Rajamanickam. 2008. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Transactions on Mathematical Software (TOMS)* 35, 3 (2008).

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

Keenan Crane. 2020. An Excursion through Discrete Differential Geometry. Proceedings of Symposia in Applied Mathematics, Vol. 76. American Mathematical Society, Chapter Conformal Geometry of Simplicial Surfaces.

Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. 2013a. Digital Geometry Processing with Discrete Exterior Calculus. In *ACM SIGGRAPH 2013 Courses (SIGGRAPH '13)*. ACM, New York, NY, USA, Article 7.

Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013b. Robust Fairing via Conformal Curvature Flow. *ACM Trans. Graph.* 32, 4 (2013).

Mathieu Desbrun, Mark Meyer, and Pierre Alliez. 2002. Intrinsic Parameterizations of Surface Meshes. In *Computer Graphics Forum*, Vol. 21.

Nadav Dym, Raz Slutsky, and Yaron Lipman. 2019. Linear Variational Principle for Riemann Mappings and Discrete Conformality. *Proceedings of the National Academy of Sciences* 116, 3 (2019).

Jeff Erickson and Amir Nayyeri. 2013. Tracing Compressed Curves in Triangulated Surfaces. *Discrete & Computational Geometry* 49, 4 (2013).

François Fillastre. 2008. Polyhedral Hyperbolic Metrics on Surfaces. *Geometriae Dedicata* 134, 1 (2008).

Matthew Fisher, Boris Springborn, Peter Schröder, and Alexander I Bobenko. 2007. An Algorithm for the Construction of Intrinsic Delaunay Triangulations with Applications to Digital Geometry Processing. *Computing* 81, 2-3 (2007).

Mark Galassi, Jim Davies, James Theiler, Brian Gough, Gerard Jungman, Patrick Alken, Michael Booth, Fabrice Rossi, and Rhys Ulerich. 1994. *GNU Scientific Library*. Vol. 20. ACM New York, NY, USA.

Xianfeng David Gu, Feng Luo, Jian Sun, and Tianqi Wu. 2018a. A Discrete Uniformization Theorem for Polyhedral Surfaces. *Journal of Differential Geometry* 109, 2 (2018).

Xianfeng David Gu, Ren Guo, Feng Luo, Jian Sun, and Tianqi Wu. 2018b. A Discrete Uniformization Theorem for Polyhedral Surfaces II. *Journal of Differential Geometry* 109, 3 (2018).

Xianfeng David Gu, Feng Luo, and Tianqi Wu. 2019. Convergence of Discrete Conformal Geometry and Computation of Uniformization Maps. *Asian Journal of Mathematics* 23, 1 (2019).

Xianfeng David Gu, Feng Luo, and Shing-Tung Yau. 2020. Computational Conformal Geometry behind Modern Technologies. *Notices of the American Mathematical Society* 67, 10 (2020).

Xianfeng David Gu, Yalin Wang, Tony F Chan, Paul M Thompson, and Shing-Tung Yau. 2004. Genus Zero Surface Conformal Mapping and Its Application to Brain Surface Mapping. *IEEE transactions on medical imaging* 23, 8 (2004).

Wolfgang Haken. 1961. Theorie Der Normalflächen: Ein Isotopiekriterium Für Den Kreisknoten. *Acta Mathematica* 105, 3-4 (1961).

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

Eden Fedida Hefetz, Edward Chien, and Ofir Weber. 2019. A Subspace Method for Fast Locally Injective Harmonic Mapping. In *Computer Graphics Forum*, Vol. 38.

David Hilbert. 1901. Ueber Flächen von Constanter Gaussscher Krümmung. *Trans. Amer. Math. Soc.* 2, 1 (1901).

Claude Indermitte, Th M Liebling, Marc Troyanov, and Heinz Clémençon. 2001. Voronoi Diagrams on Piecewise Flat Surfaces and an Application to Biological Growth. *Theoretical Computer Science* 263, ARTICLE (2001).

Miao Jin, Yalin Wang, Shing-Tung Yau, and Xianfeng David Gu. 2004. Optimal Global Conformal Surface Parameterization. In *IEEE Visualization 2004.*

Michael Kazhdan, Jake Solomon, and Mirela Ben-Chen. 2012. Can Mean-Curvature Flow Be Modified to Be Non-Singular?. In *Computer Graphics Forum*, Vol. 31.

Liliya Kharevych, Boris Springborn, and Peter Schröder. 2006. Discrete Conformal Mappings via Circle Patterns. *ACM Transactions on Graphics* 25, 2 (2006).

Hellmuth Kneser. 1929. Geschlossene Flächen in Dreidimensionalen Mannigfaltigkeiten. *Jahresbericht der Deutschen Mathematiker-Vereinigung* 38 (1929).

Patrice Koehl and Joel Hass. 2015. Landmark-Free Geometric Methods in Biological Shape Analysis. *Journal of The Royal Society Interface* 12, 113 (2015).

Mina Konaković, Keenan Crane, Bailin Deng, Sofien Bouaziz, Daniel Piker, and Mark Pauly. 2016. Beyond Developable: Computational Design and Fabrication with Auxetic Materials. *ACM Trans. Graph.* 35, 4 (2016).

Zohar Levi and Denis Zorin. 2014. Strict Minimizers for Geometric Optimization. *ACM Transactions on Graphics* 33, 6 (2014).

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least Squares Conformal Maps for Automatic Texture Atlas Generation. *ACM Transactions on Graphics* 21, 3 (2002).

Yaron Lipman. 2012. Bounded Distortion Mapping Spaces for Triangular Meshes. *ACM Transactions on Graphics* 31, 4 (2012).

Yaron Lipman and Ingrid Daubechies. 2011. Conformal Wasserstein Distances: Comparing Surfaces in Polynomial Time. *Advances in Mathematics* 227, 3 (2011).

Feng Luo. 2004. Combinatorial Yamabe Flow on Surfaces. *Communications in Contemporary Mathematics* 6, 5 (2004).

Richard H MacNeal. 1949. *The Solution of Partial Differential Equations by Means of Electrical Networks*. Ph.D. Dissertation. California Institute of Technology.

Manish Mandad and Marcel Campen. 2019. Exact Constraint Satisfaction for Truly Seamless Parametrization. In *Computer Graphics Forum*, Vol. 38.

Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. 2017. Convolutional Neural Networks on Surfaces via Seamless Toric Covers. *ACM Trans. Graph.* 36, 4 (2017).

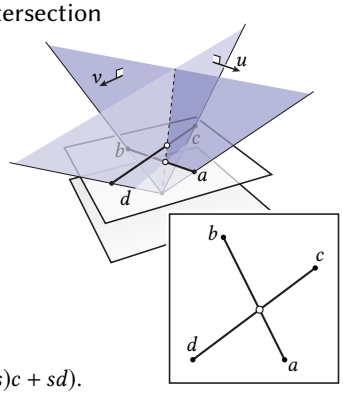
Lee Mosher. 1988. Tiling the Projective Foliation Space of a Punctured Surface. *Trans. Amer. Math. Soc.* (1988).
Patrick Mullen, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. 2008. Spectral Conformal Parameterization. In *Computer Graphics Forum*, Vol. 27.
Todd Munson, Jason Sarich, Stefan Wild, Steve Benson, and Lois Curfman McInnes. 2014. *Toolkit for Advanced Optimization (TAO) Users Manual.* Technical Report ANL/MCS-TM-322 - Revision 3.5. Argonne National Laboratory.
Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust Field-Aligned Global Parametrization. *ACM Transactions on Graphics* 33, 4 (2014).
Robert C. Penner. 2012. *Decorated Teichmüller Theory.* European Mathematical Society (EMS), Zürich.
Roman Prosanov. 2020. Ideal Polyhedral Surfaces in Fuchsian Manifolds. *Geometriae Dedicata* 206 (2020).
Igor Rivin. 1994. Intrinsic Geometry of Convex Ideal Polyhedra in Hyperbolic 3-Space. In *Analysis, Algebra, and Computers in Mathematical Research (Luleå, 1992).* Lect. Notes Pure Appl. Math., Vol. 156.
Igor Rivin. 1996. A Characterization of Ideal Polyhedra in Hyperbolic 3-Space. *Annals of Mathematics. Second Series* 143, 1 (1996).
M. Roček and R. M. Williams. 1984. The Quantization of Regge Calculus. *Zeitschrift für Physik C Particles and Fields* 21, 4 (1984).
Rohan Sawhney and Keenan Crane. 2017. Boundary First Flattening. *ACM Transactions on Graphics* 37, 1 (2017).
Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. 2002. Algorithms for Normal Curves and Surfaces. In *International Computing and Combinatorics Conference.*
Nick Sharp and Keenan Crane. 2018. Variational Surface Cutting. *ACM Trans. Graph.* 37, 4 (2018).
Nicholas Sharp and Keenan Crane. 2020. A Laplacian for Nonmanifold Triangle Meshes. In *Computer Graphics Forum*, Vol. 39.
Nicholas Sharp, Keenan Crane, et al. 2019a. Geometry-Central. (2019).
Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019b. Navigating Intrinsic Triangulations. *ACM Trans. Graph.* 38, 4 (2019).
Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019c. The Vector Heat Method. *ACM Trans. Graph.* 38, 3 (2019).
Alla Sheffer, Bruno Lévy, Maxim Mogilnitsky, and Alexander Bogomyakov. 2005. ABF++: Fast and Robust Angle Based Flattening. *ACM Transactions on Graphics* 24, 2 (2005).
Hanxiao Shen, Zhongshi Jiang, Denis Zorin, and Daniele Panozzo. 2019. Progressive Embedding. *ACM Transactions on Graphics* 38, 4 (2019).
Yousuf Soliman, Dejan Slepčev, and Keenan Crane. 2018. Optimal Cone Singularities for Conformal Flattening. *ACM Trans. Graph.* 37, 4 (2018).
Boris Springborn. 2019. Ideal Hyperbolic Polyhedra and Discrete Uniformization. *Discrete & Computational Geometry* (2019).
Boris Springborn, Peter Schröder, and Ulrich Pinkall. 2008. Conformal Equivalence of Triangle Meshes. In *ACM SIGGRAPH 2008 Papers.*
Jian Sun, Tianqi Wu, Xianfeng David Gu, and Feng Luo. 2015. Discrete Conformal Deformation: Algorithm and Experiments. *SIAM Journal on Imaging Sciences* 8, 3 (2015).
Dylan Thurston and Qiaochu Yuan. 2012. Notes on Curves on Surfaces. (2012).
Marc Troyanov. 1991. Prescribing Curvature on Compact Surfaces with Conical Singularities. *Trans. Amer. Math. Soc.* 324, 2 (1991).
Ana Maria Vintescu, Florent Dupont, and Guillaume Lavoué. 2017. Least Squares Affine Transitions for Global Parameterization. (2017).
Tianqi Wu. 2014. *Finiteness of Switches in Discrete Yamabe Flow.* Ph.D. Dissertation. Tsinghua University.
BT Thomas Yeo, Mert R Sabuncu, Tom Vercauteren, Nicholas Ayache, Bruce Fischl, and Polina Golland. 2009. Spherical Demons: Fast Diffeomorphic Landmark-Free Surface Registration. *IEEE transactions on medical imaging* 29, 3 (2009).
Xiaokang Yu, Na Lei, Yalin Wang, and Xianfeng David Gu. 2017. Intrinsic 3D Dynamic Surface Tracking Based on Dynamic Ricci Flow and Teichmuller Map. In *IEEE Vis.*
Min Zhang, Ren Guo, Wei Zeng, Feng Luo, Shing-Tung Yau, and Xianfeng David Gu. 2014. The Unified Discrete Surface Ricci Flow. *Graphical Models* 76, 5 (2014).
Jiaran Zhou, Changhe Tu, Denis Zorin, and Marcel Campen. 2020. Combinatorial Construction of Seamless Parameter Domains. In *Computer Graphics Forum*, Vol. 39.

A EXTRACTING GEODESICS

Here we give a detailed description of how to extract geodesics from normal coordinates. Note that in this appendix, and in the supplement, we use $\langle u, v\rangle := u_1v_1 + u_2v_2 + u_3v_3$ to denote the Euclidean inner product for vectors $u, v \in \mathbb{R}^3$, and $\langle u, v\rangle_{2,1} := u_1v_1 + u_2v_2 - u_3v_3$ for the Lorentz inner product.

A.1 Projective Segment Intersection

For several of our calculations, it will be useful to express the intersection between two planar line segments ab and cd in terms of the homogeneous coordinates of their endpoints. In particular, if $a, b, c, d \in \mathbb{R}^3$ are any homogeneous coordinates for the endpoints, we seek a solution to



$$(1-t)a + tb = e^u((1-s)c + sd).$$

Letting $v := a \times b$ and $w := c \times d$, we can write the solution explicitly as

$$t = \frac{\langle w, a\rangle}{\langle w, a-b\rangle}, \quad s = \frac{\langle v, c\rangle}{\langle v, c-d\rangle}, \quad u = \log\left(\frac{\langle v, d-c\rangle}{\langle w, a-b\rangle}\right). \tag{19}$$

A.2 Tracing

We here detail the algorithms for topological tracing, described in Section 6.1.1. In particular, Algorithm 2 considers the four cases depicted in Section 6.1.1. In case 1, several curves end at vertex l. Here there are three possibilities: the curve either continues through il, terminates at l, or continues through lj. In case 2, several curves end at j. This time, the curve must continue through ij. Similarly, in case 3, several curves end at i, and the curve must continue through $\overrightarrow{lj}$. In case 4, no curve ends at any vertex of ilj, and the curve continues through either $\overrightarrow{il}$ or $\overrightarrow{lj}$.

Algorithm 1 TraceEdge(n, $\overrightarrow{ij}$, p)

Input: Normal coordinates $n : E_2 \to \mathbb{Z}_{\geq 0}$, a triangle corner specified via the opposite halfedge $\overrightarrow{ij}$, and the index p of the curve to be traced.
Output: A list of pairs $\gamma = ((\overrightarrow{ij}_0, p_0), \ldots, (\overrightarrow{ij}_n, p_n))$ specifying how the traced curve crosses T_2.

1: $\gamma = ()$ ▷ *initialize list of crossings*
2: **do**
3: Append(γ, $(\overrightarrow{ij}, p)$)
4: $(\overrightarrow{ij}, p) \leftarrow$ NextEdge(n, $\overrightarrow{ij}$, p)
5: **while** $p \neq 0$ ▷ *not yet at a vertex*
6: return γ

Algorithm 2 NextEdge(n, $\overrightarrow{ij}$, p)

Input: Normal coordinates n on T_2, a halfedge $\overrightarrow{ij} \in H_2$, and an index p.

Output: The next halfedge $\overrightarrow{ij}'$ and index p' along the curve.

$\overrightarrow{ji} \leftarrow \text{Twin}(\overrightarrow{ij})$
if $n_{ij} > n_{jl} + n_{il}$ **then** ▷ *Case 1*
  if $p \le n_{il}$ **then** return $(\overrightarrow{il}, p)$
  else if $n_{il} < p \le n_{ij} + -n_{jl}$ **then** return $(l, 0)$
  else return $(\overrightarrow{lj}, p - (n_{ij} - n_{jl}))$
else if $n_{lj} > n_{ij} + n_{il}$ **then** return $(\overrightarrow{lj}, p + n_{jl} - n_{ij})$ ▷ *Case 2*
else if $n_{il} > n_{ij} + n_{lj}$ **then** return $(\overrightarrow{il}, p)$ ▷ *Case 3*
else ▷ *Case 4*
  $c_l^{ij} = (n_{lj} + n_{il} - n_{ij})/2$
  if $p \le n_{il} - c_l^{ij}$ **then** return $(\overrightarrow{il}, p)$
  else
    $c_i^{lj} \leftarrow (n_{il} + n_{ij} - n_{lj})/2$
    return $(\overrightarrow{lj}, p - c_i^{lj} + c_l^{ij})$

A.3 Hyperbolic Geodesics

We here describe how to extract the points where a hyperbolic geodesic γ_{ab} intersects the sequence of halfedges computed via Algorithm 1. For each such halfedge $\overrightarrow{ij}$ we extract the barycentric coordinates s, t along γ_{ab} and $\overrightarrow{ij}$, *resp.*, plus a log scale factor u associated with the intersection. As in Section 6.2 we lay out a triangle strip, but this time place vertices on the light cone $\mathcal{L}$ (Section 3.3.1). As derived in the supplement, the first triangle aij has vertices

$$\begin{aligned} q_a &= \tfrac{2}{\sqrt{3}} \ell_{ai} \ell_{aj} \ell_{ij}^{-1} (1, 0, 1), \\ q_i &= \tfrac{2}{\sqrt{3}} \ell_{ai} \ell_{ij} \ell_{aj}^{-1} (\cos(2\pi/3), \sin(2\pi/3), 1), \\ q_j &= \tfrac{2}{\sqrt{3}} \ell_{aj} \ell_{ij} \ell_{ai}^{-1} (\cos(4\pi/3), \sin(4\pi/3), 1). \end{aligned}$$

For any triangle kjl following a known triangle ijk, we use the Ptolemy relation to get ℓ_{il}, then solve for the unknown position

$$q_l = \frac{\ell_{il}}{\ell_{ik}\ell_{ij}} \left(-\frac{\ell_{jl}\ell_{kl}}{\ell_{il}} q_i + \frac{\ell_{ik}\ell_{kl}}{\ell_{jk}} q_j + \frac{\ell_{jl}\ell_{ij}}{\ell_{jk}} q_k \right).$$

This process repeats until we have laid out the whole strip, including the endpoints q_a and q_b—to account for uniformization, we scale just these endpoints by e^{-u_a} and e^{-u_b}, *resp.* If we then imagine that points $q \in \mathbb{R}^3$ in the light cone are homogeneous coordinates for points $x \in \mathbb{R}^2$ from the Beltrami-Klein model, then Equation 19 gives us the desired values s, t, and u at each intersection.

B REFINING FACES

We here describe how to build a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ describing the connectivity of S within a face $ijk \in \mathsf{T}^B$. Recall the nomenclature defined in Section 6.3. In Step I below we determine which edge points connect to form segments; in Step II we determine which segments intersect; in Step III we extract faces of $\mathcal{G}$. We first sort all edge points p in counter-clockwise order (starting at any edge point), assigning them indices $\varphi_p \in \mathbb{Z}$ which provide a sort of combinatorial analogue to the angle around the boundary.

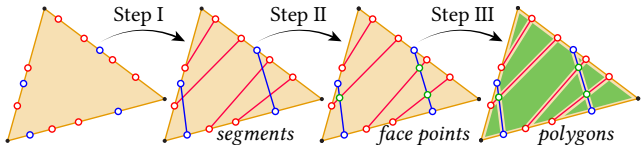
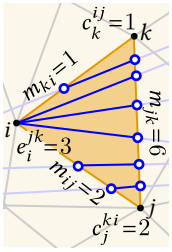
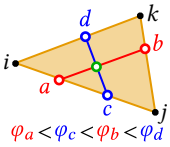


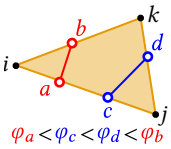
Fig. 34. The three stages of connectivity extraction.



Step I (Segments). We compute the segments from T^A and T^C independently, *à la* Sharp et al. [2019b, Section 3.4]. In each case, the number of edge points determine three normal coordinates m_{ij}; e_i^{jk} and c_i^{jk} are then defined as in Section 5.1. If $e_i^{jk} \neq 0$, we connect the first c_j^{ki} edge points along jk to the first c_j^{ki} edge points along ji, the next e_i^{jk} edge points to vertex i, and the remaining ones to the edge points along ki (see inset).



Step II (Segment-Segment Intersections). To find face points, we consider all pairs of segments ab and cd from T^A and T^C, *resp.* Two segments cross if the values of φ are interleaved, *i.e.*, if they come in a (cyclic) order like $\varphi_a, \varphi_c, \varphi_b, \varphi_d$. If so, we add a new face point, maintaining a list of all face points along each segment, sorted by ϕ (which defines the fragments). After computing intersections, we use the ϕ values at segment endpoints to sort the fragments around each face point.



Step III (Polygons). To extract the final polygons, we iterate over fragments. For each fragment we visit the next vertex, then the next fragment in counter-clockwise order, until we return to the original fragment. If desired, the final (global) tessellation S can be collected in an ordinary mesh whose vertices consist of all face points, edge points, and vertices from the original vertex set V.

Note that the sorting procedure in Step II is the only moment where floating point errors have any chance of invalidating the connectivity of the refinement.

Supplemental Material

This supplement provides additional pseudocode for the main uniformization algorithm, details for improving numerical robustness, and derivations of formulas.

C PSEUDOCODE

Below we give complete pseudocode for the uniformization procedure described in Section 4; note that this code does not track correspondence, as described in Section 5, nor handle the spherical case. The only routines not given here are either elementary numerical/geometric calculations, or operations that depend on the choice of mesh data structure. In particular:

- $\textsc{Solve}(A, x, b)$ — solves the linear system $Ax = b$ using a sparse linear solver that can handle a positive-semidefinite matrix A, such as sparse Cholesky or LDL factorization.
- $\textsc{Clausen}(x)$ — *Clausen's integral* $Cl_2(x)$, provided by standard numerical libraries [Galassi et al. 1994].
- $\textsc{Angle}(\ell_{ij}, \ell_{jk}, \ell_{ki})$ — given the edge lengths of a triangle ijk, computes angle θ_i^{jk} at corner i (*e.g.*, using the law of cosines).
- $\textsc{OppositeVertices}(ij, \mathsf{T})$ — given an edge ij contained in faces ijk, jil of a triangulation T, returns the vertices k and l.
- $\textsc{IsDelaunay}(\mathsf{T}, \ell, ij)$ — true if edge ij satisfies the local ideal Delaunay condition (Equation 10), false otherwise.
- $\textsc{IdealDelaunay}(\mathsf{T}, \ell)$ — same as $\textsc{IntrinsicDelaunay}$, except $\textsc{FlipEuclidean}$ is replaced with $\textsc{FlipPtolemy}$.
- $\textsc{Push}(Q, ij)$, $\textsc{Pop}(Q)$ — push/pop an edge to/form queue Q.
- $\textsc{InsertTriangles}(\mathsf{T}, i_1j_1k_1, i_2j_2k_2, \ldots)$, $\textsc{EraseTriangles}(\mathsf{T}, i_1j_1k_1, i_2j_2k_2, \ldots)$ — add/remove the given triangles to/from a triangulation T.

Note that we do not detail the routine $\textsc{Layout}$ which cuts the final mesh and lays it out in the plane—such algorithms are well-described in, *e.g.*, Springborn et al. [2008, Section 3.3] and Mullen et al. [2008], *resp.* Note also that there may be better numerical implementations of some methods (*e.g.*, for computing angles or their cotangents); see [Sharp et al. 2019b, Appendix A] for further discussion.

Algorithm 3 $\textsc{FlattenMesh}(\mathsf{T}^A, f, \Omega^*)$

Input: A triangle mesh $\mathsf{T}^A = (\mathsf{V}, \mathsf{E}^A, \mathsf{F}^A)$, vertex positions $f : \mathsf{V} \to \mathbb{R}^3$, and target cone angles $\Omega^* : \mathsf{V} \to \mathbb{R}$ that satisfy Equation 11.

Output: A triangle mesh $\mathsf{T}^C = (\mathsf{V}, \mathsf{E}^C, \mathsf{F}^C)$ with texture coordinates $z : \mathsf{V} \to \mathbb{R}^2$.

$\ell_{ij}^A \leftarrow |f_j - f_i|,\ \forall ij \in \mathsf{E}^A$ ▷ *measure edge lengths*
$(\mathsf{T}^B, \ell^B) \leftarrow \textsc{IntrinsicDelaunay}(\mathsf{T}^A, \ell^A)$ ▷ *Euclidean flips §3.4.1*
$u \leftarrow \textsc{MinimizeEnergy}(\mathsf{T}^B, \ell^B, \Omega^*)$
$(\mathsf{T}^C, \ell^C) \leftarrow \textsc{ScaleConformal}(u, \mathsf{T}^B, \ell^B)$
$z \leftarrow \textsc{Layout}(\mathsf{T}^C, \ell^C)$ ▷ *§4.5*
return (T^C, z)

Algorithm 4 $\textsc{MinimizeEnergy}(\mathsf{T}, \ell, \Omega^*)$

Input: A triangle mesh $\mathsf{T} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$, edge lengths $\ell : \mathsf{E} \to \mathbb{R}$, and target cone angles $\Omega^* : \mathsf{V} \to \mathbb{R}$ that satisfy Equation 11. A constant parameter $\varepsilon > 0$ determines the stopping tolerance, and parameters $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$ control line search (for details, see Boyd and Vandenberghe [2004] Algorithms 9.2 and 9.5).

Output: Scale factors $u : \mathsf{V} \to \mathbb{R}$ that realize the given angle defects.

$u \leftarrow 0 \in \mathbb{R}^{|\mathsf{V}|}$ ▷ *initial scale factors*
while true **do** ▷ *run Newton's method*
  $g \leftarrow \textsc{Gradient}(u, \mathsf{T}, \ell, \Omega^*)$ ▷ *§4.2.2*
  $L \leftarrow \textsc{Hessian}(u, \mathsf{T}, \ell)$ ▷ *§4.2.3*
  $\textsc{Solve}(L, v, -g)$ ▷ *compute descent direction v*
  if $v^\mathsf{T} g \leq 2\varepsilon$ **then** ▷ *check for convergence*
    break
  $\tau \leftarrow 1$
  $\mathcal{E}_0 \leftarrow \textsc{Energy}(u, \mathsf{T}, \ell)$ ▷ *§4.2*
  while $\textsc{Energy}(u + \tau v, \mathsf{T}, \ell) > \mathcal{E}_0 + \alpha\tau g^\mathsf{T} v$ **do** ▷ *line search*
    $\tau \leftarrow \beta\tau$
  $u \leftarrow u + \tau v$ ▷ *take Newton step*
return u

Algorithm 5 $\textsc{ScaleConformal}(u, \mathsf{T}, \ell)$

Input: A triangulation $\mathsf{T} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$, edge lengths $\ell : \mathsf{E} \to \mathbb{R}$, and conformal scale factors $u : \mathsf{V} \to \mathbb{R}$.

Output: A conformally equivalent Delaunay triangulation $(\widetilde{\mathsf{T}}, \tilde{\ell})$.

$\tilde{\ell}_{ij} \leftarrow e^{(u_i + u_j)/2},\ \forall ij \in \mathsf{E}$ ▷ *scale edge lengths §3.2.2*
$(\widetilde{\mathsf{T}}, \tilde{\ell}) \leftarrow \textsc{IdealDelaunay}(\mathsf{T}, \tilde{\ell})$ ▷ *Ptolemy flips §3.4.2*
return $(\widetilde{\mathsf{T}}, \tilde{\ell})$

Algorithm 6 $\textsc{Energy}(u, \mathsf{T}, \ell)$

Input: A triangulation $\mathsf{T} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$, edge lengths $\ell : \mathsf{E} \to \mathbb{R}$, conformal scale factors $u : \mathsf{V} \to \mathbb{R}$, and target angle defects $\Omega^* : \mathsf{V} \to \mathbb{R}$.

Output: The conformal energy at u (Section 4.2.1).

$(\widetilde{\mathsf{T}}, \tilde{\ell}) \leftarrow \textsc{ScaleConformal}(u, \mathsf{T}, \ell)$
$\mathcal{E} \leftarrow 0$ ▷ *will accumulate energy into $\mathcal{E}$*
for each $ij \in \widetilde{\mathsf{E}}$ **do** $\tilde{\lambda}_{ij} \leftarrow 2\log \tilde{\ell}_{ij}$ ▷ *Penner coordinates §3.3.3*
for each $_i^{jk} \in \widetilde{\mathsf{T}}$ **do** $\tilde{\theta}_i^{jk} \leftarrow \textsc{Angle}(\tilde{\ell}_{ij}, \tilde{\ell}_{jk}, \tilde{\ell}_{ki})$ ▷ *at all corners*
for each $i \in \mathsf{V}$ **do** $\mathcal{E} \leftarrow \mathcal{E} + (2\pi - \Omega_i^*)u_i$
for each $ij \in \widetilde{\mathsf{E}}$ **do** $\mathcal{E} \leftarrow \mathcal{E} - \pi\tilde{\lambda}_{ij}$
for each $ijk \in \widetilde{\mathsf{F}}$ **do**
  $\mathcal{E} \leftarrow \mathcal{E} + \tilde{\theta}_i^{jk}\tilde{\lambda}_{jk} + \tilde{\theta}_j^{ki}\tilde{\lambda}_{ki} + \tilde{\theta}_k^{ij}\tilde{\lambda}_{ij}$
  $\mathcal{E} \leftarrow \mathcal{E} + \textsc{Clausen}(2\tilde{\theta}_i^{jk} + \textsc{Clausen}(2\tilde{\theta}_j^{ki})) + \textsc{Clausen}(2\tilde{\theta}_k^{ij})$
return g

Algorithm 7 $\textsc{Gradient}(u, \mathsf{T}, \ell, \Omega^*)$

Input: A triangulation $\mathsf{T} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$, edge lengths $\ell : \mathsf{E} \to \mathbb{R}$, conformal scale factors $u : \mathsf{V} \to \mathbb{R}$, and target angle defects $\Omega^* : \mathsf{V} \to \mathbb{R}$.

Output: A vector $g \in \mathbb{R}^{|\mathsf{V}|}$ giving the gradient of the conformal energy $\mathcal{E}$ at u (Section 4.2.2).

$(\widetilde{\mathsf{T}}, \tilde{\ell}) \leftarrow$ ScaleConformal(u, T, ℓ)
for each $i \in \mathsf{V}$ **do**
  $\widetilde{\Omega}_i \leftarrow 2\pi$ ▷ *measure current angle defect*
  for each $ijk \in \widetilde{\mathsf{F}}$ containing i **do** $\widetilde{\Omega}_i \leftarrow \widetilde{\Omega}_i -$ Angle$(\tilde{\ell}_{ij}, \tilde{\ell}_{jk}, \tilde{\ell}_{ki})$
  $g_i \leftarrow \Omega_i^* - \widetilde{\Omega}_i$
return g

Algorithm 8 Hessian(u, T, ℓ)

Input: A triangulation $\mathsf{T} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$, edge lengths $\ell : \mathsf{E} \to \mathbb{R}$, and conformal scale factors $u : \mathsf{V} \to \mathbb{R}$.

Output: A matrix $L \in \mathbb{R}^{|\mathsf{V}| \times |\mathsf{V}|}$ giving the Hessian of the conformal energy $\mathcal{E}$ at u (Section 4.2.3).

$(\widetilde{\mathsf{T}}, \tilde{\ell}) \leftarrow$ ScaleConformal(u, T, ℓ)
$L \leftarrow 0 \in \mathbb{R}^{|V| \times |V|}$ ▷ *initialize empty sparse matrix*
for each $ij \in \mathsf{E}$ **do**
  $k, l \leftarrow$ OppositeVertices(ij, T)
  $\theta_k^{ij} \leftarrow$ Angle$(\tilde{\ell}_{ki}, \tilde{\ell}_{ij}, \tilde{\ell}_{jk})$
  $\theta_l^{ji} \leftarrow$ Angle$(\tilde{\ell}_{lj}, \tilde{\ell}_{ji}, \tilde{\ell}_{il})$
  $w_{ij} \leftarrow \frac{1}{2}(\cot \theta_k^{ij} + \cot \theta_l^{ji})$ ▷ *cotangent weight*
  $L_{ii}, L_{jj} += w_{ij}$
  $L_{ij}, L_{ji} -= w_{ij}$
return g

Algorithm 9 IntrinsicDelaunay(T, ℓ)

Input: A triangulation $\mathsf{T} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$ with edge lengths $\ell : \mathsf{E} \to \mathbb{R}$.

Output: An intrinsic Delaunay triangulation $\widetilde{\mathsf{T}}$ with edge lengths $\tilde{\ell} : \widetilde{\mathsf{E}} \to \mathbb{R}$ that encode the same Euclidean polyhedron.

$(\widetilde{\mathsf{T}}, \tilde{\ell}) \leftarrow (\mathsf{T}, \ell)$ ▷ *copy input*
for each $ij \in \widetilde{\mathsf{E}}$ **do** Push(Q, ij) ▷ *initialize queue Q*
while !Empty(Q) **do**
  $ij \leftarrow$ Pop(Q)
  if !IsDelaunay$(ij, \widetilde{\mathsf{T}}, \tilde{\ell})$ **then**
    $(\widetilde{\mathsf{T}}, \tilde{\ell}) \leftarrow$ FlipEuclidean$(ij, \widetilde{\mathsf{T}}, \tilde{\ell})$
    $k, l \leftarrow$ OppositeVertices$(ij, \widetilde{\mathsf{T}})$
    Push(Q, jk)
    Push(Q, ki)
    Push(Q, il)
    Push(Q, lj)
return $(\widetilde{\mathsf{T}}, \tilde{\ell})$

Algorithm 10 FlipEuclidean(T, ℓ, ij)

Input: A triangulation $\mathsf{T} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$, edge lengths $\ell : \mathsf{E} \to \mathbb{R}$, and an edge $ij \in \mathsf{E}$.

Output: An updated triangulation (T, ℓ) where ij has been flipped, and ℓ encodes the same Euclidean polyhedron.

$k, l \leftarrow$ OppositeVertices(ij, T)
$\theta_i^{lk} \leftarrow$ Angle$(\ell_{ij}, \ell_{jk}, \ell_{ki}) +$ Angle$(\ell_{il}, \ell_{lj}, \ell_{ji})$
$\ell_{kl} \leftarrow \sqrt{\ell_{ik}^2 + \ell_{il}^2 - 2\ell_{ik}\ell_{il}\cos\theta_i^{lk}}$ ▷ *law of cosines*

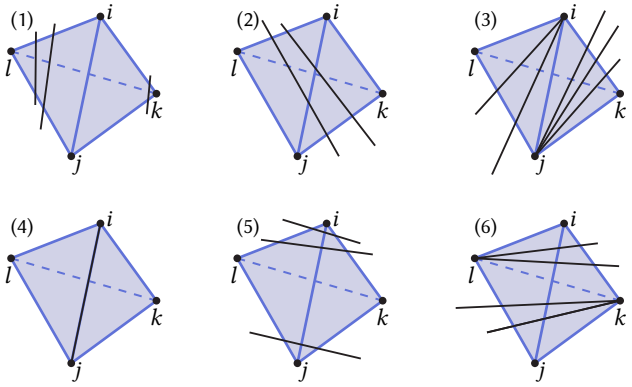


Fig. 35. Edges come in 6 types

EraseTriangles(ijk, jil)
InsertTriangles(ilk, jkl)
return (T, ℓ)

Algorithm 11 FlipPtolemy(T, ℓ, ij)

Input: A triangulation $\mathsf{T} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$, edge lengths $\ell : \mathsf{E} \to \mathbb{R}$, and an edge $ij \in \mathsf{E}$.

Output: An updated triangulation (T, ℓ) where ij has been flipped, and ℓ encodes the same discrete conformal structure.

$k, l \leftarrow$ OppositeVertices(ij, T)
$\ell_{kl} \leftarrow (\ell_{ki}\ell_{lj} + \ell_{jk}\ell_{li})/\ell_{ij}$
EraseTriangles(ijk, jil)
InsertTriangles(ilk, jkl)
return (T, ℓ)

D NORMAL COORDINATE UPDATE RULE

To derive Equation 15 of Section 5.1.1, consider first the case that lk is not an edge of T_1. Then the edges of T_1 intersect the interior of the quadrilateral $ikjl$ in segments of the following types (Figure 35):

(1) crossing corner l of ijl or crossing corner k of ijk
(2) crossing corners i of ijl and j of ijk, or crossing corners j of ijl and i of ijk
(3) emanating in corner i or j of ijl or ijk
(4) the edge ij
(5) crossing both corners i of ijk and ijl or both corners j of ijk and ijl
(6) emanating in corner l of ijl or emanating in corner k of ijk

Segments of types 1–4 are counted by

(1) $c_l^{ij} + c_k^{ij}$
(2) $\frac{1}{2}\left|c_j^{il} - c_j^{ki}\right| + \frac{1}{2}\left|c_i^{lj} - c_i^{jk}\right| - \frac{1}{2}e_l^{ji} - \frac{1}{2}e_k^{ij}$
(3) $e_i^{lj} + e_i^{jk} + e_j^{il} + e_j^{ki}$
(4) $\delta_{n_{ij}}$

and each contributes 1 to n_{jk}, while segments of types 5–6 contribute 0. To see the counting formulas for cases 2 and 4, note that $\frac{1}{2}\left|c_j^{il} - c_j^{ki}\right| + \frac{1}{2}\left|c_i^{lj} - c_i^{jk}\right|$ counts #{type 2} + $\frac{1}{2}$ #{type 6}, and that $n_{ij} = 0$ if and only if ij is also an edge of T_1.

Finally, consider the case that lk is an edge of T_1. Then term 2 above equals -1 and terms 1, 3, and 4 are zero. So Equation 15 is satisfied with both sides equal to zero.

E NUMERICALLY-ROBUST HYPERBOLIC GEODESICS

The hyperbolic layout procedure of Appendix A.3 can sometimes fail on meshes with near-degenerate triangles due to floating point issues. On many meshes, this can be ameliorated by tracing T^C over T^B instead of tracing T^B over T^C (Appendix E.1). When this is not sufficient, we fall back to an iterative straightening scheme (Appendix E.2).

E.1 Tracing Over the Intermediate Triangulation

In many examples, the final triangulation T^C has lower-quality triangles than the intermediate triangulation T^B. So although we need to trace T^B over T^C to interpolate texture coordinates, the algorithm performs better numerically if we trace T^C over T^B and then transpose to obtain the edges of T^B traced out over T^C. However, normal coordinates accumulated during the uniformization stage (Section 4) do not allow us to trace T^C over T^B directly. Our solution is to perform topological tracing (Section 6.1.1) of T^B over T^C, "transpose" these data to obtain topological edges of T^C traced over T^B, then straighten those curves to geodesics along T^B, and then transpose once more to obtain the edges of T^B traced out as geodesics along T^C.

E.2 Iterative Straightening

Consider the curve ab that we want to straighten to a hyperbolic geodesic passing through two adjacent ideal triangles ijk and kjl. Let x be the point where it intersects their common edge jk, and let y and z be the other edge intersection points. We will discuss the case where $y = ab \cap ij$ and $z = ab \cap lk$ shown in Figure 36. The other cases can be treated similarly. In each iteration of our straightening procedure, we update the barycentric coordinates of x such that the curve is geodesic within these two triangles.

Though we never actually compute homogeneous coordinates vectors q_i, q_j, q_k, q_l in $\mathbb{R}^3$, suppose for the moment that we write the homogeneous coordinates vectors of y and z as

$$\begin{aligned} q_y &= \sigma_y((1-t_{ij})q_i + t_{ij}q_j), \\ q_z &= \sigma_z((1-t_{lk})q_l + t_{lk}q_k). \end{aligned} \tag{20}$$

(The scale factors σ_y and σ_z are necessary because we do not assume the homogeneous coordinates to be normalized in any way.) Writing

$$q_x = (1-t_{jk})q_j + t_{jk}q_k$$

we determine t_{jk} using $\det(q_y, q_z, q_x) = 0$:

$$t_{jk} = \frac{\det(q_y, q_z, q_j)}{\det(q_y, q_z, q_j) - \det(q_y, q_z, q_k)}. \tag{21}$$

We substitute the expressions from Equation 20 into Equation 21 to obtain terms involving determinants of various combinations of q_i, q_j, q_k, q_l. Using the equation

$$|\det(q_i\ q_j\ q_k)| = 4\,\ell_{ij}\ell_{jk}\ell_{ik}. \tag{22}$$

(see Section F.1 for a derivation) and taking the orientations of the triangles into account, we can express these determinants purely in terms of the known edge lengths $\ell_{ij}, \ell_{jl}, \ell_{lk}, \ell_{ki}, \ell_{jk}$ and the length ℓ_{il} obtained via Ptolemy's formula (Equation 8). This enables us to calculate the barycentric coordinate t_{jk} of x directly from the known edge lengths, without ever actually laying out any triangles.

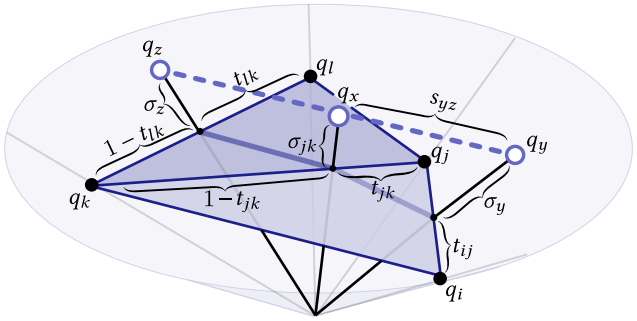


Fig. 36. In our iterative straightening procedure, we lay out two triangles at a time, and connect our segment's endpoints by a straight line.

Once this iterative straightening has converged, we compute some additional data. To get the barycentric coordinate s_{yz} of the intersection point along the segment from y to z, as well as the factor $\sigma_{jk} > 0$ relating the two intersection points, we solve the linear system

$$(1-s_{yz})\,q_y + s_{yz}q_z = \sigma_{jk}((1-t_{jk})\,q_j + t_{jk}q_k).$$

As before, since we don't know the coordinates q we take inner products with the homogeneous coordinate vectors q_j and q_k to get an equivalent linear system purely in terms of the edge lengths (using Equation 24 from the next section). In the event that this system is singular, we take inner products with q_y and q_z as well to get a full-rank system. This equation also provides us with the intersection's scale factor σ_{jk}. We obtain the log scale factor as $u_{jk} := \log(\sigma_{jk})$.

Finally, we still need the barycentric coordinate $s_{ab,x}$ for x relative to the whole geodesic ab—however, we do not yet know the exact barycentric coordinates $s_{ab,y}$ and $s_{ab,z}$ for y and z. To get an estimate, we therefore just take a weighted combination

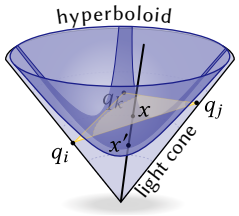
$$s_{ab,x} \leftarrow (1-s_{yz})\,s_{ab,y} + s_{yz}s_{ab,z}.$$

At the endpoints a and b of ab, we simply use barycentric coordinates 0 and 1, *resp.*.

F JUSTIFICATION OF HYPERBOLIC LAYOUT FORMULAS

F.1 Euclidean Triangles in the Hyperboloid Model

In the hyperboloid model of H^2, three points q_i, q_j, q_k on different rays in the positive light cone $\mathcal{L}^+$ determine an ideal hyperbolic triangle decorated with horocycles at the vertices (Section 3.3.1, Figure 13). If we connect q_i, q_j, q_k by straight lines in $\mathbb{R}^3$, we obtain a triangle in the affine plane spanned by the three points, whose sides are chords of the



light cone (see inset). The restriction of the Lorentz inner product $\langle \cdot, \cdot \rangle_{2,1}$ to this plane is positive definite, whenever the Euclidean slope is less than 45°. In this case, the affine plane is called *spacelike* because the Lorentz inner product provides a Euclidean metric, turning the chord triangle in Lorentz space into a genuine Euclidean triangle with side lengths

$$\ell_{ij} = \tfrac{1}{2}\sqrt{\langle q_i - q_j, q_i - q_j \rangle_{2,1}}. \tag{23}$$

(See the Remark below for an explanation of the factor $\frac{1}{2}$). Since q_i, q_j are light-like (*i.e.*, $\langle q, q \rangle_{2,1} = 0$), we have

$$\langle q_i - q_j, q_i - q_j \rangle_{2,1} = -2\langle q_i, q_j \rangle_{2,1},$$

and therefore

$$\langle q_i, q_j \rangle_{2,1} = -2\ell_{ij}^2. \tag{24}$$

The affine plane spanned by q_i, q_j, q_k is spacelike if and only if the chord lengths $\ell_{ij}, \ell_{jk}, \ell_{ki}$ obtained from Equation 23 satisfy the triangle inequalities. This gives us a direct mapping between any Euclidean triangle and its ideal hyperbolic counterpart: take the Euclidean triangle, and find points q_i on the light cone such that $\|q_i - q_j\|_{2,1} = 2\ell_{ij}$, yielding a copy of the Euclidean triangle sitting in $\mathbb{R}^{2,1}$. Each point x in the Euclidean triangle (except the vertices, which are light-like) can be normalized to obtain a point $x' = x/\sqrt{-\langle x, x \rangle_{2,1}}$ on the unit hyperboloid, which we identify with the hyperbolic plane (see the inset at the beginning of this section).

Equation 22 expresses the determinant of the three light-like vectors q_i, q_j, q_k in terms of the Euclidean edge lengths, at least up to the sign. To derive this equation, note that

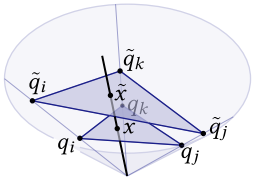
$$-2\begin{pmatrix} 0 & \ell_{ij}^2 & \ell_{ik}^2 \\ \ell_{ij}^2 & 0 & \ell_{jk}^2 \\ \ell_{ik}^2 & \ell_{jk} & 0 \end{pmatrix} = \begin{pmatrix} \langle q_i, q_i \rangle_{2,1} & \langle q_i, q_j \rangle_{2,1} & \langle q_i, q_k \rangle_{2,1} \\ \langle q_j, q_i \rangle_{2,1} & \langle q_j, q_j \rangle_{2,1} & \langle q_j, q_k \rangle_{2,1} \\ \langle q_k, q_i \rangle_{2,1} & \langle q_k, q_j \rangle_{2,1} & \langle q_k, q_k \rangle_{2,1} \end{pmatrix}$$
$$= (q_i \; q_j \; q_k)^T \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} (q_i \; q_j \; q_k)$$

and take determinants.

Remark. In Equation 23 measuring length in Lorentz space, we insert the global scale factor $\frac{1}{2}$ to be consistent with Equation 6 describing the relation between truncated hyperbolic lengths λ and Euclidean lengths ℓ. This relation was originally derived by Bobenko et al. [2015] (and used in this form by Springborn et al. [2008]) via a different construction involving ideal tetrahedra in hyperbolic 3-space. Both constructions provide the same correspondence between Euclidean and decorated ideal triangles, up to scale. The natural scale for Euclidean lengths in the construction of Bobenko et al. [2015] happens to differ from the natural scale in the light cone by a factor of 2.

F.2 Vertex Scaling and Projective Interpolation

Now consider the points $\tilde{q}_i = e^{u_i} q_i$, $\tilde{q}_j = e^{u_j} q_j$, $\tilde{q}_k = e^{u_k} q_k$ on the same rays in the light cone, describing the same ideal triangle but decorated with different horocycles. By Equation 24, their chordal distances $\tilde{\ell}$ are related to the chordal distances ℓ by Equation 3.



Moreover, if the scaled lengths $\tilde{\ell}$ satisfy the triangle inequalities, then the second triangle is also Euclidean and the circumcircle preserving projective map between them [Springborn et al. 2008, Section 3.4] is just central projection mapping a point x to the point $\tilde{x}$ in the same ray from the origin (see inset).

More explicitly, suppose we have a linear function on triangle $\widetilde{ijk}$ defined by values $\tilde{f}_i, \tilde{f}_j, \tilde{f}_k$ at the vertices. We want to pull this function back to the lower triangle ijk by defining $f(x) = \tilde{f}(\tilde{x})$. Suppose

$$x = \alpha_i q_i + \alpha_j q_j + \alpha_k q_k.$$

Since $\tilde{q}_i = e^{u_i} q_i$, we can also write

$$x = \alpha_i e^{-u_i} \tilde{q}_i + \alpha_j e^{-u_j} \tilde{q}_j + \alpha_k e^{-u_k} \tilde{q}_k.$$

To scale x to lie in the triangle spanned by $\tilde{q}_i, \tilde{q}_j, \tilde{q}_k$, we just have to normalize its coefficients to sum to 1:

$$\tilde{x} = \frac{\alpha_i e^{-u_i} \tilde{q}_i + \alpha_j e^{-u_j} \tilde{q}_j + \alpha_k e^{-u_k} \tilde{q}_k}{\alpha_i e^{-u_i} + \alpha_j e^{-u_j} + \alpha_k e^{-u_k}}.$$

Finally, we can evaluate our function f. Since $\tilde{f}$ is linear, we find that

$$f(x) = \tilde{f}(\tilde{x}) = \frac{\alpha_i e^{-u_i} \tilde{f}_i + \alpha_j e^{-u_j} \tilde{f}_j + \alpha_k e^{-u_k} \tilde{f}_k}{\alpha_i e^{-u_i} + \alpha_j e^{-u_j} + \alpha_k e^{-u_k}}.$$

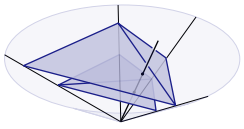
This is precisely the circumcircle-preserving projective map between our two triangles. We can write it more compactly by introducing *homogeneous coordinates*.

$$h(x) = \alpha_i e^{-u_i}(\tilde{f}_i, 1) + \alpha_j e^{-u_j}(\tilde{f}_j, 1) + \alpha_k e^{-u_k}(\tilde{f}_k, 1), \tag{25}$$

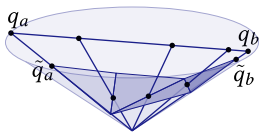
and we obtain $f(x)$ by dividing the first component of $h(x)$ by the second component. In the end, our interpolation amounts to linearly interpolating the values $e^{-u_i}(\tilde{f}_i, 1)$ and then performing this homogeneous divide.

F.3 Edge Flips

The real power in the hyperboloid is that it allows us to interpolate between different triangulations of the same vertex set using exactly the same procedure. Consider for example a pair of triangles which have been flipped (by a Ptolemy flip) and rescaled. In the hyperboloid model, the Ptolemy flip really does correspond to an extrinsic flip, since the extrinsic Lorentz distance corresponds to the hyperbolic distance between horocycles. So if we take two triangles, perform a Ptolemy flip, and then rescale the edge lengths, we end up with two pairs of triangles with one hanging above the other. We can map between the two triangle pairs by rescaling, exactly as in the 1-triangle case. The rescaling map is a piecewise-projective map on the common refinement of the two meshes. Since the map is piecewise-projective, we can specify the whole map by computing how much it scales by at each vertex, and at each projective intersection of edges. In this case, we can find the intersection point and the map's scale factor at the intersection by applying Equation 19.



F.4 Piecewise-Projective Interpolation



Now, suppose the triangulations differ by more than just an edge flip. As we observed above, the piecewise projective map depends only on scale factors at vertices, and at the intersections between edges of the two triangulations. We know the scale factors at vertices, so in order to compute the piecewise-projective map, we need to determine where the edges of the two triangulations intersect, and what the appropriate scale factor is at each intersection. In our algorithm, we need to trace edges of T^B over T^C. For each edge $ab \in \mathsf{E}^B$, this amounts to laying out the strip of triangles from T^C which ab crosses in the light cone, drawing the edge from q_a to q_b above it, and computing barycentric coordinates and scale factors for each intersection. Since $\tilde{q}_a = e^{u_a} q_a$, we simply place q_a and q_b by rescaling the triangle strip's endpoints by e^{-u_a} and e^{-u_b} respectively in Appendix A.3. Once we have computed these scale factors, we do projective interpolation using Equation 25 on each triangle of the common refinement. The final expression appears as Equation 16.

F.5 Discrete Uniformization: Hyperboloid Model POV

To understand mapping between the intrinsic Delaunay triangulation (T^B, ℓ) and the discretely conformally equivalent intrinsic Delauny triangulation $(\mathsf{T}^C, \tilde{\ell})$ that is obtained by vertex scaling with logarithmic factors u and Ptolemy flips, it is useful to picture this process in the hyperboloid model as follows.

First, imagine laying out the triangulation T^B in the light cone. We will provide more detail in the following two sections, but the idea is straightforward: Place the vertices q_i, q_j, q_k of a first triangle ijk on arbitrary rays in $\mathcal{L}^+$ so that the chordal distances are ℓ_{ij}, ℓ_{j_k}, ℓ_{ki}. Then for a neighboring triangle, say jil, the position $q_l \in \mathcal{L}^+$ of the third vertex is determined by the side lengths ℓ_{il}, ℓ_{jl}. Note that as you layout the triangles around one vertex, this will never close up. Instead, if you keep laying out, each triangle of T^B will correspond to infinitely many chordal triangles. The result is a polyhedral surface P_1 with vertices in the light cone, all of which have infinite degree. Yet, every ray from the origin contained in the light cone will intersect this polyhedral surface exactly once. Moreover, the ideal Delaunay condition on (T^B, ℓ^B) is precisely the condition that this polyhedral surface is convex [Penner 2012, Lemma 1.7, p. 128].

The next step, corresponding to the vertex scaling (Equation 3), is to slide all the laid out vertices along their rays in the light cone by applying the scale factors e^u as in Appendix F.2. The resulting polyhedral surface P_2 will in general not be convex, nor will all its triangles span spacelike planes.

The process of applying Ptolemy flips to obtain the ideal Delaunay triangulation $(\mathsf{T}^C, \tilde{\ell})$ corresponds, in the hyperboloid model, to applying extrinsic edge flips to the polyhedral surface P_2 to obtain a convex surface P_3. All of its triangles will then automatically be in spacelike planes. Finally, the map from P_1 to P_3 is just central projection from the origin.

F.6 Layout in the Light Cone I: Placing the First Triangle

We will now derive some practical equations for laying out a Euclidean triangulation in the light cone. Note that in practice we only ever lay out triangle strips of one triangulation that are crossed by an edge of another triangulation.

To lay out the first triangle ijk we place the vertices at the points

$$\begin{aligned} q_i &= w_i\,(1, 0, 1), \\ q_j &= w_j\,(\cos(2\pi/3), \sin(2\pi/3), 1), \\ q_k &= w_k\,(\cos(4\pi/3), \sin(4\pi/3), 1), \end{aligned}$$

in $\mathcal{L}^+$, where the positive scalar factors w_i, w_i, w_i are determined by the edge lengths via Equation 24:

$$\begin{aligned} \ell_{ij}^2 &= -\tfrac{1}{2}\langle q_i, q_j\rangle_{2,1} = \tfrac{3}{4} w_i w_j \\ \ell_{jk}^2 &= -\tfrac{1}{2}\langle q_j, q_k\rangle_{2,1} = \tfrac{3}{4} w_j w_k \\ \ell_{ki}^2 &= -\tfrac{1}{2}\langle q_k, q_i\rangle_{2,1} = \tfrac{3}{4} w_k w_i \end{aligned}$$

The solution of this system of equations is

$$w_i = \frac{2\,\ell_{ij}\ell_{ki}}{\sqrt{3}\,\ell_{jk}}, \qquad w_j = \frac{2\,\ell_{jk}\ell_{ij}}{\sqrt{3}\,\ell_{ki}}, \qquad w_k = \frac{2\,\ell_{ki}\ell_{jk}}{\sqrt{3}\,\ell_{ij}}\,.$$

F.7 Layout in the Light Cone II: Placing the Next Triangle

Suppose we have already determined the vertex positions $q_i, q_j, q_k \in \mathcal{L}^+$ of the triangle ijk, and we want to determine the position q_l of third vertex in the adjacent triangle jil. Note that q_i, q_j, q_k form a basis of $\mathbb{R}^3$ so and we can write the unknown vertex position q_l as a linear combination. To obtain more symmetric expressions, we will determine coefficients c_i, c_j, c_k, c_l for which

$$c_i q_i + c_j q_j + c_k q_k + c_l q_l = 0. \tag{26}$$

By taking the inner product of Equation 26 with each of the four vertex positions q and using Equation 24, we get a system of linear equations

$$\begin{bmatrix} 0 & \ell_{ij}^2 & \ell_{ik}^2 & \ell_{il}^2 \\ \ell_{ij}^2 & 0 & \ell_{jk}^2 & \ell_{jl}^2 \\ \ell_{ik}^2 & \ell_{jk}^2 & 0 & \ell_{kl}^2 \\ \ell_{il}^2 & \ell_{jl}^2 & \ell_{kl}^2 & 0 \end{bmatrix} \begin{bmatrix} c_i \\ c_j \\ c_k \\ c_l \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

where $\ell_{ij}, \ell_{jk}, \ell_{ki}, \ell_{il}, \ell_{lj}$ are the sides lengths of the triangles ijk and jil, and ℓ_{kl} is determined by Ptolemy's formula (Equation 8). A solution of this system is given by

$$c_i = \frac{\ell_{jl}\ell_{kl}}{\ell_{il}}, \quad c_j = -\frac{\ell_{ik}\ell_{kl}}{\ell_{jk}}, \quad c_k = -\frac{\ell_{jl}\ell_{ij}}{\ell_{jk}}, \quad c_l = \frac{\ell_{ik}\ell_{ij}}{\ell_{il}}.$$

We can use these coefficients in Equation 26 to get the next vertex position q_l.