Boundary First Flattening

ROHAN SAWHNEY and KEENAN CRANE, Carnegie Mellon University

A *conformal flattening* maps a curved surface to the plane without distorting angles—such maps have become a fundamental building block for problems in geometry processing, numerical simulation, and computational design. Yet existing methods provide little direct control over the shape of the flattened domain, or else demand expensive nonlinear optimization. *Boundary first flattening (BFF)* is a linear method for conformal parameterization which is faster than traditional linear methods, yet provides control and quality comparable to sophisticated nonlinear schemes. The key insight is that the boundary data for many conformal mapping problems can be efficiently constructed via the *Cherrier formula* together with a pair of *Poincaré-Steklov operators*; once the boundary is known, the map can be easily extended over the rest of the domain. Since computation demands only a single factorization of the real Laplace matrix, the amortized cost is about 50x less than any previously published technique for boundary-controlled conformal flattening. As a result, BFF opens the door to real-time editing or fast optimization of high-resolution maps, with direct control over boundary length or angle. We show how this method can be used to construct maps with sharp corners, cone singularities, minimal area distortion, and uniformization over the unit disk; we also demonstrate for the first time how a surface can be conformally flattened directly onto any given target shape.

CCS Concepts: • **Computing methodologies → Mesh models**; • **Mathematics of computing → Partial differential equations**;

Additional Key Words and Phrases: discrete differential geometry, digital geometry processing, surface parameterization, conformal geometry

1 INTRODUCTION

In recent years conformal flattening has evolved far beyond its humble origins as a tool for texture mapping [Desbrun et al. 2002; Lévy et al. 2002], providing new perspectives on a broad range of applications including surface remeshing [Alliez et al. 2003; Zhong et al. 2014], comparative data analysis [Hurdal and Stephenson 2009; Lipman and Daubechies 2011], computational biology [Koehl and Hass 2015], physical simulation [Segall et al. 2016], sensor networks [Sarkar et al. 2009], and computational design [Konakovic et al. 2016]. Why such great interest in maps that preserve *angles*? One answer is computational: conformal mapping typically amounts to solving easy linear equations, providing fast, scalable algorithms—or cheap initialization for more difficult nonlinear tasks [Chao et al. 2010; Liu et al. 2008]. Another is that angle preservation is directly linked to real mechanical or constitutive properties of physical systems [Kim et al. 2012]. Continued advancement of basic tools for conformal mapping therefore has high utility across a variety of domains. At present, however, linear algorithms for conformal flattening do not exploit the full space of possibilities.
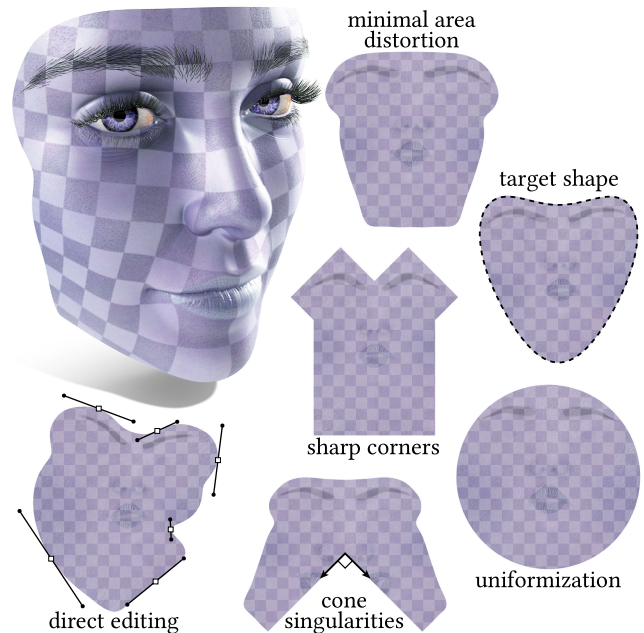
Fig. 1. *Boundary first flattening (BFF)* provides sophisticated control over the shape of an angle-preserving or *conformal* map to the plane. Even on very large meshes, this shape can be edited in real time.

In the smooth setting, conformal maps are quite flexible: a map from a disk-like surface to the flat plane can achieve any target shape as long as it is free to "slide" along the boundary. In stark contrast, existing linear conformal flattening algorithms provide *no* explicit control over the target shape: the user obtains a single, automatic solution and must "take it or leave it." Nonlinear methods provide additional control over shape, but at significantly greater cost. *Boundary first flattening (BFF)* is the first conformal flattening method to provide full control over the target shape via a single sparse matrix factorization, including:

- automatic flattening with optimal area distortion,
- direct manipulation of boundary length or angle,
- exact preservation of sharp corners,
- seamless cone parameterization,
- uniformization over the unit disk, and
- mapping to a given target shape.

(See examples in Figure 1.) The target shape can be efficiently updated via backsubstitution, providing a new paradigm for conformal parameterization: rather than waiting for a single, predetermined flattening, one can interactively edit or optimize the map—even on meshes with hundreds of thousands of elements. More broadly, BFF is a drop-in replacement for widely-used schemes like least-squares conformal maps (LSCM) while providing sophisticated control over features like shape and area distortion.

1.1 Algorithm Outline

Given a target length (or curvature) function along the boundary, the BFF algorithm involves three basic steps:

    I. Solve for a compatible curvature (or length) function.
    II. Integrate this data to get a boundary curve.
    III. Extend this curve over the interior of the domain.

Computation amounts to solving three sparse linear problems (in steps I and III) connected by a simple nonlinear change of variables (step II); all linear problems use the same fixed Laplace matrix. From here, the essential question is how to devise boundary data (curvatures or lengths) suitable for a variety of mapping problems, as explored in Sec. 6. An important limitation of BFF is that it applies only to domains with disk topology, though of course a surface of any topology can be cut into one or more disks; see Sec. 7.5 for further discussion. Detailed pseudocode is provided in App. B.

2 RELATED WORK

The literature on surface flattening and computational conformal geometry is vast [Gu and Yau 2008; Sheffer et al. 2006]—here we focus on conformal flattening methods that provide control over boundary shape, or that can be modified to provide such control. Early methods for conformal flattening compute a piecewise linear least-squares solution of the *Cauchy-Riemann equations* (**LSCM** [Lévy et al. 2002]), or equivalently, minimize the difference between *Dirichlet energy* and the flattened area (**DCP** [Desbrun et al. 2002]); quality was later improved via a sparse eigenvalue problem (**SCP** [Mullen et al. 2008]). A different line of methods directly optimizes angles in the flattened mesh to approximate angles in the input [Sheffer and de Sturler 2001; Sheffer et al. 2005], a process that can be linearized without compromising quality (**LinABF** [Zayer et al. 2007]).

Free Boundary Conditions. The methods mentioned so far forgo the question of boundary control, instead opting for so-called "free" boundary conditions where a discrete energy is minimized without explicit constraints on boundary degrees of freedom. Although this approach provides a unique solution in the discrete setting, it has no meaningful interpretation in the smooth setting: in the absence of explicit boundary conditions there is an enormous space of perfect conformal flattenings, obtained by flattening and then applying an in-plane conformal map (see inset). The only possibility is that *the unique solution chosen by free boundary methods **must** depend on discretization*, meaning that results will change depending on the particular choice of mesh or numerical treatment (see Figure 2, and Mullen et al. [2008, Figure 1]). This phenomenon further motivates the need for more careful treatment of boundary conditions.

Prescribed Boundary Length/Angle. Later methods explicitly incorporate boundary constraints. For instance, the method of *circle patterns* (**CP** [Kharevych et al. 2006]) provides control over the direction of the boundary (but not its length); methods based on *discrete Ricci flow* [Jin et al. 2008; Luo 2004] can provide control over both length and direction (**CETM** [Springborn et al. 2008]). All of these
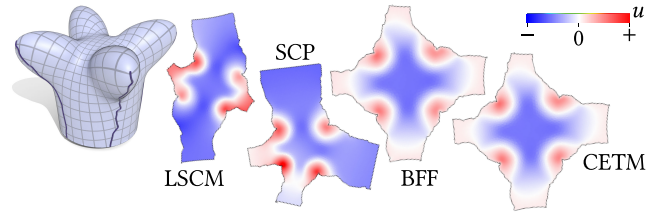

Fig. 2. A surface is cut and conformally flattened (u is the scale factor). In the absence of proper boundary conditions, traditional methods (LSCM and SCP) produce different solutions since the underlying smooth energy does not have a unique minimizer. By adding explicit boundary conditions our method (BFF) provides a canonical map with minimal area distortion, nearly identical to more expensive nonlinear methods (CETM).

methods demand nonlinear optimization involving, *e.g.*, repeated matrix factorization for each Newton iteration. In Sec. 7.1 we explore how linearization of both angle-based (LinABF) and length-based strategies (**CPMS** [Ben-Chen et al. 2008]) can be adapted to provide complete boundary control; even then, these methods remain at least 30x slower than BFF and can exhibit significant artifacts. A very different approach is to discretize a *time-independent Dirac equation* that governs conformal surface deformations in 3D [Crane 2013, Section 6.1]; this approach provides control over boundary direction (but not length) and must solve an eigenvalue problem that cannot be prefactored for varying boundary data.

Uniformization. A natural idea for achieving a given target shape is to compose maps to a canonical domain like the circular disk (Sec. 6.5). In the discrete setting, however, piecewise linear conformal maps do not compose; more importantly, methods for computing such maps [Bobenko et al. 2010; Zeng et al. 2008] are already more expensive than just directly editing the boundary via BFF.

2D Shape Editing. Methods for planar shape deformation are generally not suitable for conformal flattening since they depend on boundary element methods or closed-form expressions that are available only on planar domains [Chen et al. 2013; Lipman et al. 2012; Weber and Gotsman 2010]. One idea is to apply 2D deformation to an initial flattening, but the resulting nonlinear map may be incompatible with the standard geometry processing pipeline, especially when one requires precise control over individual lengths and angles (Sec. 6). Moreover, a composition of methods offers no clear advantage in terms of speed or simplicity over the unified scheme we propose here.

Outside of strictly conformal methods a variety of algorithms provide boundary control, albeit with very different performance characteristics—see for example Weber & Zorin [2014] and references therein. At a high level, all previous boundary-controlled *conformal* methods (namely, CP and CETM plus our modifications of CPMS and LinABF) indirectly encode a flattening via metric data (lengths or angles). In order for this data to describe a valid planar triangulation it must satisfy nonlinear integrability conditions over the entire domain. In contrast, BFF requires only that data describes a closed boundary *curve*—a condition that is far easier to satisfy, once the associated integrability conditions are clearly understood.
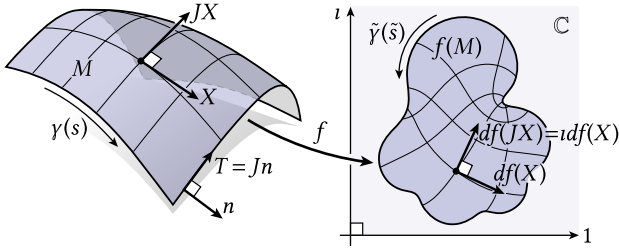
Fig. 3. Basic quantities used in our algorithm, which computes a conformal map f from a surface M to the plane $\mathbb{C}$.

3 BACKGROUND

We first provide some key definitions from (discrete) differential geometry; a more pedagogical introduction can be found in Crane *et al.* [2013] (*esp.* Chapter 7). Throughout the document, clicking on most symbols will provide a hyperlink back to their definition.

3.1 Notation

Single brackets denote the norm $|\cdot|$ and real inner product $\langle\cdot,\cdot\rangle$ of finite-dimensional vectors. For any complex number $z \in \mathbb{C}$, $\mathrm{Re}(z)$ and $\mathrm{Im}(z)$ denote the real and imaginary parts, and the imaginary unit ι denotes a quarter-turn in the counter-clockwise direction (hence $\iota^2 = -1$). The *argument* $\arg(z)$ of a complex number is the smallest (in magnitude) signed angle from the real axis to z; the angle from z_1 to z_2 can therefore be expressed as $\arg(z_1^{-1} z_2)$. Italic glyphs $(A, b, \ldots)$ typically indicate a continuous quantity whereas sans-serif characters (A, b) denote discrete quantities and/or matrices.

3.2 Smooth Setting

Our main object of study is a map $f : M \to \mathbb{C}$ from a disk-like surface M (with Riemannian metric) to the flat complex plane $\mathbb{C}$ (Figure 3). At each point the map J rotates any tangent vector X by a quarter-turn in the counter-clockwise direction so that (like the imaginary unit) $J^2 X = -X$. The boundary ∂M is a single closed loop parameterized by a curve γ with arc-length parameter s. We likewise use $\tilde{\gamma}(\tilde{s})$ to parameterize the image of the boundary $f(\partial M)$; it is the shape of this curve that we seek to control. By convention, if T is the unit tangent in the direction $\frac{d}{ds}\gamma$, then $n := -JT$ is the outward unit normal. We use K to denote the Gaussian curvature of M, and $\kappa, \tilde{\kappa}$ for the (geodesic) curvatures of γ and $\tilde{\gamma}$, respectively.

3.2.1 Conformal Maps. Intuitively, a map $f : M \to \mathbb{C}$ is conformal if at each point it preserves the angle between any two vectors, permitting only a uniform change in length. More precisely, let df denote the *differential* of f, which determines how a given tangent vector X on the surface gets mapped to a tangent vector $df(X)$ in the complex plane (in coordinates, df is represented by the familiar *Jacobian matrix*). A map f is *holomorphic* if

$$df(JX) = \iota df(X) \qquad (1)$$

for all tangent vectors X, *i.e.*, if a quarter-turn J on the surface yields the same result as a quarter-turn ι in the plane; this relationship is known as the *Cauchy-Riemann equation*. If in addition df is nondegenerate (*i.e.*, it maps nonzero vectors to nonzero vectors) then f is

conformal. The *conformal (scale) factor* $e^u := |df(X)|/|X|$ quantifies the change in length at each point (which is independent of direction X); the function $u : M \to \mathbb{R}$ is called the *log conformal factor*.

Conformal maps can also be expressed as pairs of *conjugate harmonic functions*. A real function $a : M \to \mathbb{R}$ is *harmonic* if it solves the *Laplace equation* $\Delta a = 0$ (Sec. 4.1), where Δ is the *Laplace-Beltrami operator* (or just *Laplacian*) associated with the domain M. Suppose we express a holomorphic map as $f = a + b\iota$ for a pair of coordinate functions $a, b : M \to \mathbb{R}$. Then (by Cauchy-Riemann)

$$J\nabla a = \nabla b, \qquad (2)$$

i.e., the gradients ∇ of the two coordinates are orthogonal and have equal magnitude. Since a quarter-rotation of a gradient field is divergence-free, we have

$$\Delta a \;=\; \nabla \cdot \nabla a \;=\; -\nabla \cdot (J\nabla b) \;=\; 0,$$

and similarly, $\Delta b = 0$. In other words, the two real components of a holomorphic function are both *harmonic*—we say that a and b form a *conjugate harmonic* pair.

To what degree can one control the target shape of a conformal flattening? Equation 2 suggests there must be some limitation, since two arbitrary functions on the boundary may not extend to a conjugate harmonic pair on the interior. And yet the *Riemann mapping theorem* states that any disk-like region $U \subset \mathbb{C}$ can be conformally mapped to the unit circular disk (Sec. 6.5); hence any shape can be obtained via composition with an (inverse) Riemann map. These two facts do not contradict each-other: although one can surjectively map onto any given region ($f(M) = U$), one cannot arbitrarily "pin" each boundary point to a specific location ($f|_{\partial M} = \tilde{\gamma}$) and hope to remain conformal. We therefore express our basic algorithm in terms of geometric quantities that can always be realized: one can specify *either* length *or* curvature density along the boundary, *but not both simultaneously* (see Figure 4, Sec. 4.2 and [Springborn et al. 2008, Section 4]). This data can then be further manipulated to provide a variety of intuitive control schemes (Sec. 6).
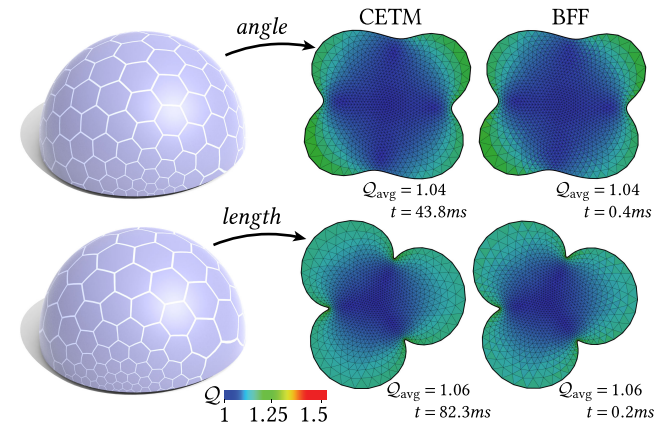

Fig. 4. A conformal flattening is uniquely determined (up to global similarity) by either length *(bottom)* or curvature *(top)* data along the boundary. Here we prescribe simple oscillatory functions, achieving results virtually indistinguishable from a reference solution (CETM) despite the fact that BFF is over 100x faster. (Hexagonal pattern emphasizes preservation of angles.)

3.2.2 Curvature Density.

In developing boundary conditions it helps to distinguish between the usual pointwise curvatures κ, K, and the corresponding curvature *densities*, because of the way these quantities transform under a conformal map. Loosely speaking, a *density* assigns a positive volume to each little piece of a curve or surface, *e.g.*, the length density ds on a curve or the area density dA on a surface. Under a conformal map, new and old densities scale according to the conformal factor: $d\tilde{s} = e^u ds$ and $d\tilde{A} = e^{2u} dA$. A *curvature density* multiples a density by a curvature function—in particular, κds is the (geodesic) curvature density of a curve, and KdA is the Gauss curvature density of a surface. The change of curvature density under a conformal map is studied in Sec. 4.2.

3.3 Discrete Setting
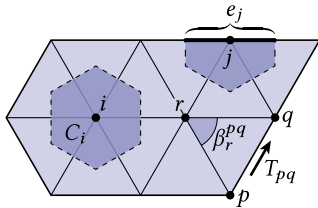
We discretize our surface M as a manifold triangle mesh $\mathsf{M} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$ with disk topology, using $\mathsf{B} \subseteq \mathsf{V}$ to denote the set of vertices on the boundary $\partial \mathsf{M}$, and $\mathsf{I} := \mathsf{V}\backslash\mathsf{B}$ for interior vertices. Tuples of vertex indices are used to specify simplices, *e.g.*, $ij \in \mathsf{E}$ is an edge from vertex i to vertex j. An expression of the form $a_i = \sum_{ij\in\mathsf{E}} b_{ij}$ means that a quantity b is summed over all edges containing vertex i to get the value of a at i. Likewise, $a_i = \sum_{ijk\in\mathsf{F}} b_{ijk}$ denotes a sum over faces containing vertex i. A quantity at corner i of triangle ijk is denoted by a subscript i and superscript jk; for instance, we use $\beta_i^{jk} \in \mathbb{R}$ to denote the interior angle at the corner of a triangle. Throughout we use C_i to denote the dual cell associated with vertex $i \in \mathsf{V}$, and e_j to denote the barycentric dual edge associated with a boundary vertex $j \in \mathsf{B}$; T_{pq} denotes the unit tangent along boundary edge pq (see inset). We use ℓ_{ij} to denote the length of edge ij in the input mesh, and $\ell_j := \frac{1}{2}(\ell_{ij} + \ell_{jk})$ to denote the length of the dual boundary edge e_j, where i, j, and k are consecutive vertices along the boundary.

3.3.1 Discrete Curvature.

Gaussian curvature. For each interior vertex i of a triangulation the angle defect $\Omega_i := 2\pi - \sum_{ijk\in\mathsf{F}} \beta_i^{jk}$ quantifies the "flatness" of a vertex as a deviation from

the planar angle sum of 2π. This quantity encodes not the *pointwise* Gaussian curvature, but rather the curvature integrated over a small region around the vertex: $\Omega_i = \int_{C_i} K\, dA$ (since for a discrete surface K is a distribution concentrated at vertices). We therefore refer to Ω as a *discrete curvature density*.
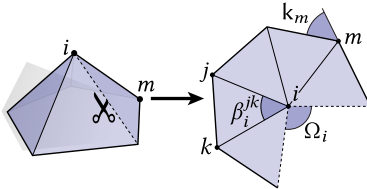
Geodesic curvature. At boundary vertices, the Gaussian curvature (density) Ω_i is zero, since a small neighborhood around any boundary vertex $i \in \mathsf{B}$ can be flattened into the plane without stretching. Here we consider a different quantity $\mathsf{k}_i := \pi - \sum_{ijk\in\mathsf{F}} \beta_i^{jk}$, which encodes the integral of the *geodesic* curvature κ (again a distribution at vertices). In the plane, k_i are also the *exterior angles*, *i.e.*, the change in tangent direction from one edge to the next.

The discrete setting helps illustrate the challenge of prescribing pointwise curvature rather than curvature density. Suppose we want to achieve a boundary curvature κ. If we provide only exterior angles k, then there are in general many

polygons that agree with this data (see inset). To uniquely prescribe a target shape, we must be able to control the change in angle *per unit length*. Likewise, to prescribe Gaussian curvature, we must be able to control the angle defect *per unit area*. Algorithmically this goal is difficult to achieve since the final lengths and areas are are determined by a scale factor u that is not known *a priori*—any algorithm that prescribes angle defects (such as CPMS or CETM) actually controls the *(discrete) curvature density* rather than the curvature itself. One approach to this problem is described in Sec. 6.6.
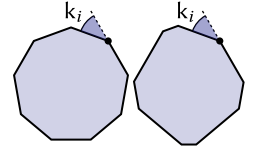
4 TOOLBOX

We first describe a collection of basic building blocks, which are assembled into the final algorithm in Sec. 5.

4.1 Poisson Problems

The development of BFF hinges on careful treatment of boundary conditions for the *Dirichlet-Poisson problem*

$$\Delta a = \phi \quad \text{on} \quad M; \qquad a = g \quad \text{on} \quad \partial M \qquad (3)$$

and the *Neumann-Poisson problem*

$$\Delta a = \phi \quad \text{on} \quad M; \qquad \frac{\partial a}{\partial n} = h \quad \text{on} \quad \partial M, \qquad (4)$$

where a and ϕ are real-valued functions on M, and $g, h : \partial M \to \mathbb{R}$ determine the values or normal derivatives along the boundary, *resp.* The solution to Eqn. 4 is unique only up to a constant, which in BFF just determines the global scale and translation. On a triangle mesh, integrating $\Delta a = \phi$ over dual cells yields a matrix equation

$$\mathsf{A}a = \mathsf{P}\phi,$$

where $\mathsf{A} \in \mathbb{R}^{|\mathsf{V}|\times|\mathsf{V}|}$ is the so-called *cotan-Laplace matrix* since its nonzero entries can be expressed as

$$\mathsf{A}_{ij} = -\tfrac{1}{2}(\cot \beta_p^{ij} + \cot \beta_q^{ij})$$

for each edge $ij \in \mathsf{E}$ with opposite vertices p, q, and $\mathsf{A}_{ii} = -\sum_{ij\in\mathsf{E}} \mathsf{A}_{ij}$ for each vertex $i \in \mathsf{V}$ (see Crane et al. [2013, Section 6.3] for a derivation); omitting the unknown cotans at boundary edges corresponds to zero-Neumann boundary conditions (MacNeal [1949, Eqn. 3.3]). The matrix P is the *mass matrix*, but is not needed for BFF since all values appearing on the right-hand side will be integrated quantities (*e.g.*, the discrete Gaussian curvature Ω).

If we partition into interior vertices I and boundary vertices B, a Neumann-Poisson problem can be expressed in block form as

$$\begin{bmatrix} \mathsf{A}_{\mathsf{II}} & \mathsf{A}_{\mathsf{IB}} \\ \mathsf{A}_{\mathsf{IB}}^\mathsf{T} & \mathsf{A}_{\mathsf{BB}} \end{bmatrix} \begin{bmatrix} a_I \\ a_B \end{bmatrix} = \begin{bmatrix} \phi_{\mathsf{I}} \\ \phi_B - \mathsf{h} \end{bmatrix}, \qquad (5)$$

where $\mathsf{h} \in \mathbb{R}^{|\mathsf{B}|}$ is the discrete Neumann boundary data, corresponding to the *integral* of $\partial a/\partial n$ over each dual boundary edge e_i. A Dirichlet problem with boundary values $a_B = \mathsf{g} \in \mathbb{R}^{|\mathsf{B}|}$ is then obtained by solving the first row of Eqn. 5 for the interior values a_I:

$$\mathsf{A}_{\mathsf{II}}a_I = \phi_I - \mathsf{A}_{\mathsf{IB}}\mathsf{g}. \qquad (6)$$

Efficient Solution. Solving a sequence of Dirichlet- and Neumann-Poisson problems requires only a single sparse Cholesky factorization $A = LL^T$ of the matrix A from the Neumann problem. To see why, consider the blockwise expansion

$$\begin{bmatrix} A_{II} & A_{IB} \\ A_{IB}^T & A_{BB} \end{bmatrix} = \begin{bmatrix} L_{II} & 0 \\ L_{BI} & L_{BB} \end{bmatrix} \begin{bmatrix} L_{II}^T & L_{BI}^T \\ 0 & L_{BB}^T \end{bmatrix}, \quad (7)$$

which means the Cholesky factorization for the Dirichlet problem is already given by the upper-left block: $A_{II} = L_{II}L_{II}^T$. All subsequent problems can then be solved (via backsubstitution) at a small fraction of the factorization cost—since BFF solves *only* Poisson problems, the computational bottleneck is the single factorization of A. (In practice one needs to be careful about reordering; see Sec. 7.4 for details.) Note that no corresponding treatment is known for LSCM/SCP despite repeated A blocks—the closest proposal entails either dense factorization, or else iterative solvers with no amortized gains from prefactorization [Alexa and Wardetzky 2011, Section 5.2].

4.2 Cherrier Formula

The change in curvature under a conformal mapping has a close relationship with the scale factor *u*. For domains without boundary this relationship is captured by the *Yamabe problem* [Aubin 1998, Chapter 5], but our method depends critically on additional boundary conditions studied by Pascal Cherrier [1984]. In particular, for a conformal map $f : M \to \widetilde{M}$ between any two surfaces,

$$\begin{aligned} \Delta u &= K - e^{2u}\widetilde{K} & \text{on} \quad M \\ \frac{\partial u}{\partial n} &= \kappa - e^u \tilde{\kappa} & \text{on} \quad \partial M \end{aligned} \quad (8)$$

where Δ is the Laplacian on *M*, and $\widetilde{K}, \tilde{\kappa}$ are the new curvatures on $\widetilde{M}$. Although the equation $\Delta u = K - e^{2u}\widetilde{K}$ is standard, its boundary conditions have been largely neglected by conformal flattening algorithms, which instead optimize discrete variables without reference to an underlying continuous equation [Kharevych et al. 2006; Springborn et al. 2008], or consider only the special case where $\tilde{\kappa}$ is a delta distribution [Bunin 2008; Myles and Zorin 2013]. We address the general case by multiplying Eqn. 8 through by *dA* and *ds* (*resp.*) to obtain a linear relationship between *densities*:

$$\Delta u \, dA = KdA - \widetilde{K}d\tilde{A}, \quad (9)$$

$$\frac{\partial u}{\partial n} ds = \kappa ds - \tilde{\kappa}d\tilde{s}. \quad (10)$$

Integrating Eqn. 9 over dual cells hence yields a linear relationship

$$Au = \Omega - \widetilde{\Omega}, \quad (11)$$

where $\Omega, \widetilde{\Omega} : V \to \mathbb{R}$ are the source and target angle defects (see Sections 3.3.1 and 4.1); for flattening, $\widetilde{\Omega} = 0$. Likewise, integrating Eqn. 10 over dual boundary edges e_i yields the linear equation

$$h = k - \tilde{k}, \quad (12)$$

where h is the discrete Neumann data, and k, $\tilde{k}$ are discrete boundary curvatures on the domain and target. Note that (as discussed in Ben-Chen *et al.* [2008, Section 2.3]) one does *not* obtain exact target curvatures if u values are used to rescale edge lengths (*à la* CETM); in BFF we take an alternate route where angles are obtained directly from the discrete Cherrier formula, and exactly satisfy necessary conditions for closure of the target boundary loop (Proposition 1) and realization of target cone angles (Proposition 2).

finite difference   Poincaré-Steklov

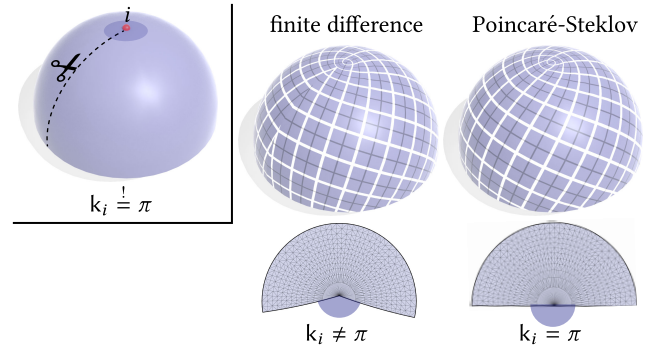$k_i \overset{!}{=} \pi$

$k_i \neq \pi$ $k_i = \pi$

Fig. 5. Accurate evaluation of boundary data is essential for achieving the correct boundary shape. Here we attempt to prescribe an angle π at vertex *i* of a cut hemisphere using either a simple finite difference formula *(left)* or a Poincaré-Steklov operator *(right)* to obtain Neumann data; only the latter scheme yields the correct angle, producing a seamless map.

4.3 Poincaré-Steklov Operators

A given solution to an elliptic boundary value problem (like Poisson) can often be explained by several different types of boundary conditions—for instance, a harmonic function is uniquely determined by either its values (Dirichlet) or normal derivatives (Neumann) along the boundary. In general, a *Poincaré-Steklov operator* maps boundary data from one solution to alternative boundary data that yields an identical solution. We require two such operators: the *Dirichlet-to-Neumann map* for a Poisson equation, and the *Hilbert transform* for the Cauchy-Riemann equation.

4.3.1 Dirichlet to Neumann. Given Dirichlet boundary values g for a discrete Poisson equation, we seek Neumann values h that yield the same solution. One idea is to simply solve the Dirichlet problem, then evaluate the normal derivative directly—*e.g.*, dot unit normals with the solution gradient, then integrate over dual boundary edges e_i. In practice, however, this approach yields poor behavior (see Figure 5, *center*). A more principled approach is to solve for Neumann data h that exactly reproduces the discrete Dirichlet solution. For a Laplace problem (*i.e.*, $\phi = 0$) solving Equations 5 and 6 for h entails evaluating the so-called *Schur complement* of A:

$$h = (A_{IB}^T A_{II}^{-1} A_{IB} - A_{BB})g. \quad (13)$$

For a Poisson equation with nonzero source term ϕ, the Dirichlet-to-Neumann map becomes an *affine* operator given by the expression

$$\Lambda_\phi g := \phi_B - A_{IB}^T A_{II}^{-1}(\phi_I - A_{IB}g) - A_{BB}g. \quad (14)$$

In practice this map can be evaluted via a single linear solve involving the (prefactored) matrix A_{II} together with a sequence of basic matrix operations, as detailed in Alg. 3. In effect, Λ_ϕ solves a Dirichlet-Poisson equation, then takes the difference between the source term ϕ and the Laplacian of the Poisson solution at each boundary node. Beyond improved numerical behavior (Figure 5, *right*), this discretization provides a useful invariant in the context of conformal flattening: discrete curvatures k computed via Eqn. 12 always sum to *exactly* 2π (App. A), automatically satisfying a necessary condition for integrability (Sec. 4.5).

4.3.2 Neumann to Dirichlet. The reverse direction is more straight-forward: one can simply solve the Neumann-Poisson equation (Eqn. 5) for a and read off the boundary values g := a_B (Alg. 4). Since the solution is determined only up to a constant (Sec. 4.1), we denote the Neumann-to-Dirichlet map by the pseudoinverse $\Lambda_\phi^\dagger$.

4.3.3 Hilbert Transform. On a disk-like domain, the *Hilbert transform* $\mathcal{H}$ maps the tangential derivative of a harmonic function a to the normal derivative of its harmonic conjugate b, providing boundary data for a holomorphic function $f = a + bi$ (Sec. 3.2.1). Notions of conjugacy have been studied for a wide variety of discretizations [Bobenko and Günther 2015; Mercat 2001; Polthier 2000; Weber and Gotsman 2010]; in our context, we seek a pair of standard piecewise linear functions with degrees of freedom at vertices. The basic idea is to fix a and solve for the function b that minimizes the *least-squares conformal energy* E_C, which measures the failure of f to satisfy Cauchy-Riemann [Lévy et al. 2002]. This energy can also be expressed as the Dirichlet energy of f minus the area of its image [Hutchinson 1991], which in matrix form becomes

$$E_C(a, b) = \begin{bmatrix} a^T & b^T \end{bmatrix} \begin{bmatrix} A & U \\ U^T & A \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}, \qquad (15)$$

where U encodes the signed area of the boundary polygon:

$$a^T U b := \tfrac{1}{2} \sum_{ij \in \partial M} a_j b_i - a_i b_j \qquad (16)$$

(see [Mullen et al. 2008, Section 2.2]). Minimizing E_C with fixed a therefore amounts to solving the Neumann-Laplace equation $Ab = -U^T a$. The Neumann boundary data $h = U^T a$ can be obtained by differentiating Eqn. 16 with respect to b, yielding the simple expression

$$h_j := \tfrac{1}{2}(a_k - a_i), \qquad (17)$$

for any three consecutive vertices i, j, k along the boundary. Notably, this expression looks like a naïve finite difference—the variational interpretation above verifies that it nonetheless yields a solution that is "as conjugate as possible" in the least-squares sense.

4.4 Interpolation

Suppose we want to extend a given function $\tilde{\gamma} : \partial M \to \mathbb{C}$ over the interior of the domain, *i.e.*, find a map $f : M \to \mathbb{C}$ such that $f|_{\partial M} = \tilde{\gamma}$. A simple strategy is to independently interpolate each coordinate of $\tilde{\gamma}$ by a harmonic function, *i.e.*, solve a pair of Laplace problems

$$\Delta a = 0 \quad \text{s.t.} \quad a|_{\partial M} = \text{Re}(\tilde{\gamma}),$$
$$\Delta b = 0 \quad \text{s.t.} \quad b|_{\partial M} = \text{Im}(\tilde{\gamma}).$$

If $\tilde{\gamma}$ is already compatible with some holomorphic map f, the Hilbert transform $\mathcal{H}$ from the previous section provides a different strategy:

(1) *(Harmonic extension.)* Solve $\Delta a = 0$ s.t. $a|_{\partial M} = \text{Re}(\tilde{\gamma})$.
(2) *(Harmonic conjugation.)* Solve $\Delta b = 0$ s.t. $\frac{\partial b}{\partial n} = \mathcal{H}a$,

where the discrete Neumann data for the latter problem is computed via Eqn. 17. These two strategies provide different algorithmic invariants: with the former, f exactly interpolates $\tilde{\gamma}$ but is not guaranteed to be exactly holomorphic; with the latter f is holomorphic but may not exactly interpolate both components of $\tilde{\gamma}$. When $\tilde{\gamma}$ comes from a holomorphic map they will coincide; in practice, each is best-suited to different applications, as discussed in Sec. 6.
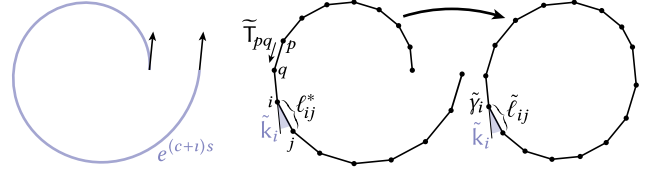
Fig. 6. Simply ensuring that curvature integrates to 2π is insufficient to guarantee that a boundary loop closes *(left)*. We minimally adjust the desired edge lengths ℓ^* to obtain a nearby closed loop with identical angles $\tilde{k}$ *(right)*.

4.5 Curve Integration

A key step in BFF is recovering a closed boundary curve $\tilde{\gamma}$ from given curvature and length data. In the smooth setting this data can be integrated directly, but in the discrete case a small amount of discretization error prevents closure—we therefore seek a closed curve that approximates the given data. Directly optimizing vertex positions leads to a difficult nonlinear problem; we instead solve an easy convex problem that closes the curve by minimally adjusting length. In the smooth setting, we formulate this problem by considering the cumulative curvature

$$\varphi(t) := \int_0^t \kappa(s)\, ds.$$

We then construct unit tangents $T(s) := e^{i\varphi(s)}$ along the boundary and solve the problem

$$\min_{r:\partial M \to \mathbb{R}} \tfrac{1}{2} \int_0^{2\pi} (r(s) - 1)^2\, ds \quad \text{s.t.} \quad \int_0^{2\pi} r(s)T(s)\, ds = 0 \quad (18)$$

If κ already describes a closed loop then we recover the solution $r \equiv 1$; otherwise, r minimally adjusts the speed of the curve such that it closes. In either case, we obtain the final curve by integrating the scaled tangents: $\tilde{\gamma}(t) := \int_0^t r(s)T(s)\, ds$.

4.5.1 Discretization. To discretize Eqn. 18, let $\tilde{k}, \ell^*$ specify the desired exterior angles and edge lengths (Figure 6, *right*). We seek a polygon with vertices $\tilde{\gamma}_i \in \mathbb{C}$ that exactly achieves these angles and closely matches the lengths. We first compute cumulative angles

$$\varphi_p := \sum_{i=1}^{p-1} \tilde{\kappa}_i$$

and target tangents $\widetilde{T}_{ij} := (\cos \varphi_i, \sin \varphi_i)$. Eqn. 18 then becomes

$$\min_{\tilde{\ell}:B \to \mathbb{R}} \tfrac{1}{2} \sum_{ij \in \partial M} \ell_{ij}^{-1} |\tilde{\ell}_{ij} - \ell_{ij}^*|^2 \quad \text{s.t.} \quad \sum_{ij \in \partial M} \tilde{\ell}_{ij} \widetilde{T}_{ij} = 0. \quad (19)$$

If $N \in \mathbb{R}^{|B| \times |B|}$ is a diagonal mass matrix with entries $N_{ii} = 1/\ell_{ij}$ and we pack the two coordinates of each unit tangent into a matrix $\widetilde{T} \in \mathbb{R}^{2 \times |B|}$, then the optimal lengths are given by

$$\tilde{\ell} = \ell^* - N^{-1}\widetilde{T}^T (\widetilde{T}N^{-1}\widetilde{T}^T)^{-1} \widetilde{T}\ell^*. \qquad (20)$$

Note that $\widetilde{T}N^{-1}\widetilde{T}^T$ is just a 2×2 matrix which costs virtually nothing to build and invert. Final vertex positions are recovered via the cumulative sums $\tilde{\gamma}_p := \sum_{i=1}^{p-1} \tilde{\ell}_{ij}\widetilde{T}_{ij}$. In principle the new lengths $\tilde{\ell}_{ij}$ could become negative, but in practice we do not observe this behavior: typical values for $\tilde{\ell}_i/\ell_i^*$ are in the range $1 \pm .001$.
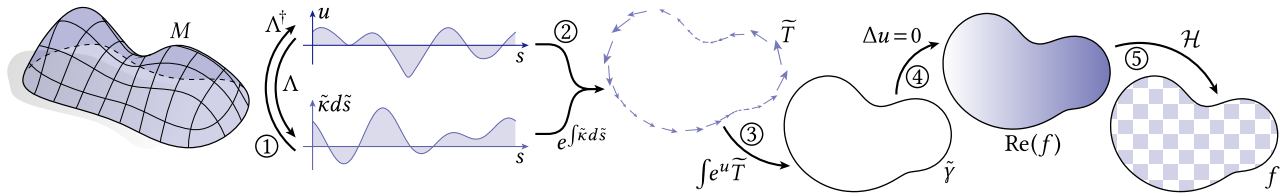
Fig. 7. Overview of the basic BFF algorithm. ① Given a surface M and either target scale factors u or target curvature density $\tilde{\kappa} d\tilde{s}$ along the boundary, the complementary quantity is obtained via the Dirichlet-to-Neumann map Λ. ② Curvature density is integrated to obtain unit tangents $\widetilde{T}$. ③ Integrating rescaled tangents $e^u \widetilde{T}$ yields the target boundary curve $\tilde{\gamma}$. ④ The real component of $\tilde{\gamma}$ is extended harmonically. ⑤ The Hilbert transform $\mathcal{H}$ provides the imaginary coordinate, and hence the final flattening $f : M \to \mathbb{C}$.

5 ALGORITHM

We now describe the basic boundary first flattening algorithm using tools from Sec. 4, as summarized in Figure 7.

Input. A triangle mesh with disk topology and *either* (i) desired scale factors u *or* (ii) target exterior angles $\tilde{k}$ at boundary vertices. In the latter case, angles must sum to 2π.

Output. A piecewise linear map f : V $\to \mathbb{C}$ which approximates a smooth conformal map with the given boundary data (u or $\tilde{k}$).

Algorithm.

I. Compute complementary boundary data (Sec. 4.2):
   - If scale factors u were given, compute compatible angles:
     $$\tilde{k} \leftarrow k - \Lambda_\Omega u \qquad \text{(Sec. 4.3.1)}.$$
   - If angles $\tilde{k}$ were given, compute compatible scale factors:
     $$u \leftarrow \Lambda_\Omega^\dagger (k - \tilde{k}) \qquad \text{(Sec. 4.3.2)}.$$

II. Construct a closed loop $\tilde{\gamma}$ that exhibits exterior angles $\tilde{k}$ and approximates edge lengths $\ell_{ij}^* := e^{(u_i + u_j)/2} \ell_{ij}$ (Sec. 4.5).

III. Compute the holomorphic extension f of $\tilde{\gamma}$ (Sec. 4.4).

Step I uses the Cherrier formula to "explain" the provided boundary data: if scale factors were given, what must the curvature look like under a conformal flattening (and vice versa)? The map Λ_Ω (or $\Lambda_\Omega^\dagger$) is used to evaluate this formula; using a source term Ω corresponds to setting the target Gaussian curvature to $\widetilde{\Omega} = 0$, *i.e.*, *flattening*. Since the resulting boundary data is compatible with some conformal flattening, the exterior angles $\tilde{k}$ and scaled boundary lengths ℓ^* already describe a valid closed loop—modulo a small amount of discretization error which is accounted for by the integration procedure in Step II. (Here we scale ℓ by the mean conformal factor *à la* CETM, though any consistent approximation will work.) Since length adjustments during integration are very small, the resulting curve $\tilde{\gamma}$ remains extremely close to the boundary of some conformal map. Therefore, in Step III a holomorphic extension of *either* coordinate function Re($\tilde{\gamma}$) or Im($\tilde{\gamma}$) will yield an accurate approximation of a conformal map that closely matches the input data—as verified by numerical experiments in Sections 6 and 7.

The overall cost is one factorization of a real $|V| \times |V|$ cotan-Laplace matrix, followed by three backsolves: one to evaluate the map Λ or $\Lambda^\dagger$ in Step I; two to compute the holomorphic extension in Step III. All other operations require only $O(|B|)$ work, involving simple closed-form expressions evaluated at boundary vertices. (See Sec. 7 for detailed performance analysis.)

6 APPLICATIONS

We now use the core BFF algorithm to solve several problems in surface parameterization. The basic question is how to construct appropriate boundary data (scale factors or curvatures) for each task.

6.1 Automatic Parameterization

In the absence of user-specified criteria, a natural choice of conformal flattening is the one with minimal area distortion. Springborn *et al.* [2008, App. E] show that such a map is obtained by prescribing zero scale factors along the boundary ($u|_{\partial M} = 0$). Figures 1, 2, and 14 show results computed via BFF, which are virtually indistinguishable from those produced by CETM (albeit at far lower cost) and respect features like symmetry even better than SCP (Figure 2). These boundary conditions provide the baseline for later comparisons with existing automatic methods (Sec. 7).

6.2 Direct Editing

A natural way to edit conformal flattenings with BFF is to directly manipulate the angles θ_{ij} (relative to the real axis) or length ℓ^* of target edges $ij \in \partial M$. These values can then be easily converted to curvatures $\tilde{k}$ and scale factors u per boundary vertex (*resp.*). In particular the (integrated) curvatures are simply

$$\tilde{k}_i = \theta_{i,i+1} - \theta_{i-1,i}.$$

Converting target lengths ℓ_{ij}^* (per boundary edge) to target scale factors u_i (per boundary vertex) is more subtle. One idea is to solve the linear system $u_i + u_j = 2 \log \ell_{ij}^* - 2 \log \ell_{ij}$ for scale factors u that satisfy discrete conformal equivalence [Springborn et al. 2008]. However, this system does not always have a solution; hence, even metric-based methods like CETM and CMPS cannot (in general) *exactly* prescribe boundary lengths. We instead use a straightforward numerical approximation: compute scale factors $u_{ij} := \log(\ell_{ij}^*/\ell_{ij})$ per boundary edge, then integrate over dual boundary edges to get

$$u_j = (\ell_{ij}^* u_{ij} + \ell_{jk}^* u_{jk})/(\ell_{ij}^* + \ell_{jk}^*),$$

where i, j, and k are consecutive vertices along the boundary.

As an experiment, we use a spline-based curve editor to nonrigidly pack surface charts into a texture atlas (Figure 8). At each vertex, boundary values are sampled from a real-valued Catmull-Rom spline (and normalized to 2π in the case of curvatures). The Dirichlet-to-Neumann map Λ is applied to the current boundary data to switch between angle- and length-based editing.
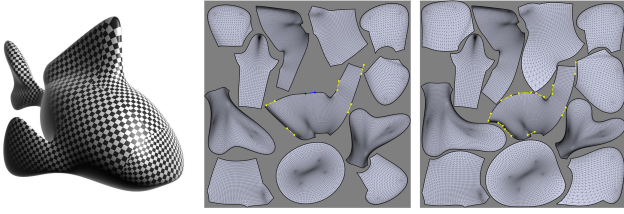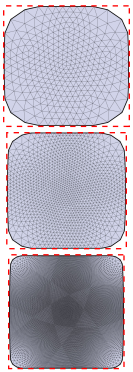
Fig. 8. Unlike conventional tools, our method can be used to interactively and nonrigidly tweak a texture layout while remaining conformal (here aiming for greater use of texture area).

6.3 Sharp Corners

For target shapes with sharp corners (like a rectangle) the standard BFF procedure can exhibit undesirable rounding (see inset, *top*). Here we can replace the holomorphic extension in Step III with a simple harmonic extension (Sec. 4.4), thereby interpolating the polygon $\tilde{\gamma}$ reconstructed in Step II, and *exactly* reproducing the requested angles. Since $\tilde{\gamma}$ approximates the boundary of a conformal map, both procedures still converge to a holomorphic function under refinement (see inset and Figure 9, *bottom*). Simpler algorithms for mapping to polygons can only handle special cases like rectangles [Zeng et al. 2008, Section 4.1] or are limited to straight edges [Driscoll and Trefethen 2002]; compare with the piecewise curved boundary in Figure 9, *top right*. Note that with piecewise linear maps it is impossible for *any* algorithm to completely eliminate angle distortion near corners, since the initial and target angle sums will differ even under refinement.
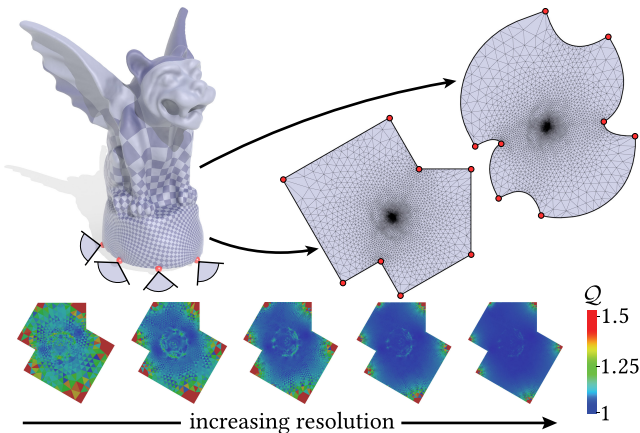

Fig. 9. *Top left:* Given a collection of points and angles on the boundary, we can map to a region with sharp corners and either straight *(top center)* or curved *(top right)* edges. *Bottom:* This map converges to a perfectly conformal map under refinement.
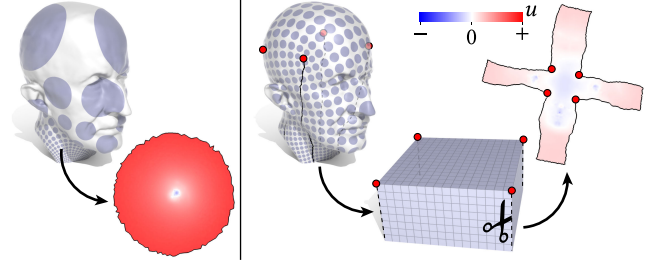
Fig. 10. Uneven area distortion (left) can be mitigated by first mapping to a surface that is flat except at a collection of *cone points* (in red), then cutting and unfolding this surface into the plane. BFF avoids the intermediate metric by directly computing boundary data for the final mapping.

6.4 Cone Parameterization

A powerful technique for mitigating area distortion (Figure 10) is to first map to a *cone surface*, which is flat ($K = 0$) away from a collection of isolated *cone points* [Kharevych et al. 2006]. After cutting through these points, it can be flattened into the plane without further distortion (Figure 10, *right*). A variety of strategies are available for picking cones [Ben-Chen et al. 2008; Kharevych et al. 2006; Springborn et al. 2008]; we assume they have been specified by the user as target curvatures Θ_i per interior vertex (mostly zero). For a closed surface of genus g, Θ must sum to $2\pi(2-2g)$, by Gauss-Bonnet. To compute a cone flattening of an initial surface M_0, we:

(1) Solve the Cherrier problem for u, with source term $\Omega - \Theta$.
(2) Cut M_0 into a disk M via a cut passing through all cones.
(3) Apply Alg. 1, prescribing u values from Step 1 along ∂M.

Springborn et al. [2008, Section 3.3] describe one possible cutting procedure. To obtain a seamless map, we associate only a *single* length degree of freedom $\tilde{\ell}$ with each pair of corresponding cut edges in Eqn. 18; angles are automatically complementary, by Proposition 2, and a harmonic extension (*à la* Sec. 6.3) ensures this data is exactly preserved in the final layout. The resulting maps are nearly identical to those produced by CETM (Figure 11), though we do not exactly preserve length cross ratios. Unlike CPMS/CETM we never compute (nor need to enforce integrability of) rescaled edge lengths, and can edit cone angles without performing additional factorization; unlike CPMS we guarantee by construction that the map is seamless. Moreover, since the Laplace matrix is modified only along the cut, fast updates of cone *locations* might also be achieved via low rank updates—see for example Essid & Solomon [2017].

6.5 Uniformization

A popular use of conformal maps is to provide a common reference domain for comparative data analysis [Koehl and Hass 2015; Lipman and Daubechies 2011]. For surfaces with disk topology, the *uniformization theorem* guarantees the existence of conformal maps to the unit circular disk—any such map has constant boundary curvature $\kappa = 1$. However, prescribing a constant curvature *density* yields a shape that is merely convex rather than circular. Instead, we use a simple fixed-point scheme: if the target curvature $\tilde{\kappa}$ were equal to 1, then the target curvature density would be $\tilde{\kappa}d\tilde{s} = d\tilde{s} = e^u ds$, *i.e.,* just the new length density. Hence, at the nth iteration we prescribe
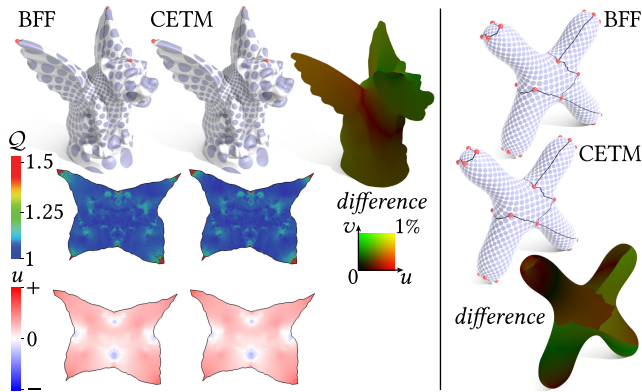
Fig. 11. *Left:* cone parameterizations produced via BFF are virtually identical in quality to those produced by more expensive nonlinear methods like CETM: in addition to similar angle distortion $\mathcal{Q}$ and scale distortion u, the map f itself differs by less than 1% relative to the diameter of the image. *Right:* example with many cone points.

target angles proportional to the most recent dual edge lengths:

$$\tilde{\mathsf{k}}_i^n \leftarrow 2\pi \tilde{\ell}_i^{n-1} / \sum_{i \in \mathsf{B}} \tilde{\ell}_i^{n-1}.$$

We then compute a conformal flattening and repeat. To stabilize this process we average with the previous guess (*i.e.*, $\tilde{\mathsf{k}}^n \leftarrow \frac{1}{2}(\tilde{\mathsf{k}}^n + \tilde{\mathsf{k}}^{n-1})$), using the discrete geodesic curvatures of the input surface as our initial guess. We also find that a harmonic rather than holomorphic extension (Sec. 6.3) yields better results, especially for domains with jagged boundaries. In practice we always converge in about 10 iterations—since each iteration involves only backsubstitution the total cost is similar to a single LSCM solve, but produces results nearly identical to nonlinear methods (as depicted in Figure 12). A canonical map can be found by picking the Möbius transformation that best balances area distortion, *à la* [Springborn 2005].

6.6 Arbitrary Curves

We can map a surface to an arbitrary target shape using a similar strategy: since a plane curve is determined by its curvature $\tilde{\kappa}$ (up to rigid motion), we iteratively prescribe the curvature density $\tilde{\kappa}e^{u^{n-1}} ds$, where u^{n-1} is our most recent guess for the scale factor. In the discrete case, let $\gamma^*(s) : [0, L] \to \mathbb{C}$ be a desired closed, arc-length parameterized curve; let $s_i := \sum_{k=1}^i \tilde{\ell}_{k,k+1}^n$ be the cumulative sum of boundary edge lengths at the current iteration, and let $S := \sum_i s_i$ be the total length. We first sample γ^* to obtain a polygon with vertices $z_i := \gamma^*((L/S)s_i)$, *i.e.*, at intervals proportional to our most recent edge lengths. We then compute the exterior angles of the sampled curve $\tilde{\mathsf{k}}_i = \arg((z_{i+1} - z_i)/(z_i - z_{i-1}))$. Since the sample points s_i are determined using the most recent length density, these angles provide an estimate for the desired curvature density. Empirically, this procedure rapidly converges to the target shape (see Figure 13). Basic as this functionality may seem, it is not provided by any previous conformal flattening algorithm (linear or nonlinear)—the closest comparison is the recent method of Segall and Ben-Chen [2016] for in-plane conformal deformations.
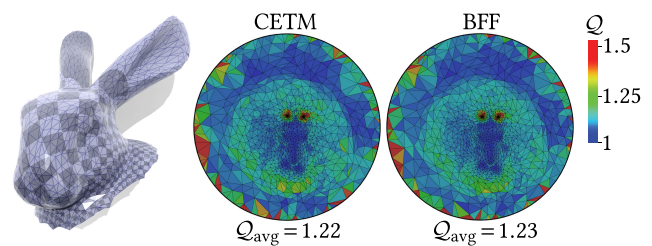

Fig. 12. Mapping to the unit disk. Even on a fairly coarse mesh of 3k triangles *(left)* we achieve results virtually indistinguishable from nonlinear methods.

7 EVALUATION AND COMPARISONS

Here we consider the numerical quality and runtime performance of both linear and nonlinear methods for boundary-controlled conformal flattening, including modifications of existing linear schemes. To help avoid erroneous comparisons, LSCM, SCP, LinABF, CP, CPMS, and CETM were independently implemented by both authors, and compared with reference implementations wherever possible. Figure 14 confirms that all implementations of boundary-controlled methods produce similar results (albeit at very different costs).

7.1 Modified Linear Methods

Though previous linear conformal flattening methods do not explicitly address boundary control, it is natural to ask whether we simply need to modify their boundary conditions. Below we explore such modifications for two linear methods (CPMS and LinABF).

7.2 Boundary-Controlled CPMS

Like BFF, CPMS [Ben-Chen et al. 2008] employs a Yamabe-type equation (Sec. 4.2) to obtain scale information. There are however several key differences. First, CPMS does not provide direct control over boundary shape: boundary vertices "absorb" curvature via a process that entails a Poisson solve per boundary vertex—even with prefactorization these solves become quite expensive (Ben-Chen et al. [2008, Section 4.2]). Second, it uses a different layout strategy: whereas BFF need only ensure that boundary data describes a closed loop (Sec. 4.5), CPMS seeks edge lengths that describe a closed, flat surface. Rather than satisfy this condition exactly (*à la* CETM), best-fit vertex positions are found via a least-squares problem [Sheffer
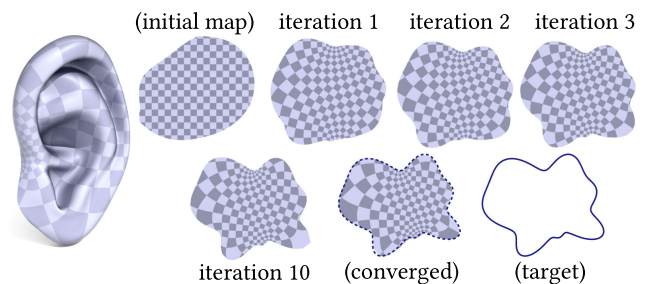

Fig. 13. BFF can be used to flatten a surface *(left)* directly onto a target shape *(bottom right)* via a simple iterative procedure. The combined cost of all iterations is not much more than the cost of the initial map.
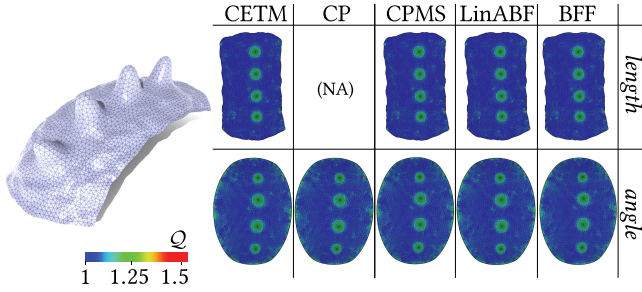
Fig. 14. For ideal meshes and smooth boundary data, all five boundary-controlled methods produce near-identical results—albeit with dramatically different update costs. Here we flatten an optimal Delaunay triangulation using either isometric lengths *(top)*, or uniform angles $\tilde{k}_i := \ell_i$ *(bottom)*. Quasi-conformal error $\mathcal{Q}_{avg}$ was identical (within 0.001) across all methods.

Fig. 15. Hemisphere mapped to an "L" shape. Though existing linear methods can be modified to incorporate angle control, they may exhibit significant artifacts—even on meshes with fairly mild imperfections *(left)*. In contrast, exact angle preservation is an algorithmic invariant of BFF.

et al. 2005] which cannot be prefactored since the matrix itself depends on the augmented lengths. We can of course add boundary control by incorporating Eqn. 8: to control angles we apply the Dirichlet-to-Neumann map (as in Step I of BFF); to control length we simply set Dirichlet boundary values for u. For nice meshes and smooth boundary data this strategy works well (Figure 14) but in general may exhibit artifacts since the least-squares layout does not respect boundary constraints (Figure 15). The amortized cost of editing a map via BFF remains about 30x faster than CPMS with our boundary modification, or about 50x faster than the original method (due to numerous solves at boundary vertices)—see Figure 17.

7.3 Boundary-Controlled LinABF

In *angle-based flattening (ABF)* [Sheffer and de Sturler 2001] a near-flat metric is found by optimizing corner angles β; a least-squares layout then provides planar vertex positions approximating these angles. Zayer *et al.* [2007] linearize ABF, solving for angle *adjustments* ϵ relative to an initial guess β^0—this strategy yields results nearly identical to the original algorithm. We modify this approach in two ways. First, to prescribe exterior

angles $\tilde{\kappa}$ we simply add linear constraints $\sum \beta_i^{jk} = \pi - \tilde{\kappa}_i$ at each boundary vertex i. Second, to prescribe boundary lengths $\tilde{\ell}_{ij}$ (up to global scale) we incorporate the condition

$$\prod_{ijk} \sin \beta_k^{ij} / \sin \beta_j^{ki} = \tilde{\ell}_{i-1,i} / \tilde{\ell}_{i,i+1}$$

for each boundary vertex, *i.e.*, we use the law of sines to prescribe the *ratio* of consecutive edge lengths along the boundary (see inset). Taking the first-order approximation of the logarithm and substituting $\beta_i^0 + \epsilon_i$ for β_i in the final system (*à la* Zayer) then yields linear constraints. As with CPMS, this strategy works well for nice meshes (Figure 14) but can produce artifacts due to both linearization and the least-squares layout (Figure 15). Moreover, neither the least-squares matrix nor the larger $3|F| \times 3|F| \approx 6|V| \times 6|V|$ angle constraint matrix can be prefactored, yielding lower performance than other linear methods (Figure 17).
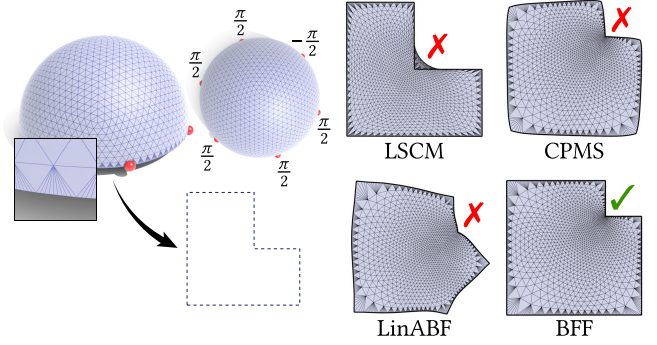
7.4 Quality and Performance

We measure angle distortion via *quasi-conformal error $\mathcal{Q}$*, which is the ratio of singular values of the mapping in each face [Sander et al. 2001]; $\mathcal{Q} = 1$ is ideal, and $\mathcal{Q}_{avg}$ denotes the area weighted average over the surface. Area distortion is measured via the log scale factor u, which is shifted to have zero mean.

Single-threaded performance was measured on a 3.3GHz Intel Core i7 with 16GB of memory. All methods were implemented in C++ using double precision. Linear methods use the supernodal factorization in CHOLMOD [Chen et al. 2008]. To extract the sub-factor L_{II} (Sec. 4.1) we compute a block-preserving reordering via `cholmod_l_camd`, which has no impact on factorization cost or factor density; overall we achieve a real-world 2x speedup over computing separate Neumann and Dirichlet factors. For CP we used MOSEK [ApS 2010]. For CETM we used Newton's method with backtracking line search [Boyd and Vandenberghe 2004, Algorithms 9.2 and 9.5], which in our tests performed better than L-BFGS or Newton trust region; during editing, using a previous solution as an initial guess did not significantly reduce the number of Newton iterations (especially for large changes to boundary data).
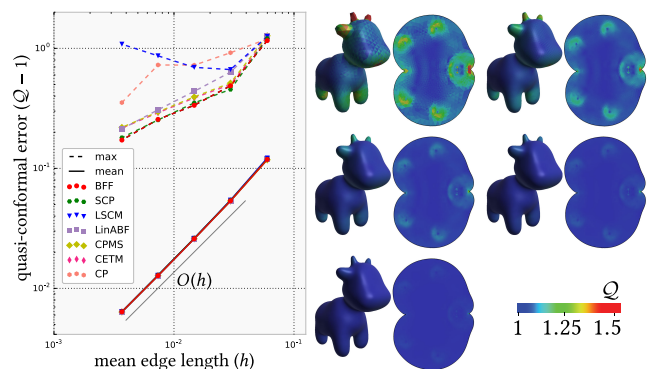

Fig. 16. For all methods (both linear and nonlinear) average angle distortion decreases linearly with respect to mean edge length h. BFF and SCP exhibit the smallest maximum distortion. *Right:* error distribution for BFF.
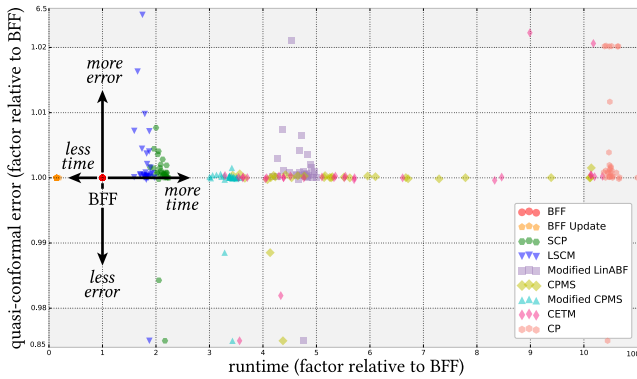
Fig. 17. Wall clock time and mean angle distortion relative to BFF, for a large collection of standard meshes (each dot is a single mesh). Overall, BFF achieves comparable error in less time than all other methods; prefactorization (orange) improves performance by about another order of magnitude.

Fig. 18. Surfaces with nontrivial topology can be parameterized by cutting them into one or more disks. Here we cut an annulus *(left)* and a torus *(right)* into disks along generators; mapping to rectangles (via prescribed corner angles of $\pi/2$) mostly eliminates seams.

Figure 16 suggests that our proposed strategy converges to a perfectly conformal map under refinement at the same (linear) rate as all other flattening methods. Figure 17 verifies that real-world timings agree with expected bottlenecks based on the size and number of matrices that must be factored [Botsch et al. 2005]. (Here LSCM/SCP/CPMS/CP use free boundary conditions; all other methods set $u|_{\partial M} = 0$.) The takeaway from these experiments is not that any method provides a big win in terms of accuracy (note the narrow vertical range in Figure 17) but simply that BFF does not sacrifice quality for speed. In particular, by using prefactorization BFF enables conformal flattenings to be edited about 30x quicker than the next-fastest boundary-controlled method (Modified CPMS), and about 50x quicker than the fastest previously-published method (CETM). This level of improvement provides a qualitative shift in the type of applications that can afford to use sophisticated conformal flattening—*e.g.*, interactive applications, or optimization for computational design [Konakovic et al. 2016].

7.5 Guarantees and Limitations

Boundary Data. Two guarantees provided by BFF are (i) exact realization of prescribed angles when using harmonic extension (as discussed in Sec. 6.3) and (ii) exact compatibility of lengths and angles along cuts (as discussed in Sec. 6.4). Neither LinABF nor CPMS can provide such guarantees since the least-squares layout step ignores any prescribed boundary data. Both CP and CETM will exactly satisfy angle constraints so long as optimization converges to an accurate solution. Exact length constraints are difficult for all methods, as discussed in Sec. 6.2.

Injectivity. A result of Floater [2003, Theorem 4.1] implies that maps produced by BFF are locally injective as long as (i) prescribed angles k are positive (describing a convex target), (ii) the discrete Laplacian A exhibits a maximum principle, and (iii) adjusted edge lengths $\tilde{\ell}$ are positive. The Delaunay condition is sufficient (but not necessary) to ensure condition (ii); length adjustments are typically miniscule and hence far from negative (Sec. 4.5.1).

In practice we find that flips are quite rare: for instance, 6 of 588 meshes in the SHREC 2011 database had one or two flipped triangles, with a total flipped area on the order of 10^{-5} when normalized to unit radius (see inset); for each mesh we automatically placed eight cones *à la* Ben-Chen et al. [2008]. Often such flips are easy to fix; one could also perform local Delaunay remeshing, or fall back to a more expensive injective method (see references in [Smith and Schaefer 2015]), though such methods may not provide the desired boundary control.
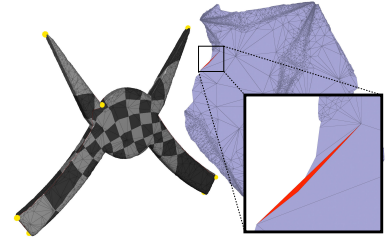
Topology. For topological spheres one can remove a single vertex star and map to the circular disk, then to the unit sphere via stereographic projection. For multiply-connected domains like the annulus, the Cherrier formula (Eqn. 8) is still valid but describes only an intrinsic flattening that may not admit a planar layout (consider cutting the tip from a circular cone); moreover, the Hilbert transform (Sec. 4.3.3) is valid only for disk-like domains. In practice, of course, BFF is used to flatten any surface by cutting it into one or more disks (Figs. 18, 8, and 10).

Target Geometry. BFF provides flattening only over Euclidean domains (possibly with cone singularities), unlike methods based on discrete Ricci flow [Jin et al. 2008; Springborn et al. 2008] which can target spherical or hyperbolic geometry. However, since the Cherrier formula is valid for any target curvature $\widetilde{K}$ it would be interesting to generalize the boundary-first strategy—here one might apply an iterative strategy akin to Sec. 6.5 to prescribe pointwise Gaussian curvature (rather than Gaussian curvature density).

8 ACKNOWLEDGEMENTS

REFERENCES

Marc Alexa and Max Wardetzky. 2011. Discrete Laplacians on General Polygonal Meshes. *ACM Trans. Graph.* 30, 4 (2011).

Pierre Alliez, Éric Colin de Verdière, Olivier Devillers, and Martin Isenburg. 2003. Isotropic Surface Remeshing. In *Proc. Shape Modeling International*.

Mosek ApS. 2010. The MOSEK optimization software. *http://www.mosek.com* (2010).

T. Aubin. 1998. *Some Nonlinear Problems in Riemannian Geometry.* Springer Berlin.

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

A. Bobenko, U. Pinkall, and B. Springborn. 2010. Discrete conformal maps and ideal hyperbolic polyhedra. *ArXiv e-prints* (2010).

A. I. Bobenko and F. Günther. 2015. Discrete complex analysis on planar quad-graphs. *ArXiv e-prints* (2015).

Mario Botsch, David Bommes, and Leif Kobbelt. 2005. Efficient Linear System Solvers for Mesh Processing. In *Proc. IMA Int. Conf. Math. Surf.* 62–83.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization.* Cambridge University Press.

Guy Bunin. 2008. A continuum theory for unstructured mesh generation in two dimensions. In *CAGD*, Vol. 25. 14–40.

Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. 2010. A Simple Geometric Model for Elastic Deformations. *ACM Trans. Graph.* 29, 4 (2010).

Renjie Chen, Ofir Weber, Daniel Keren, and Mirela Ben-Chen. 2013. Planar Shape Interpolation with Bounded Distortion. *ACM Trans. Graph.* 32, 4 (2013).

Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. 2008. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Trans. Math. Softw.* 35, 3 (2008).

Pascal Cherrier. 1984. Problèms de Neumann non linéaires sur les variétés riemanniennes. *Journal of Functional Analysis* 57 (1984), 154–206.

Keenan Crane. 2013. *Conformal Geometry Processing.* Ph.D. Dissertation. Caltech.

Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. 2013. Digital Geometry Processing with Discrete Exterior Calculus. In *SIGGRAPH Courses*.

Mathieu Desbrun, Mark Meyer, and Pierre Alliez. 2002. Intrinsic Parameterizations of Surface Meshes. *CG Forum* (2002).

T.A. Driscoll and L.N. Trefethen. 2002. *Schwarz-Christoffel Mapping.* Cambridge University Press.

Monty Essid and Justin Solomon. 2017. Quadratically-Regularized Optimal Transport on Graphs. *(in submission)* (2017).

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

X.D. Gu and S.T. Yau. 2008. *Computational Conformal Geometry.* International Press.

Monica K. Hurdal and Ken Stephenson. 2009. Discrete conformal methods for cortical brain flattening. *NeuroImage* 45, 1 (2009), 86–98.

John E. Hutchinson. 1991. Computing conformal maps and minimal surfaces. In *Theoretical and Numerical Aspects of Geometric Variational Problems.* 140–161.

Miao Jin, Xianfeng Gu, Feng Luo, and Junho Kim. 2008. Discrete Surface Ricci Flow. *IEEE Trans. Vis. Comp. Graph.* 14 (2008), 1030–1043.

Liliya Kharevych, Boris Springborn, and Peter Schröder. 2006. Discrete Conformal Mappings via Circle Patterns. *ACM Trans. Graph.* 25, 2 (2006), 412–438.

Jungwook Kim, James A. Hanna, Myunghwan Byun, Christian D. Santangelo, and Ryan C. Hayward. 2012. Designing Responsive Buckled Surfaces by Halftone Gel Lithography. *Science* 335, 6073 (2012), 1201–1205.

Patrice Koehl and Joel Hass. 2015. Landmark-free geometric methods in biological shape analysis. *Journal of The Royal Society Interface* 12, 113 (2015).

Mina Konakovic, 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).

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least Squares Conformal Maps. *ACM Trans. Graph.* 21, 3 (2002).

Y. Lipman and I. Daubechies. 2011. Conformal Wasserstein distances: comparing surfaces in polynomial time. *ArXiv e-prints* (2011).

Yaron Lipman, Vladimir G. Kim, and Thomas A. Funkhouser. 2012. Simple Formulas for Quasiconformal Plane Deformations. *ACM Trans. Graph.* 31, 5 (2012).

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven Gortler. 2008. A Local/Global Approach to Mesh Parameterization. In *Proc. Symp. Geom. Proc.* 1495–1504.

Feng Luo. 2004. Combinatorial Yamabe Flow on Surfaces. *Comm. Contemp. Math.* / 06, 05 (2004), 765–780.

Richard MacNeal. 1949. *The Solution of Partial Differential Equations by means of Electrical Networks.* Ph.D. Dissertation. Caltech.

Christian Mercat. 2001. Discrete Riemann Surfaces and the Ising Model. *Comm. Math. Phys.* 218, 1 (2001), 177–216.

Patrick Mullen, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. 2008. Spectral Conformal Parameterization. In *Proc. Symp. Geom. Proc.* 1487–1494.

Ashish Myles and Denis Zorin. 2013. Controlled-distortion Constrained Global Parametrization. *ACM Trans. Graph.* 32, 4 (July 2013).

Konrad Polthier. 2000. Conjugate Harmonic Maps and Minimal Surfaces. *Preprint No. 446, TU-Berlin, SFB 288* (2000).

Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. 2001. Texture Mapping Progressive Meshes. In *Proc. ACM/SIGGRAPH.* 409–416.

Rik Sarkar, Xiaotian Yin, Jie Gao, Feng Luo, and Xianfeng David Gu. 2009. Greedy routing with guaranteed delivery using Ricci flows.. In *IPSN.* 121–132.

Aviv Segall and Mirela Ben-Chen. 2016. Iterative Closest Conformal Maps between Planar Domains. *CG Forum* (2016).

Aviv Segall, Orestis Vantzos, and Mirela Ben-Chen. 2016. Hele-Shaw Flow Simulation with Interactive Control. In *In Proc. Symp. Comp. Anim.* 85–95.

A. Sheffer and E. de Sturler. 2001. Parameterization of Faceted Surfaces for Meshing using Angle-Based Flattening. *Engineering with Computers* 17, 3 (2001), 326–337.

Alla Sheffer, Bruno Lévy, Maxim Mogilnitsky, and Alexander Bogomyakov. 2005. ABF++ : Fast and Robust Angle Based Flattening. *ACM Trans. Graph.* 24, 2 (2005).

Alla Sheffer, Emil Praun, and Kenneth Rose. 2006. Mesh Parameterization Methods and Their Applications. *Found. Trends. Comput. Graph. Vis.* 2, 2 (2006), 105–171.

Jason Smith and Scott Schaefer. 2015. Bijective Parameterization with Free Boundaries. *ACM Trans. Graph.* 34, 4 (July 2015).

Boris Springborn. 2005. A Unique Representation of Polyhedral Types. Centering via Möbius Transformations. *Math. Z.* 249 (2005), 513–517.

Boris Springborn, Peter Schröder, and Ulrich Pinkall. 2008. Conformal Equivalence of Triangle Meshes. *ACM Trans. Graph.* 27, 3 (2008).

Ofir Weber and Craig Gotsman. 2010. Controllable Conformal Maps for Shape Deformation and Interpolation. *ACM Trans. Graph.* 29, 4 (2010).

Ofir Weber and Denis Zorin. 2014. Locally injective parametrization with arbitrary fixed boundaries. *ACM Trans. Comp. Sys.* 33, 4 (2014).

Rhaleb Zayer, Bruno Lévy, and Hans-Peter Seidel. 2007. Linear Angle Based Parameterization. In *Proc. Symp. Geom. Proc.* 135–141.

Wei Zeng, Lok Ming Lui, Xianfeng Gu, and Shing-Tung Yau. 2008. Shape Analysis by Conformal Modules. *Meth. Appl. Anal.* 15, 4 (2008), 539–556.

Zichun Zhong, Liang Shuai, Miao Jin, and Xiaohu Guo. 2014. Anisotropic Surface Meshing with Conformal Embedding. *Graph. Models* 76, 5 (2014), 468–483.

A   EXACT ANGLE SUMS

PROPOSITION 1. *For any given scale factors* u, *the complementary curvatures* $\tilde{k}$ *computed by Step I of BFF will always sum to exactly* 2π.

PROOF. Integrating the Cherrier formula over a boundary dual cell C_i yields $(Au)_i + h_i = \Omega_i$ (since $\widetilde{\Omega} = 0$ for a flattening). Since $\Omega_i = 0$ at boundary vertices (Sec. 3.3.1), the new boundary curvatures can be expressed as

$$\tilde{k}_i = k_i - h_i = k_i + (Au)_i = \int_{e_i} \kappa \, ds - \int_{\partial C_i \setminus e_i} \frac{\partial u}{\partial n} \, ds,$$

where e_i is the dual boundary edge at i. Their sum is then

$$\sum_i \int_{e_i} \kappa \, ds - \sum_i \int_{\partial C_i \setminus e_i} \frac{\partial u}{\partial n} \, ds = \int_{\partial M} \kappa \, ds + \int_{\partial D} \frac{\partial u}{\partial n} \, ds,$$

where D is the union of all interior dual cells (see inset)—note the change in orientation, and cancellation due to equal and opposite normal derivatives in adjacent cells. Applying Gauss-Bonnet to the first term and the divergence theorem to the second, we get

$$2\pi\chi - \int_M K \, dA - \int_D \Delta u \, dA,$$

where for a disk $\chi = 1$. But since $\Delta u = K$ (and curvature is nonzero only on D), we are left with just 2π.   □
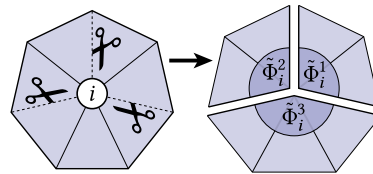
PROPOSITION 2. *Target angles* $\tilde{k}$ *computed as in Sec. 6.4 will sum to exactly the desired cone angles* Θ.

PROOF. Suppose our cut partitions an interior vertex i into several regions, and let $\tilde{\Phi}_i^p$ be the sum of interior angles at

i in the pth region (see inset). Recall that $\tilde{k} = k - h$ (Eqn. 12), hence

$$\sum_p \tilde{k}_i^p = \sum_p k_i^p - \sum_p \int_{\partial C_i^p \setminus e_i^p} \frac{\partial u}{\partial n} \, ds$$

Since $\tilde{k}$ and $\tilde{\Phi}$ are supplementary angles, we get

$$\sum_p (\pi - \tilde{\Phi}_i^p) = \sum_p (\pi - \Phi_i^p) - \int_{C_i} \Delta u \, dA,$$

where ∂C_i is the cell boundary on the original (uncut) mesh. Canceling π terms and recalling that for our cone parameterization $\Delta u = \Omega - \Theta$ (Sec. 6.4), we get the desired result:

$$\sum_p \tilde{\Phi}_i^p = (2\pi - \Omega_i) - (\Omega_i - \Theta_i) = 2\pi - \Theta_i.$$

$\square$

B PSEUDOCODE

Pseudocode for the basic BFF algorithm (Sec. 5) is included below. Boundary vertices are enumerated by cyclic indices $1, \ldots, |B|$ throughout. In lieu of vertex positions, the geometry of the input mesh is specified purely by its edge lengths ℓ, which is useful when the surface is not embedded. In practice, of course, it may be easier to compute quantities like angles directly from vertex positions. For an initial implementation it may be easiest to first separately factor the Dirichlet and Neumann Laplace matrices; extracting a subfactor (*à la* Eqn. 7) will subsequently improve performance.

Algorithm 1 BoundaryFirstFlattening$(M, \ell, [u|\tilde{k}])$

Input: A manifold triangle mesh $M = (B \subseteq V, E, F)$ with disk topology, edge lengths $\ell : E \to \mathbb{R}_{>0}$ satisfying the triangle inequality in each face, and *either* scale factors $u : B \to \mathbb{R}$ *or* exterior angles $\tilde{k} : B \to \mathbb{R}$ that sum to 2π.

Output: A flattening $f : V \to \mathbb{C}$.

1: $\beta \leftarrow$ InteriorAngles(M, ℓ)
2: $\Omega, k \leftarrow$ DiscreteCurvatures(M, β)
3: $A \leftarrow$ BuildLaplace(M, β)
4: $L \leftarrow$ CholeskyFactor(A)
5: **if** u was given **then**
6:     $\tilde{k} \leftarrow k -$ DirichletToNeumann$(A, L, -\Omega, u)$
7: **else**
8:     $u \leftarrow$ NeumannToDirichlet$(L, -\Omega, k - \tilde{k})$
9: **for each** $ij \in \partial M$ **do** $\ell_{ij}^* \leftarrow e^{(u_i + u_j)/2} \ell_{ij}$
10: $\tilde{\gamma} \leftarrow$ BestFitCurve$(M, \ell, \ell^*, \tilde{k})$
11: **return** ExtendCurve$(M, L, \tilde{\gamma})$

Algorithm 2 BuildLaplace(M, ℓ)

Input: A mesh M with edge lengths ℓ.

Output: A zero-Neumann Laplace matrix $A \in \mathbb{R}^{|V| \times |V|}$.

1: $A \leftarrow 0 \in \mathbb{R}^{|V| \times |V|}$ ▷ initialize an empty *sparse* matrix
2: **for each** $pqr \in F$ **do**
3:     **for each** $ijk \in \mathcal{C}(pqr)$ **do**   ▷ $\mathcal{C}$: circular shifts
4:        $A_{ii}, A_{jj} \mathrel{+}= \frac{1}{2}\cot(\beta_k^{ij})$
5:        $A_{ij}, A_{ji} \mathrel{-}= \frac{1}{2}\cot(\beta_k^{ij})$
6: **return** A

Algorithm 3 DirichletToNeumann(A, L, ϕ, g)

Input: Zero-Neumann Laplace matrix A and its factorization L, source term ϕ, and Dirichlet boundary data $g : B \to \mathbb{R}$.

Output: Neumann data $h : B \to \mathbb{R}$.

1: BackSolve$(L_{II}, a, \phi_I - A_{IB}g)$
2: **return** $\phi_B - A_{IB}^\mathsf{T}a - A_{BB}g$

Algorithm 4 NeumannToDirichlet(L, ϕ, h)

Input: Cholesky factor L of the zero-Neumann Laplace matrix, source term $\phi : V \to \mathbb{R}$, and Neumann data $h : B \to \mathbb{R}$.

Output: Dirichlet data $g : B \to \mathbb{R}$.

1: BackSolve$(L, a, \phi - [0; h])$
2: **return** a_B

Algorithm 5 BestFitCurve$(M, \ell, \ell^*, \tilde{k})$

Input: A disk (M, ℓ), desired edge lengths $\ell^* : B \to \mathbb{R}_{>0}$, and target exterior angles $\tilde{k} : B \to \mathbb{R}$ that sum to 2π

Output: Vertex positions $\tilde{\gamma} : B \to \mathbb{C}$.

1: $\widetilde{T} \leftarrow 0 \in \mathbb{R}^{2 \times |B|}$ ▷ dense matrix of tangents
2: $\varphi_{0,1} \leftarrow 0$ ▷ direction of first tangent
3: **for** $i = 1, .., |B| - 1$ **do**   ▷ walk along boundary
4:     $\varphi_{i,i+1} \leftarrow \varphi_{i-1,i} + \tilde{k}_i$ ▷ accumulate exterior angles
5:     $\widetilde{T}_{i,i+1} \leftarrow e^{\iota\varphi_{i,i+1}}$ ▷ put tangent in column of matrix
6: $N \leftarrow 0 \in \mathbb{R}^{|B| \times |B|}$ ▷ boundary mass matrix
7: **for each** $i \in B$ **do** $N_{ii} \leftarrow 1/\ell_i$ ▷ ℓ_i are dual lengths
8: $\tilde{\ell} \leftarrow \ell^* - N^{-1}\widetilde{T}^\mathsf{T}(\widetilde{T}N^{-1}\widetilde{T}^\mathsf{T})^{-1}\widetilde{T}\ell^*$ ▷ adjust lengths to close
9: $\tilde{\gamma}_1 \leftarrow 0 \in \mathbb{C}$ ▷ put first vertex at origin
10: **for** $i = 2, .., |B|$ **do**   ▷ for remaining vertices
11:     $\tilde{\gamma}_i \leftarrow \tilde{\gamma}_{i-1} + \tilde{\ell}_{i-1,i}\widetilde{T}_{i-1,i}$ ▷ accumulate scaled tangents
12: **return** $\tilde{\gamma}$

Algorithm 6 ExtendCurve$(M, L, \tilde{\gamma})$

Input: A mesh M with disk topology, Cholesky factor L of the zero-Neumann Laplace matrix, and a closed loop $\tilde{\gamma} : B \to \mathbb{C}$.

Output: A flattening $f : V \to \mathbb{C}$.

1: BackSolve$(L_{II}, a, -A_{IB} \operatorname{Re}(\tilde{\gamma}))$ ▷ harmonic extension
2: **for each** $i \in B$ **do** $h_i \leftarrow \frac{1}{2}(a_{i+1} - a_{i-1})$ ▷ Hilbert transform
3: BackSolve(L, b, h) ▷ harmonic conjugation
4: **return** $a + \iota b$

All remaining procedures involve either evaluating simple formulas, or calling standard library routines:

- InteriorAngles(M, ℓ) — computes the angles β at each triangle corner.
- DiscreteCurvature(M, β) — computes discrete Gauss and geodesic curvature using formulas from Sec. 3.3.1. (Note that $\Omega_i = 0$ at all boundary vertices $i \in B$.)
- CholeskyFactor(A) — returns sparse Cholesky factor L.
- BackSolve(L, x, b) — solves $Ax = b$ using the Cholesky factor of A.