DOI: 10.1111/cgf.70310

COMPUTER GRAPHICS *forum*
Volume 0 (2026), number 0, e70310

Real-Time Conformal Maps and Parameterizations

Q. Chang,[1] C. Gotsman[2] and K. Hormann[1]

[1]Università della Svizzera italiana, Lugano, Switzerland
{qingjun.chang, kai.hormann}@usi.ch
[2]New Jersey Institute of Technology, Newark, New Jersey, USA
gotsman@njit.edu

Abstract

We present a simple algorithm to conformally map between two simple and bounded planar domains based on the concept of harmonic measure, which is a conformal invariant. With suitable preprocessing, the algorithm is fast enough to compute all possible conformal maps (having three real degrees of freedom) between the two domains in real time in an interactive system, which seems to be the first to 'bring to life' the classical Riemann mapping theorem in this way. The same method can be used to conformally parameterize genus-0 manifold 3D surfaces with a boundary over arbitrary simple and bounded planar domains and interactively adjust the parameterization.

Keywords: conformal maps, harmonic measure, mesh parameterization

CCS Concepts: • Computing methodologies → Parametric curve and surface models; • Mathematics of computing → Interpolation;

1. Introduction

According to the (smooth) *Riemann mapping theorem* [Lan99], any simply connected bounded domain $\Omega_1 \subset \mathbb{R}^2$ with (piecewise) smooth boundary may be conformally mapped to any other simply connected bounded domain $\Omega_2 \subset \mathbb{R}^2$ with (piecewise) smooth boundary. This map $f : \Omega_1 \to \Omega_2$ has three real degrees of freedom, manifesting in the choice of a correspondence between two interior points $x_1 \in \operatorname{int}\Omega_1$ and $x_2 \in \operatorname{int}\Omega_2$ and a correspondence between two boundary points $y_1 \in \partial\Omega_1$ and $y_2 \in \partial\Omega_2$.

A conformal map has many useful mathematical properties. An important one is that locally it behaves like a simple similarity transformation and, as such, is convenient to describe as a complex function. In fact, a conformal map is just a classical *holomorphic* (or *analytic*) complex function whose derivative never vanishes in the domain.

Planar conformal maps have many important applications in multiple fields, such as science, engineering, geometry, and art, and they have been studied extensively by mathematicians. Alas, conformal maps can be described analytically only for very few canonical domains (e.g., the unit disk), thus we must resort to numerical methods for almost all other domains, and it turns out to be quite difficult to compute such a map. Especially challenging is mapping between domains that are significantly different from each other in shape or even when the shapes are similar, but the constraints are not. The distorted maps that result suffer from the *crowding* phenomenon [DP93], namely, extremely dense clustering of points on the boundary, making for a numerically sensitive scenario in which it is difficult to distinguish between the points and, more importantly, their ordering along the boundary. Consequently, beyond the mathematical investigations, numerical computation of conformal maps has been the subject of intense research for the last four decades.

Most of the numerical algorithms to compute conformal maps have focused on the accuracy of the result, which is important for engineering applications, thus are typically quite elaborate and employ sophisticated numerical techniques. The interested reader is referred to the book by Kythe [Kyt19] for a comprehensive review and comparison of some of the most popular numerical algorithms. For other applications, especially interactive ones in computer graphics and geometry processing, the accuracy of the result is less important and the primary goal is speed, namely, being able to compute the map as quickly as possible, the holy grail being real-time. Our work addresses this scenario: we describe an extremely efficient numerical algorithm to compute conformal maps between two simple domains. The key is some preprocessing of the two domains such that the subsequent computation of the map depending on the three

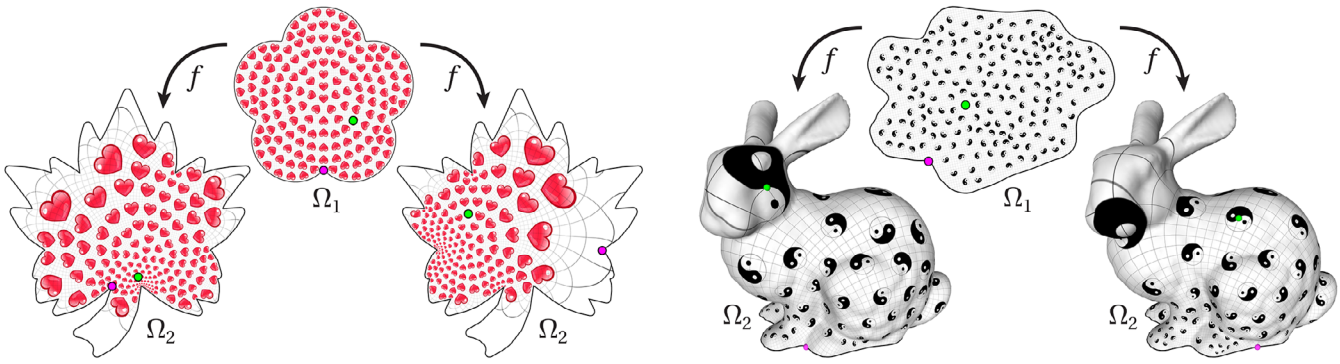

Figure 1: *Our interactive tool allows the user to explore the space of all conformal maps $f : \Omega_1 \to \Omega_2$ from a planar* cinquefoil *to a* maple leaf *by specifying and interactively changing two corresponding interior (green) and two corresponding boundary (magenta) points (left). It can also be used to compute all conformal parameterizations of the* Stanford bunny *with boundary (at its base) over an arbitrary planar domain (here the base itself) in real time (right).*

degrees of freedom is very quick and can be done in real time as the parameters are changed interactively (see Figure 4).

1.1. Related work

There are three main families of numerical methods to approximate a conformal map of a given planar domain Ω_1 to another planar domain Ω_2. In these approaches, the domain boundary is typically described in discrete form as an ordered sequence of points in the plane (or elements of $\mathbb{C}$), forming a simple polygon oriented counterclockwise, essentially a dense sample of the continuous $\partial\Omega_1$. The objective then becomes to position points on $\partial\Omega_2$ corresponding to those on $\partial\Omega_1$, defining the conformal map between the boundaries, which can then be extended into the interiors. The traditional approach maps an arbitrary simple domain to or from the unit disk $\mathbb{D}$, bounded by the unit circle, as it is sometimes easier to operate on the circle than on an arbitrary contour. A map between two arbitrary domains Ω_1 and Ω_2 may then be constructed using $\mathbb{D}$ as an intermediary, namely, computing the conformal maps $f_1 : \Omega_1 \to \mathbb{D}$ and $f_2 : \Omega_2 \to \mathbb{D}$. The function $f = f_2^{-1} \circ f_1$ is then the required one, since the inverse of a conformal map is also conformal, as is the composition of two conformal maps.

The finite element methods (FEM) represent Ω_1 by small mesh elements (typically triangles) and minimize a so-called 'conformal energy' for the images of these elements. For example, if Ω_1 is discretised into a set of small triangles forming a mesh, the conformal energy is a quadratic form on the coordinates of the images of the mesh vertices, involving the geometric Laplacian matrix of the mesh [DMA02, LPRM02, CG17]. The main challenge then becomes setting up the appropriate set of constraints which ensure that $\partial\Omega_1$ maps to $\partial\Omega_2$, the positional constraints $f(x_1) = x_2$ and $f(y_1) = y_2$ are satisfied, and solving the resulting constrained optimization problem, which is typically no longer convex [KLYY21, TZ22]. A notable advance was made by Dym et al. [DSL19], who realized that it may be better to use the equilateral triangle as an intermediary instead of the disk. The advantage is that the algorithm to compute a conformal map to the triangle can then be described as a sparse linear system of equations (essentially minimizing the Dirichlet energy of the interior discretised in the FEM manner), making for quite an efficient algorithm.

An alternative to the FEM methods are the boundary element methods (BEM), where the computation of the conformal map is done based on the boundaries alone. This is feasible thanks to the fact that a conformal (and harmonic) map of a bounded domain is uniquely determined by its boundary map. Indeed, according to *Cauchy's integral formula* [Lan99], any holomorphic function $f : \Omega \to \mathbb{C}$ can be expressed as the complex-valued boundary integral

$$f(z) = \frac{1}{2\pi \mathrm{i}} \oint_{\partial\Omega} \frac{f(w)}{w - z}\, \mathrm{d}w. \tag{1}$$

Within the BEM family are many classical iterative methods which allow the boundary map to evolve over time, converging to the final conformal map. Each iteration is typically the solution of a differential equation on the boundary. Well-known examples are the methods of Fornberg [For80] and Wegmann [Weg86], which map from an arbitrary contour to the circle. The Zipper algorithm [MR07] also belongs to the BEM family.

A third family of methods that similarly focus on the boundary are *Schwarz–Christoffel mappings* (see [DT02] and the references therein), which conformally map in the opposite direction: from the unit disk (or the upper half-plane) to a domain bounded by an arbitrary simple polygon. They are based on the famous *Schwarz–Christoffel formula* and require solving a set of nonlinear equations for the preimages of the polygon's vertices. This technique has led to a variety of sophisticated software packages, including the *SC Toolbox* for *MATLAB* [Dri25].

An efficient way to describe and control planar conformal maps is provided by *Green coordinates* [LLCO08], derived from Green's third integral identity or the equivalent *Cauchy–Green coordinates* [WBCG09], obtained by discretizing the integral in Equation (1). Segall and Ben-Chen [SBC16] describe an iterative algorithm, based on Cauchy–Green coordinates, for computing conformal maps between two domains, However, their Achilles heel is the inevitable crowding, which also causes the solution, in the case of

significant distortion, to be very noisy and to tangle (i.e., not necessarily respect the correct ordering of the points along the boundary), which is fatal.

Conformal maps originate in the planar setting, but the Riemann mapping theorem extends also to arbitrary Riemann surfaces, including 2-manifolds embedded in the ambient $\mathbb{R}^3$. This can be applied to the important problem of 3D mesh parameterization, where the 3D surface is to be conformally mapped to a planar domain. The seminal (and equivalent) algorithms by Lévy et al. [LPRM02] and Desbrun et al. [DMA02] compute an approximate conformal parameterization of a 3D mesh by solving a sparse linear system akin to that based on Dirichlet energy in the 2D FEM scenario, but apart from interpolation constraints at two boundary points, which essentially fix the global translation, rotation, and uniform scaling of the mapping, the user cannot control the shape of the planar parameter domain. The same is true for an improved variant of this approach [MTAD08], as well as for nonlinear, angle-based parameterization [SdS01, SLMB05] and its linearized version [ZLS07].

More boundary shape control can be achieved by conformal parameterization methods based on circle patterns [KSS06] or discrete conformal equivalence [SSP08]. They allow to specify metric data (lengths or angles) at the boundary, but involve nonlinear optimization. A similar approach, which allows to specify length or curvature density along the boundary, was described by Weber and Gotsman [WG10] and more recently by Sawhney and Crane [SC18]. The latter *boundary first flattening* (BFF) method comes with the advantage of involving only sparse linear systems and provides sophisticated interactive control over the shape of the parameter domain. However, parameterization over an arbitrary given target shape still requires an iterative procedure, akin to the method of Segall and Ben-Chen [SBC16].

1.2. A simple interactive tool

A simple design for an interactive tool to explore all conformal maps between two planar domains goes back to the idea of using the unit disk as an intermediate domain (see Figure 2). We first compute two arbitrary conformal maps $g_1 : \Omega_1 \to \mathbb{D}$ and $g_2 : \Omega_2 \to \mathbb{D}$ in a preprocessing step. During the online session, finding the conformal map $f : \Omega_1 \to \Omega_2$ with constraints $f(x_1) = x_2$ and $f(y_1) = y_2$ then essentially boils down to computing the images $a_i = g_i(x_i)$ and $b_i = g_i(y_i)$, $i = 1, 2$ of the constraints under the precomputed maps, and then defining f as $f = g_2^{-1} \circ \varphi_2^{-1} \circ \varphi_1 \circ g_1$, where

$$\varphi_i(z) = \frac{(a_i - z)(1 - \overline{a_i} b_i)}{(a_i - b_i)(1 - \overline{a_i} z)}$$

is the *Möbius transformation*, expressed in complex number form, of the unit disk to itself that maps a_i to 0 and b_i to 1. It is now easy to see that the desired constraints on f are satisfied.

While theoretically sound, this approach can suffer from significant numerical issues when implemented, due to the inevitable crowding incurred in g_1 and g_2. Consider, for example, the V-shaped domain Ω in Figure 3, designed such that the exact conformal map of Ω to the unit disk $\mathbb{D}$ is $g = h_1 \circ h_2$, where

$$h_1(u) = \frac{\tanh(u \operatorname{arctanh}(r))}{r}, \qquad h_2(u) = \frac{\mathrm{i}}{4}\left(1 - \sqrt{8\mathrm{i}u - 7}\right),$$

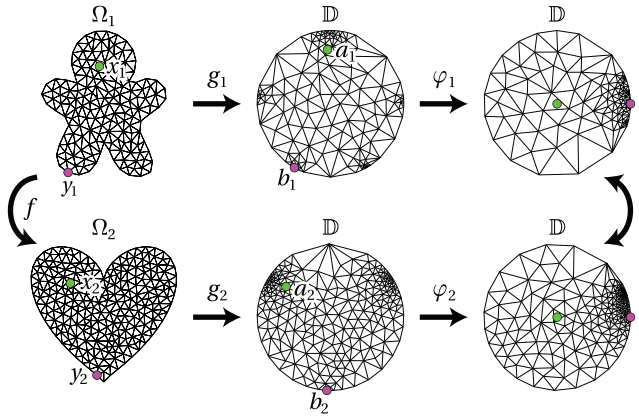

Figure 2: *Given the precomputed conformal maps g_1 and g_2 of Ω_1 and Ω_2 to the disk $\mathbb{D}$, the specific conformal map from Ω_1 to Ω_2 that maps x_1 and y_1 to x_2 and y_2 can be defined, after computing the images of the constraints under g_1 and g_2 and determining the Möbius transformations that further map them to 0 and 1, as $f = g_2^{-1} \circ \varphi_2^{-1} \circ \varphi_1 \circ g_1$.*

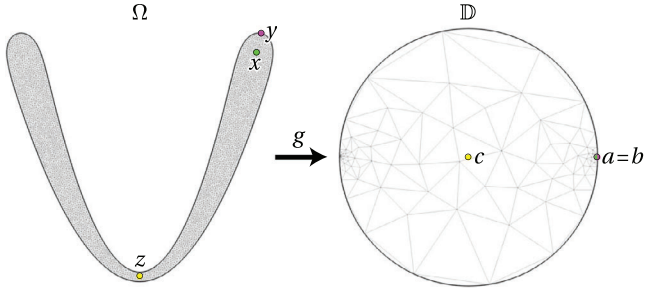

Figure 3: *The precomputed conformal map g from a V-shaped domain Ω to the unit disk $\mathbb{D}$ can exhibit crowding so severe that the image $a = g(x)$ of the interior constraint x lies on the boundary of $\mathbb{D}$ in double precision arithmetic, making it impossible to apply a Möbius transformation.*

and r is the constant $1 - e^{-12.5\pi}$. Suppose that we have somehow been able to precompute some conformal g, which happens to map $z = -\mathrm{i}$ to $c = g(z) = 0$ and $y = 1 + \mathrm{i}$ to $b = g(y) = 1$, numerically. If the user now chooses, for example, $x = 0.96 + 0.8432\mathrm{i}$ as the interior and y as the boundary constraint, then the method described above will fail. This is because the image $a = g(x)$ of x is so close to the boundary of $\mathbb{D}$ that it equals $b = 1$ in double precision and the subsequent Möbius transformation φ that is supposed to map a to 0 is not well-defined. We will revisit this example in Section 5.2. Note that similar problems can occur for any other precomputed conformal map g and that even CRDT [DV98], a Schwarz–Christoffel method specifically developed to deal with extreme crowding, cannot prevent it. We conclude that using $\mathbb{D}$ as an intermediate domain, coupled with the Möbius transformation, leads to significant numerical risks and it is better to map directly between domains. This is our approach.

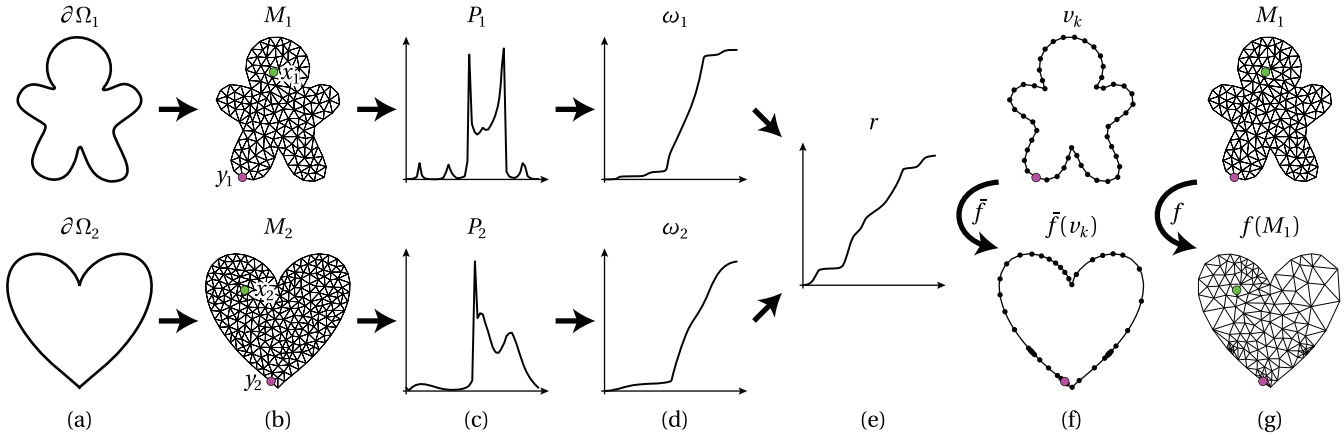

Figure 4: *Overview of our interactive tool.* ***Preprocessing:*** *(a) user loads two shape boundaries $\partial\Omega_i$ ($i = 1, 2$); (b) uniformly sample $\partial\Omega_i$, create Delaunay meshes M_i, and compute and store harmonic coordinates Φ_i of all mesh vertices.* ***Online:*** *(c) for a pair of interactively chosen interior points x_i (green) and boundary points y_i (magenta), use Φ_i to approximate the Poisson kernel P_i of Ω_i at x_i; (d) starting at y_i, integrate P_i to obtain the harmonic measure ω_i of x_i; (e) match harmonic measures by computing the reparameterisation $r = \omega_2^{-1} \circ \omega_1$; (f) use r to construct $\bar{f}$, which conformally maps the boundary vertices v_k of ∂M_1 to ∂M_2; (g) use Φ_1 to extend $\bar{f}$ to a conformal map f of all the interior vertices of M_1.*

1.3. Contribution and overview

Our conformal mapping algorithm belongs to the BEM family and overcomes the crowding problem to the extent possible by avoiding using any intermediate domain. Instead, we map the boundary of the source domain Ω_1 *directly* to the boundary of the target domain Ω_2. This mapping is then extended from the boundary to the interior.

Our method uses the concept of *harmonic measure*, exploiting the fact that it is a *conformal invariant*. The key mechanism is an efficient algorithm to 'match' the harmonic measure along the source and target boundaries, resulting in a conformal map $\bar{f} : \partial\Omega_1 \to \partial\Omega_2$ which can be extended to a conformal $f : \Omega_1 \to \Omega_2$. The result is fast enough to 'bring to life' the Riemann mapping theorem in an interactive system.

After reviewing the definition of harmonic measure and its relation to the Poisson kernel (Section 2), we reformulate both in the parametric setting and explain how they can be used to find the unique conformal map $f : \Omega_1 \to \Omega_2$ that satisfies a user-specified correspondence between two interior and two boundary points (Section 2.1). To compute f numerically, we suggest approximating the parametric Poisson kernel at any interior point with a piecewise linear function and its parametric harmonic measure with a piecewise quadratic function (Section 3.1). We then approximate the boundary map $\bar{f}$ by matching the harmonic measures of the corresponding interior points and extend it to an approximation of f (Section 3.2). Overall, this strategy (see Figure 4) leads to the first real-time tool (Section 4) for interactively exploring the space of all conformal maps between two given domains (see Figure 1, left). A straightforward extension of our tool (Section 4.3) can be used to compute all conformal parameterizations of a 3D mesh with boundary over an arbitrary domain (see Figure 1, right) in real time. Results and comparisons to the state of the art are provided in Section 5.

2. Background

Given a simply connected compact domain $\Omega \subset \mathbb{R}^2$ with smooth boundary, the *Poisson kernel* of Ω is the unique positive bivariate function $P_\Omega : \operatorname{int}\Omega \times \partial\Omega \to \mathbb{R}$, such that

$$h(x) = \int_{\partial\Omega} P_\Omega(x, y)\, h(y)\, \mathrm{d}y \tag{2}$$

for any *harmonic* function $h : \Omega \to \mathbb{R}$ (i.e., $\Delta h(x) = 0$) and any interior point $x \in \operatorname{int}\Omega$. Since constant and linear functions are harmonic, it follows that

$$\int_{\partial\Omega} P_\Omega(x, y)\, \mathrm{d}y = 1, \qquad \int_{\partial\Omega} P_\Omega(x, y)\, y\, \mathrm{d}y = x, \tag{3}$$

hence the Poisson kernel has unit measure and reproduces linear functions.

For any boundary segment $Y \subseteq \partial\Omega$, the *harmonic measure* of Y at $x \in \operatorname{int}\Omega$ with respect to Ω is defined as [Kra16]

$$\omega_\Omega(x, Y) = h_Y(x),$$

where h_Y is the harmonic function with boundary values $h(y) = 1$ for $y \in Y$ and $h(y) = 0$ for $y \in \partial\Omega \setminus Y$. It then follows from Equation (2) that the harmonic measure can be expressed as the integral of the Poisson kernel over the segment,

$$\omega_\Omega(x, Y) = \int_Y P_\Omega(x, y)\, \mathrm{d}y.$$

Alternatively, and equivalently, harmonic measure can be defined as the probability that a random walk starting at x exits the interior of Ω at some point $y \in Y$ [Kak44].

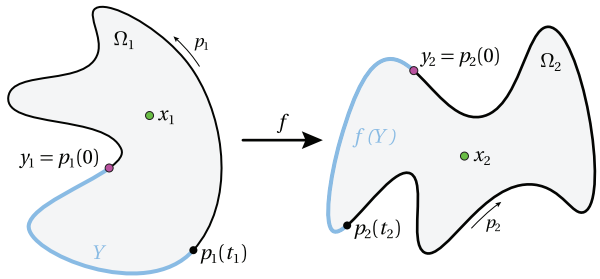

Figure 5: *If the conformal map $f : \Omega_1 \to \Omega_2$ with $f(x_1) = x_2$ and $f(y_1) = y_2$ maps the boundary segment $p_1([0, t_1]) \subset \partial\Omega_1$ to the boundary segment $p_2([0, t_2]) \subset \partial\Omega_2$, then $\omega_{x_1}(t_1) = \omega_{x_2}(t_2)$ and vice versa.*

An important property of harmonic measure is that it is a *conformal invariant* [Kra16]. If $f : \Omega_1 \to \Omega_2$ is a conformal map, then

$$\omega_{\Omega_1}(x, Y) = \omega_{\Omega_2}(f(x), f(Y))$$

for any $x \in \operatorname{int}\Omega_1$ and any $Y \subseteq \partial\Omega_1$.

2.1. Parametric setting

Fixing some boundary point $y \in \partial\Omega$ and denoting by ℓ the length of $\partial\Omega$, let $p : [0, \ell] \to \partial\Omega$ be the (counter-clockwise) *arc length parameterization* (i.e., $\|p'(t)\| = 1$) of the domain boundary with $p(0) = p(\ell) = y$. We then define the *parametric* Poisson kernel at $x \in \operatorname{int}\Omega$ as the positive function $P_x : [0, \ell] \to \mathbb{R}$ with

$$P_x(t) = P_\Omega(x, p(t))$$

and the *parametric* harmonic measure relative to x as the non-negative and monotonically increasing function $\omega_x : [0, \ell] \to \mathbb{R}$ with

$$\omega_x(t) = \omega_\Omega(x, p([0, t])) = \int_0^t P_x(s)\,\mathrm{d}s. \tag{4}$$

In this parametric setting, the conformal invariance of harmonic measure implies that if $f : \Omega_1 \to \Omega_2$ is the conformal map which maps the fixed boundary point $y_1 \in \partial\Omega_1$ to $y_2 = f(y_1) \in \partial\Omega_2$ and some $x_1 \in \operatorname{int}\Omega_1$ to $x_2 = f(x_1) \in \operatorname{int}\Omega_2$, then

$$\omega_{x_1}(t_1) = \omega_{x_2}(t_2) \tag{5}$$

for any $t_1 \in [0, \ell_1]$, where t_2 is the arc length parameter of the image of $p_1(t_1)$ under f, that is, $f(p_1(t_1)) = p_2(t_2)$; see Figure 5. In other words, if $r : [0, \ell_1] \to [0, \ell_2]$ denotes the *reparameterisation* of p_2 induced by the image of p_1 under f, that is, $f \circ p_1 = p_2 \circ r$, then $\omega_{x_1} = \omega_{x_2} \circ r$.

Conversely, given the harmonic measure functions ω_{x_1} and ω_{x_2} for some $x_1 \in \operatorname{int}\Omega_1$ and $x_2 \in \operatorname{int}\Omega_2$, we can find the conformal map f with $f(x_1) = x_2$ and $f(y_1) = y_2$ in two steps. We first use the harmonic measures to determine the boundary map $f \circ p_1 = p_2 \circ \omega_{x_2}^{-1} \circ \omega_{x_1}$. We then take advantage of the fact that both components of a conformal map are harmonic. Therefore, f can be expressed, using the parametric analogue of Equation (2), as

$$f(x) = \int_0^{\ell_1} P_x(t) f(p_1(t))\,\mathrm{d}t. \tag{6}$$

Alternatively, we can use the parametric version of Cauchy's integral formula Equation (1) to write f in complex number formulation as

$$f(z) = \frac{1}{2\pi\mathrm{i}} \int_0^{\ell_1} \frac{f(p_1(t))}{p_1(t) - z}\, p_1'(t)\,\mathrm{d}t. \tag{7}$$

3. Theory

The first step towards handling conformal maps numerically consists of sampling the boundary of a given smooth domain Ω at m parameters $0 \le s_1 < \cdots < s_m < \ell$ and approximating Ω by the polygon $\tilde{\Omega} = [v_1, \ldots, v_m]$ with (boundary) vertices $v_k = p(s_k) \in \partial\Omega$. In a second step, we discretize $\tilde{\Omega}$ by generating n interior vertices $v_{m+1}, \ldots, v_{m+n} \in \operatorname{int}\tilde{\Omega}$ and computing the (boundary-conforming) Delaunay triangulation of all vertices, resulting in a planar triangle mesh M with $m + n$ vertices and boundary $\partial\tilde{\Omega}$.

3.1. Approximating Poisson kernel and harmonic measure

In order to approximate the Poisson kernel and the harmonic measure of Ω, we first compute the *m harmonic coordinates* [JMD*07] $\phi_i = (\phi_{i1}, \ldots, \phi_{im})$ of each vertex v_i of M (see Appendix A). Harmonic coordinates approximate the Poisson kernel P_Ω in the sense that they are positive and provide a discrete analogue of the properties in Equation (3),

$$\sum_{k=1}^{m} \phi_{ik} = 1, \qquad \sum_{k=1}^{m} \phi_{ik} v_k = v_i$$

for $i = 1, \ldots, m + n$, and in Equation (2),

$$h(v_i) \approx \sum_{k=1}^{m} \phi_{ik} h(v_k), \tag{8}$$

but instead of this discrete, pointwise approximation, we need a continuous approximation of the parametric Poisson kernel.

To this end, let t_k be the arc length parameter of the boundary vertex v_k, $k = 1, \ldots, m$, and let $\tilde{\ell}$ be the length of $\partial\tilde{\Omega}$. For every mesh vertex v_i, we then derive from its harmonic coordinates ϕ_i the values $\psi_i = (\psi_{i1}, \ldots, \psi_{im})$ of a piecewise linear function $\tilde{P}_{v_i} : [0, \tilde{\ell}] \to \mathbb{R}$ with $\tilde{P}_{v_i}(t_k) = \psi_{ik}$, which approximates $\tilde{P}_{v_i}$ (see Appendix B). We apply linear interpolation to extend this construction to other domain points. If x is a point inside the triangle $[v_i, v_j, v_k]$ of M with barycentric coordinates (ϱ, σ, τ), that is, $x = \varrho v_i + \sigma v_j + \tau v_k$, then the harmonic coordinates of x are

$$\phi_x = \varrho\phi_i + \sigma\phi_j + \tau\phi_k \tag{9}$$

and the parametric Poisson kernel at x is approximated by

$$\tilde{P}_x = \varrho\tilde{P}_{v_i} + \sigma\tilde{P}_{v_j} + \tau\tilde{P}_{v_k},$$

the piecewise linear function over $[0, \tilde{\ell}]$ with $\tilde{P}_x(t_k) = \psi_{x,k}$, where

$$\psi_x = \varrho\psi_i + \sigma\psi_j + \tau\psi_k. \tag{10}$$

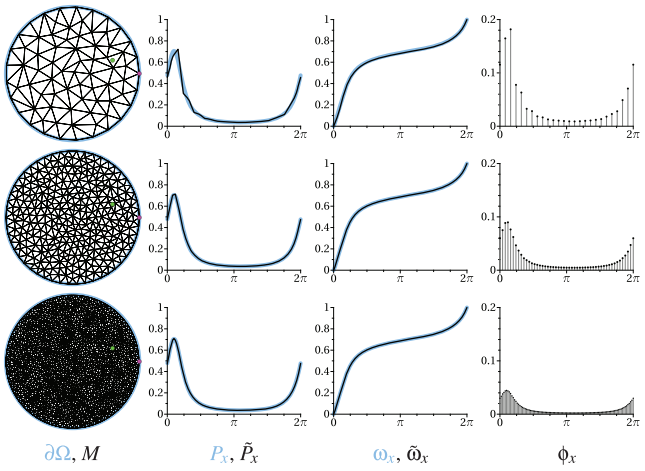

Figure 6: *Exact (blue) and approximate (black) Poisson kernel (P_x, $\tilde{P}_x$) and harmonic measure (ω_x, $\tilde{\omega}_x$) for $x = (0.6, 0.2)$ (green) inside the unit disk $\mathbb{D}$, derived from the harmonic coordinates ϕ_x of x, using a mesh M with $m = 25$ (top), $m = 50$ (middle), and $m = 100$ (bottom) boundary and $n = 50, 273, 1152$ interior vertices, respectively. The first boundary vertex is (1,0) (magenta).*

By construction, $\tilde{P}_x$ is positive for any $x \in \operatorname{int} \tilde{\Omega}$ and satisfies

$$\int_0^{\tilde{\ell}} \tilde{P}_x(t)\,\mathrm{d}t = 1, \qquad \int_0^{\tilde{\ell}} \tilde{P}_x(t)\,\tilde{p}(t)\,\mathrm{d}t \approx x,$$

the parametric analogue of Equation (3), at least in an approximate sense.

Following Equation (4), we finally approximate the harmonic measure ω_x by the piecewise quadratic function $\tilde{\omega}_x : [0, \tilde{\ell}] \to \mathbb{R}$ with

$$\tilde{\omega}_x(t) = \int_0^t \tilde{P}_x(s)\,\mathrm{d}s, \tag{11}$$

which is non-negative and monotonically increasing, because $\tilde{P}_x$ is positive. Expressing $t \in [0, \tilde{\ell}]$ as $t = (1-\lambda)t_k + \lambda t_{k+1}$ for some $\lambda \in [0, 1]$ and some $k \in \{1, \dots, m\}$, we find that

$$\tilde{\omega}_x(t) = \sum_{j=1}^{k-1} e_j \frac{\psi_{x,j} + \psi_{x,j+1}}{2} + e_k \frac{\lambda(2-\lambda)\psi_{x,k} + \lambda^2 \psi_{x,k+1}}{2}. \tag{12}$$

Figure 6 shows an example where Ω is the unit disk $\mathbb{D}$. Using the arc length parameterization of the unit circle, starting at $y = (1, 0)$,

$$p : [0, 2\pi] \to \partial\Omega, \quad p(t) = (\cos t, \sin t),$$

the exact parametric Poisson kernel and the exact parametric harmonic measure for $x \in \operatorname{int} \Omega$ are known to be [Kra16]

$$P_x(t) = \frac{1}{2\pi} \cdot \frac{1 - \|x\|^2}{\|p(t) - x\|^2}, \qquad \omega_x(t) = \frac{2\theta(t) - t}{2\pi},$$

where $\theta(t) \in [0, 2\pi]$ is the angle between $p(0) - x$ and $p(t) - x$. While the approximating Poisson kernel $\tilde{P}_x$ tends to 'zig-zag' around the exact Poisson kernel P_x, especially for low-resolution meshes, its integral $\tilde{\omega}_x$ provides a very good approximation of the exact harmonic measure ω_x. Note that the graphs of $\tilde{P}_x$ and $\tilde{\omega}_x$ (and also ϕ_x) are stretched slightly by the factor $2\pi/\tilde{\ell}$ in the horizontal direction, to allow for a better comparison to the graphs of P_x and ω_x in Figure 6.

3.2. Approximating conformal maps

Given two domains Ω_1 and Ω_2, we can now approximate the conformal map $f : \Omega_1 \to \Omega_2$ with the boundary point constraint $f(y_1) = y_2$ and the interior point constraint $f(x_1) = x_2$ as follows.

We first discretize Ω_1 by a mesh M_1 with $m + n$ vertices $v_i := v_{1,i}$ and Ω_2 by a mesh M_2 with $m' + n'$ vertices $v'_i := v_{2,i}$ and assume without loss of generality that the boundary point constraint requires f to map the first boundary vertex of M_1 to the first boundary vertex of M_2, that is, $y_1 = v_1$ and $y_2 = v'_1$, and that x_1 and x_2 are points inside some triangles of M_1 and M_2, so that we can compute the values $\psi := \psi_{1,x_1}$ and $\psi' := \psi_{2,x_2}$ of the piecewise linear approximations of the Poisson kernels at x_1 and x_2 as in Equation (10).

Next, we generate the approximate images of the other boundary vertices of M_1 under f by exploiting the fact that harmonic measure is conformally invariant. Using the approximate parametric harmonic measures $\tilde{\omega}_1 := \tilde{\omega}_{1,x_1}$ and $\tilde{\omega}_2 := \tilde{\omega}_{2,x_2}$ relative to x_1 in Ω_1 and to x_2 in Ω_2, we approximate the reparameterisation $r = \omega_{x_2}^{-1} \circ \omega_{x_1}$ by the piecewise linear function $\tilde{r} : [0, \tilde{\ell}_1] \to [0, \tilde{\ell}_2]$ with $\tilde{r}(t_k) = r_k$ for $k = 1, \dots, m+1$, where r_k is the unique parameter that satisfies

$$\tilde{\omega}_1(t_k) = \tilde{\omega}_2(r_k).$$

Clearly, $r_1 = 0$ and $r_{m+1} = \tilde{\ell}_2$. To find the remaining parameters $r_2, \dots, r_m$ that match the parameters $t_2, \dots, t_m$, we use Equation (12) to compute $A_k = \tilde{\omega}_1(t_k)$ as

$$A_1 := 0, \qquad A_{k+1} = A_k + e_k \frac{\psi_k + \psi_{k+1}}{2}, \quad k = 1, \dots, m,$$

and likewise, $B_l = \tilde{\omega}_2(t'_l)$ as

$$B_1 := 0, \qquad B_{l+1} = B_l + e'_l \frac{\psi'_l + \psi'_{l+1}}{2}, \quad l = 1, \dots, m'.$$

We further conclude from Equation (12) that $r_k = (1-\lambda)t'_l + \lambda t'_{l+1}$, where $l \in \{1, \dots, m'\}$ is the unique index such that

$$B_l \le A_k < B_{l+1}$$

and

$$\lambda = \begin{cases} \mu, & \Delta = 0, \\ \operatorname{sign}(\Delta)\sqrt{\xi^2 + 2\mu\xi + \mu} - \xi, & \Delta \neq 0, \end{cases}$$

where

$$\Delta = \psi'_{l+1} - \psi'_l, \qquad \mu = \frac{A_k - B_l}{B_{l+1} - B_l}, \qquad \xi = \frac{\psi'_l}{\Delta}.$$

This *matching algorithm* is the core component of our interactive tool (see Section 4). Using $\tilde{r}$ and the arc length parameterization $\tilde{p}_2$ of $\partial\tilde{\Omega}_2$, we define

$$w_k = \tilde{p}_2(\tilde{r}(t_k)) = \tilde{p}_2(r_k) = (1-\lambda)v'_l + \lambda v'_{l+1}, \tag{13}$$

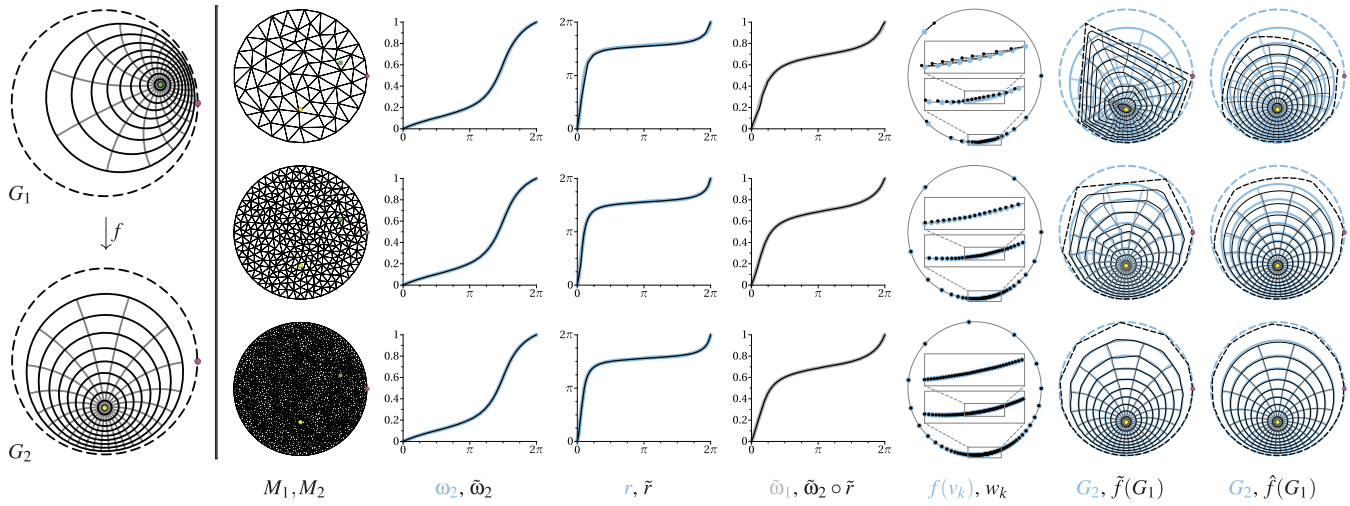

Figure 7: *Approximations $\tilde{f}$ and $\hat{f}$ (right) of the conformal Möbius map $f : \mathbb{D} \to \mathbb{D}$ (left), which maps $x_1 = (0.6, 0.2)$ (green) to $x_2 = (0, -0.5)$ (yellow) and $y_1 = (1, 0)$ to $y_2 = (1, 0)$ (magenta), using meshes $M_1 = M_2$ with $m = 25$ (top), $m = 50$ (middle), and $m = 100$ (bottom) boundary vertices (cf. Figure 6). Both approximations are based on the approximations (black) of the exact (blue) boundary vertex images (w_k, $f(v_k)$), using harmonic and Cauchy–Green coordinates, respectively. The points w_k are computed with the piecewise linear approximation (black) of the exact (blue) reparameterisation ($\tilde{r}$, r), which in turn is determined by matching the approximate harmonic measure (grey) at x_1 with the reparameterised approximate harmonic measure (black) at x_2 ($\tilde{\omega}_1$, $\tilde{\omega}_2 \circ \tilde{r}$). Note that $\tilde{f}$ reproduces the boundary mapping, whereas $\hat{f}$ does not necessarily do so.*

which approximates $f(v_k) = f(\tilde{p}_1(t_k))$ for $k = 1, \dots, m$, because $f \circ \tilde{p}_1 \approx \tilde{p}_2 \circ \tilde{r}$.

Finally, we recall that f on the interior can be expressed using the Poisson kernel Equation (6) and thus can be approximated using harmonic coordinates Equation (8). Consequently, we define the approximation $\tilde{f} : \tilde{\Omega}_1 \to \mathbb{R}^2$ of f as

$$\tilde{f}(x) = \sum_{k=1}^{m} \phi_{x,k} w_k, \tag{14}$$

where ϕ_x are the harmonic coordinates of x in Equation (9).

Alternatively, we may use Cauchy's integral formula Equation (7) and approximate f with the complex function $\hat{f} : \tilde{\Omega}_1 \to \mathbb{C}$,

$$\hat{f}(z) = \sum_{k=1}^{m} \gamma_k(z) w_k, \tag{15}$$

where

$$\gamma_k(z) = \frac{1}{2\pi \mathrm{i}} \left(\frac{v_{k+1} - z}{v_{k+1} - v_k} \log \frac{v_{k+1} - z}{v_k - z} - \frac{v_{k-1} - z}{v_k - v_{k-1}} \log \frac{v_k - z}{v_{k-1} - z} \right)$$

for $k = 1, \dots, m$ are the Cauchy–Green coordinates of z with respect to $\tilde{\Omega}_1$ [WBCG09].

Figure 7 shows an example where we use our approach to approximate the conformal map f from the unit disk $\mathbb{D}$ to itself that satisfies the boundary constraint $f(1, 0) = (1, 0)$ and the interior constraint $f(0.6, 0.2) = (0, -0.5)$. This is a Möbius transformation and can be written using complex numbers as

$$f(z) = \frac{1 - 5\mathrm{i} + 6\mathrm{i}z}{6 + (-5 + \mathrm{i})z}.$$

To visualize f and its approximations, we use the uniform polar grid G with grid points $g_{jk} = \frac{j}{10} \exp\left(\frac{k}{24} 2\pi \mathrm{i}\right) \in \mathbb{C}$ for $j = 0, \dots, 9$ and $k = 0, \dots, 23$ and consider the Möbius transformations

$$g_1(z) = \frac{3 + \mathrm{i} + (3 - 4\mathrm{i})z}{5 + (1 - 3\mathrm{i})z}, \qquad g_2(z) = \frac{-5\mathrm{i} + (6 + 8\mathrm{i})z}{10 - (4 - 3\mathrm{i})z},$$

which both map (1,0) to $y_1 = y_2 = (1, 0)$ and the origin (0,0) to $x_1 = (0.6, 0.2)$ and $x_2 = (0, -0.5)$, respectively. Applying g_1 and g_2 to G gives the conformal grids $G_1 = g_1(G)$ and $G_2 = g_2(G)$, which are centred at x_1 and x_2, respectively. Since $f = g_2 \circ g_1^{-1}$, it follows that $f(G_1) = G_2$ (see Figure 7, left), and we can appreciate the approximation quality of $\tilde{f}$ and $\hat{f}$ by comparing $\tilde{f}(G_1)$ and $\hat{f}(G_1)$ with G_2 (see Figure 7, right). Note that we did not include the unit circle in the polar grid G, because $\tilde{f}$ and $\hat{f}$ are only defined over $\tilde{\Omega}_1$, which is inscribed in the unit circle. Instead, we show $\tilde{f}(\partial\tilde{\Omega}_1)$ and $\hat{f}(\partial\tilde{\Omega}_1)$, which approximate $f(\partial\Omega_1)$, that is, the unit circle, as dashed curves. Figure 7 confirms that conformal maps are very sensitive to noise: although $\tilde{\omega}_1$ (cf. Figure 6) and $\tilde{\omega}_2$ approximate the exact harmonic measures ω_1 and ω_2 very well, even for low-resolution meshes, and thus lead to a very good approximation $\tilde{r}$ of the exact reparameterisation r, the error is amplified significantly when we use $\tilde{r}$ to approximate the boundary vertex images $f(v_k)$ with w_k, because the derivative of r is large near the origin. However, since all errors seem to be on the order of $O(1/\sqrt{m})$, according to our experiments (see Figure 15), they decrease rather quickly as the number of boundary vertices increases.

Table 1: *Preprocessing times (in seconds) of our method for triangulating Ω and determining the values Ψ of the piecewise linear approximations of the Poisson kernels at all mesh vertices and of the other methods for computing a conformal map from Ω to $\mathbb{D}$.*

				our method				
	m	n	N	triangulation	compute Ψ	BFF	zipper	SC Toolbox
gingerbread man	500	13620	27738	0.005	0.119	0.165	0.342	∼180
heart	1000	90181	181360	0.048	1.633	0.835	1.047	∼1800
blob	1000	97649	196296	0.043	1.731	0.888	1.064	∼2400
cinquefoil	1000	101985	204968	0.044	1.843	0.946	1.063	∼1700
V-shape	2000	62668	127334	0.028	2.119	0.598	4.242	∼9600
maple leaf	3200	344911	693020	0.145	33.22	3.300	10.04	—

4. Practice

To turn the considerations from Section 3 into an interactive tool, which allows the user to explore the space of conformal maps from a *source* domain Ω_1 to a *target* domain Ω_2, we exploit the fact that all compute-intensive tasks can be dealt with in a preprocessing step, while the computational effort during the exploration phase is minimal. We implemented this tool in C++ on a MacBook Air with an Apple M3 (8 cores) processor and 24 GB RAM. The pseudocode can be found in Appendix C.

4.1. Preprocessing

In our tool, we assume the boundary of a domain Ω to be given in terms of the vertices of a (possibly dense) polygon. After loading the data, we uniformly resample this polygon with the desired number (m) of boundary vertices, providing the user with the option to include the given vertices as samples, so that actual corners of the domain are reproduced. Moreover, the user loads a texture image I associated with the source mesh, whose image under the conformal map will be shown on the target side.

We use *Triangle* [She96] to create a conforming Delaunay triangulation of the resampled boundary with no additional boundary points, no angles smaller than 20°, and a user-specific maximal triangle area $A_{\max}$, which is set to $h^2/2$ by default, where h is the average boundary edge length. The resulting mesh M has $n \in O(m^2)$ interior vertices and $N = m + 2n - 2 \in O(m^2)$ triangles, and it is typically generated in $O(n \log n)$ time.

We then recall that the harmonic coordinates of the boundary vertices $\bar{v} = (v_1, \ldots, v_m)$ are given by $\bar{\Phi} = I_m$ and determine the harmonic coordinates $\hat{\Phi}$ of the interior mesh vertices $\hat{v} = (v_{m+1}, \ldots, v_{m+n})$ by solving Equation (A.1). To this end, we use CHOLMOD [CDHR08] to compute and store the sparse Cholesky factorization of the sparse, symmetric, positive-definite matrix $\hat{L}$ in approximately $O(n^{1.5})$ time, followed by back-substitution for each of the m columns of $\hat{\Phi}$. Since CHOLMOD generates Cholesky factors with $O(n \log n)$ non-zero elements, the latter has a time complexity of $O(m\, n \log n)$. Moreover, we compute and store Ψ, the values of the approximate Poisson kernels, by appropriately scaling the columns of Φ, which can then be deleted, and (optionally) compute and store the Cauchy–Green coordinates for all v_i in a matrix $\Gamma \in \mathbb{C}^{(m+n)\times m}$ with $\Gamma_{ik} = \gamma_k(v_i)$, which can be done in $O(mn) = O(n^{1.5})$ time.

These preprocessing tasks have to be carried out for both the source and the target domain and must be repeated for either, if the user changes m or $A_{\max}$ for that domain. For concrete timings, we refer to Section 5.2 and Table 1.

4.2. Online

During the exploration phase, the user can move the interior constraint point x to any position inside the polygon defined by the boundary vertices of M, while keeping track of the triangle T of M that contains x as well as the barycentric coordinates of x with respect to T. Likewise, the boundary constraint point y can be moved along the boundary to any boundary vertex of M.

Whenever there is a change of either a source or target constraint point, we update the conformal map from source to target by first computing ψ and ψ' as in Equation (10) and then using the matching algorithm from Section 3.2 to find the images of the source boundary vertices $\bar{w} = (w_1, \ldots, w_m)$ as in Equation (13). Note that the matching algorithm runs in $O(m)$ time with just one pass through the sequences $(A_1, \ldots, A_m)$ and $(B_1, \ldots, B_{m'})$, because they are both strictly increasing. We then find the images $w = (\bar{w}, \hat{w})$ of all vertices either by keeping $\bar{w}$ and determining $\hat{w} = \tilde{f}(\hat{v}) = \hat{\Phi}\bar{w}$, using the precomputed Cholesky factors of $\hat{L}$ to solve $\hat{L}\hat{w} = -\bar{L}\bar{w}$ in $O(n \log n)$ time or by letting $w = \hat{f}(v) = \Gamma\bar{w}$ (which overwrites $\bar{w}$), following Equation (15) and using the precomputed Cauchy–Green coordinates in $O(mn)$ time. To visualize the image of I under $\tilde{f}$ or $\hat{f}$, we finally render the source mesh M_1 with the mapped vertices w and texture coordinates v on the target side.

Besides this *forward* mapping, our tool can also use *backward* mapping to show the conformal image of I. In this setting, we reverse the roles of M_1 and M_2, compute the approximation of the constrained conformal map from Ω_2 to Ω_1, and render M_2 with its vertices v' using the mapped vertices w' as texture coordinates.

Our tool runs with at least 25 frames per second (fps) on our platform, if both meshes have up to $N = 400\,\mathrm{K}$ triangles. Again, we refer to Section 5.2 and Table 2 for concrete timings.

4.3. Conformal parameterizations

The concept of harmonic measure and the fact that it is a conformal invariant both carry over to manifolds with smooth boundaries in $\mathbb{R}^n$, in particular to genus-0 manifold surfaces with a boundary

Table 2: *Processing times (in milliseconds) of our method for updating the conformally mapped target mesh whenever the user changes a source or target constraint.*

	update ψ or ψ' + matching	compute $\tilde{f}(v)$
Figure 11	0.059	0.833
Figure 12	0.344	3.462
Figure 13	0.383	22.82

in $\mathbb{R}^3$. As the *uniformization theorem* [Abi81] guarantees the existence of a conformal map from such a surface to the unit disk and it is possible to map the unit disk conformally to any other simply connected planar domain, it follows that our framework can also be used to approximate all conformal parameterizations of a 3D triangle mesh with boundary over a given simply connected planar domain.

To compute and visualize these parameterizations with our tool, we let the parameter domain be M_1, load the triangle mesh as M_2, and use backward mapping with harmonic coordinates to map the texture associated with M_1 onto the 3D mesh. Note that harmonic coordinates depend only on angles and areas, hence carry over to the 3D setting, but that it is not possible to use Cauchy–Green coordinates in this case.

5. Examples

5.1. Conformal maps

Figure 8 shows the four different options that our tool offers for approximating a constrained conformal map. Forward mapping the source mesh to the target domain can exhibit 'undercuts' in regions near the target boundary where the source boundary vertices are spread far apart by the conformal boundary map. In contrast, the target domain is always perfectly covered by definition if backward mapping is used. Of course, backward mapping does not avoid undercuts, but since they affect the texture instead of the geometry, they are less evident. Both mappings can be carried out in two different ways. While mapping with harmonic coordinates interpolates the boundary by construction, mapping with Cauchy–Green coordinates does not and may cause the mapped boundary to 'shrink' or 'bulge'. The advantage of Cauchy–Green coordinates is that they provide a smooth conformal map, while maps based on harmonic coordinates are piecewise linear by construction, but this advantage does not manifest itself in our setting, since we map only the vertices of the mesh and then let the graphics hardware interpolate the texture linearly over each triangle. Overall, based on our experiments, we found that backward mapping with harmonic coordinates gives the best visual results and recommend using this option. However, Figure 8 also shows that differences between the mapping methods are noticeable only at low-resolution, but not at high resolution, since they all seem to converge to the exact conformal map in the limit.

This convergence behaviour is illustrated in Figure 9 for the case of the approximate conformal map with harmonic coordinates $\tilde{f}$ from the *maple leaf* (M_1) to the *cinquefoil* domain (M_2). For each triangle T of M_1 we color-coded T and its image $\tilde{f}(T)$ by the quasi-conformal error $Q = \sigma_1/\sigma_2$, where σ_1, σ_2 are the singular values of the linear map from T to $\tilde{f}(T)$. As the resolution increases, the regions with big conformal errors shrink and concentrate at the boundary points corresponding to local curvature extrema, and the area-weighted average error Q_{avg} approaches the optimal value of 1. Our experiments suggest that $Q_{\text{avg}} = 1 + O(1/m)$, but it remains future work to prove this. The results of a second numerical convergence test can be found in Section 5.2 and Figure 15.

More examples of our tool's capabilities to create and interactively explore conformal maps between two arbitrary planar domains can be seen in Figure 10 and the supplementary video.

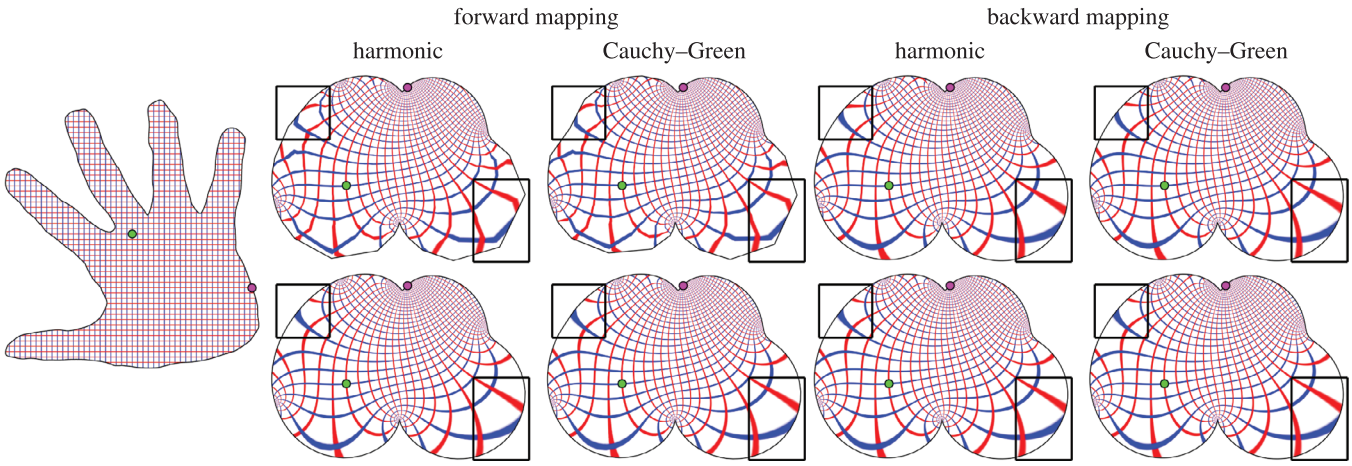

Figure 8: *The four different options (forward or backward mapping with harmonic or Cauchy–Green coordinates) for approximating a constrained conformal map from the* hand *to the* blob *domain with our tool give notably different results (top), if low-resolution meshes (hand: $m_1 = 300$, $N_1 = 4346$; blob: $m_2 = 200$, $N_2 = 7788$) are used, but these differences disappear (bottom) at higher resolution (hand: $m_1 = 1800$, $N_1 = 158512$; blob: $m_2 = 1200$, $N_2 = 282304$).*

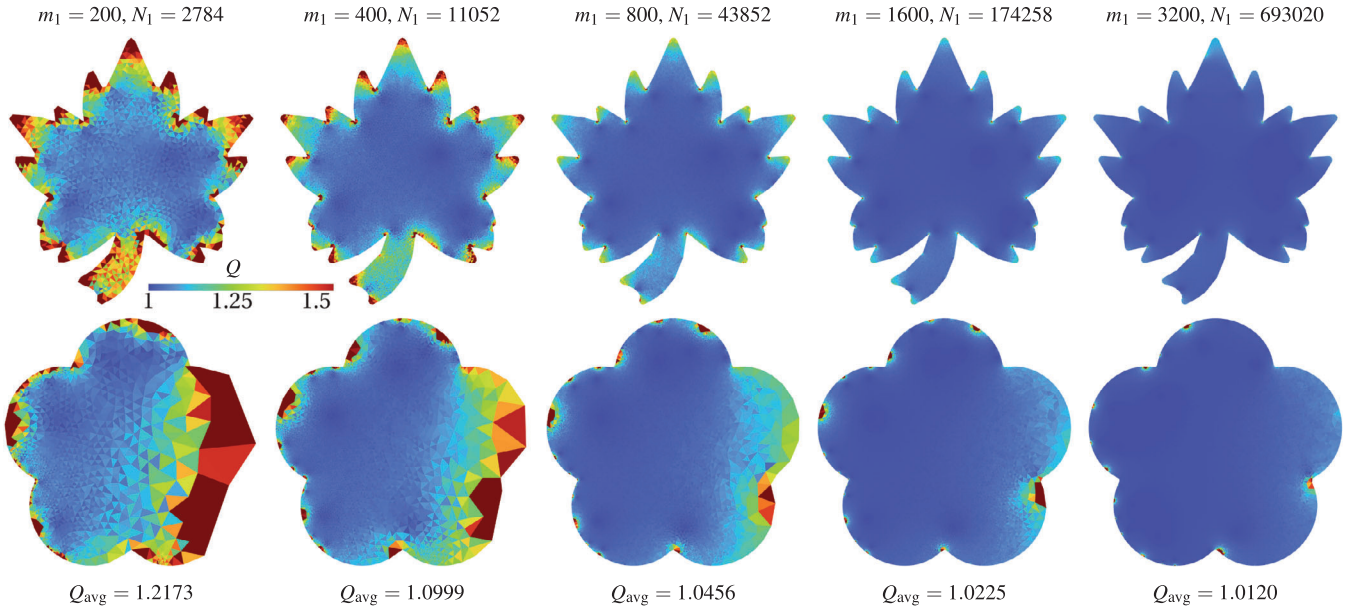

Figure 9: *Our approximate conformal maps (here with harmonic coordinates from the* maple leaf *to the* cinquefoil *domain with the constraints used in the leftmost example in Figure 1) seem to converge to an exact conformal map ($Q_{avg} = 1$) as the resolution of the source mesh increases, while the resolution of the target mesh ($m_2 = 1000$, $N_2 = 204968$) remains fixed.*

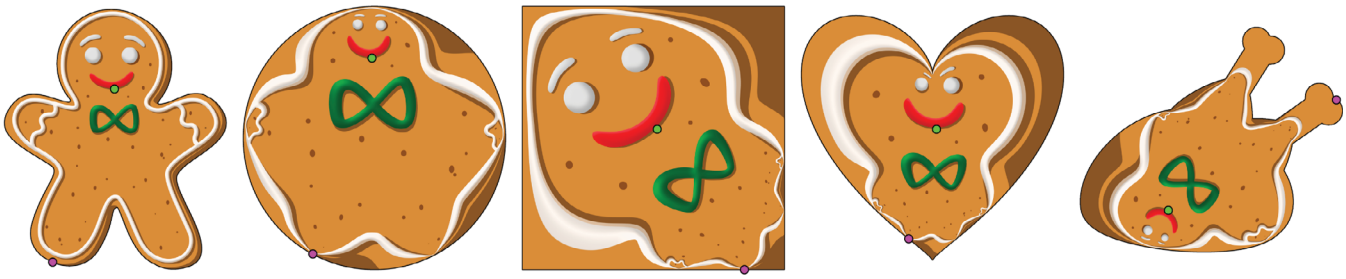

Figure 10: *With our tool, we can conformally reshape a* gingerbread man *into a* disk*, a* square*, a* heart*, and, somewhat non-conformingly, into a* roasted turkey.

5.2. Comparison

To compare our conformal mapping algorithm with the state of the art, which map domains to the unit disk $\mathbb{D}$, we follow the standard approach outlined in Section 1.2.

During the preprocessing stage, we uniformly resample the boundary of a domain Ω (like in our tool) and generate some conformal map g from Ω to $\mathbb{D}$. In the case of *SC Toolbox* [Dri25] and *Zipper* [MR07], we compute the one that maps a manually chosen point (approximating the *Fréchet mean*) to the origin. In the case of BFF [SC18], we use the 'flatten to disk' option (with the default value of 10 iterations), which does not provide any further control over the result, and we normalized the boundary vertices, since they are not guaranteed to be exactly on the unit circle.

During the exploration phase, we first find the images a and b of the interactively chosen constraints x and y under g. While the *SC Toolbox* provides the function `evalinv` for this purpose, we use harmonic coordinates (w.r.t. the same triangulation that we use in our tool) for *Zipper* and BFF. We then apply the Möbius transformation φ that maps a to 0 and b to 1 to all boundary points.

Once this has been done for both the source boundary vertices $v_1, \ldots, v_m$ and the target boundary vertices $v'_1, \ldots, v'_{m'}$, it remains to match the mapped boundary points. To this end, we express the latter in terms of their polar angles α_k and β_l, find, for each $k \in \{1, \ldots, m\}$, the unique index $l \in \{1, \ldots, m'\}$ and the unique value $\lambda \in [0, 1)$, such that $\alpha_k = (1-\lambda)\beta_l + \lambda\beta_{l+1}$, and define $w_k = (1-\lambda)v'_l + \lambda v'_{l+1}$. This procedure is not only very similar to our matching algorithm, but actually mathematically equivalent, because the parametric harmonic measure of the unit disk's centre is just $\omega_0(t) = t/(2\pi)$, that is, the polar angle, divided by the size of its range.

Finally, we extend the conformal boundary map to the interior using harmonic coordinates, just as we do in our tool. Figure 11 shows

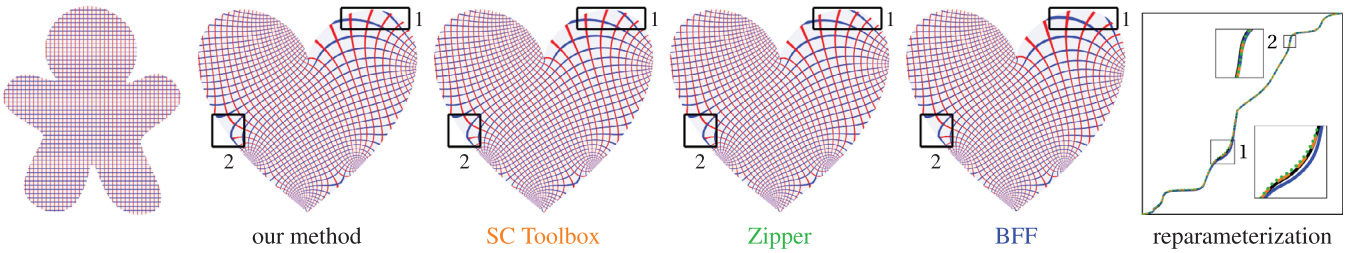

Figure 11: *Approximate conformal maps and corresponding reparameterisations from the* gingerbread man *domain ($m_1 = 500$, $N_1 = 27738$) to the* heart *domain ($m_2 = 1000$, $N_2 = 181360$) with the constraints shown in Figure 4(b).*

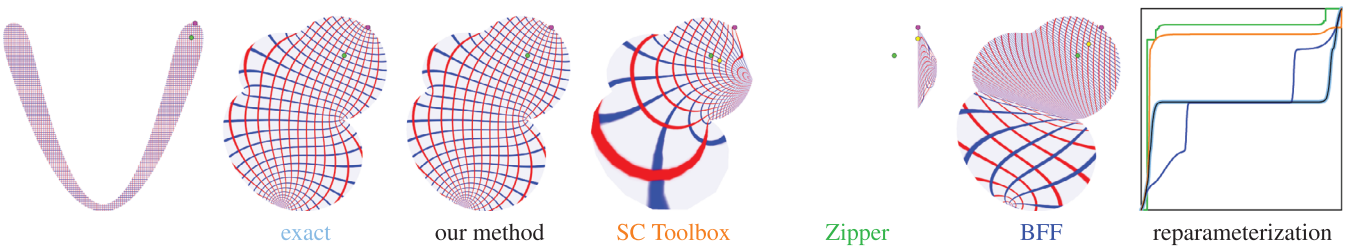

Figure 12: *Approximate conformal maps and corresponding reparameterisations from the* V-shape *domain ($m_1 = 2000$, $N_1 = 127334$) to the* blob *domain ($m_2 = 1000$, $N_2 = 196296$). The source constraints are identical to those in Figure 3.*

the results for an example where the source domain was discretised with a low-resolution mesh. All methods give nearly identical results (see, e.g., region 2), except for some minor visible differences in the marked region 1. The reparameterisation plot, which shows the accumulated arc length at the mapped boundary vertices w_k over the accumulated arc length at the boundary vertices v_k, confirms that the BFF mapping traces out the target boundary slower than the others in this region, while the Zipper mapping is a little bit ahead of our mapping and the one obtained by SC Toolbox. Overall, it is hard to say which mapping is closest to the ground truth, but they all are very reasonable approximations.

The example in Figure 12 goes back to the crowding problem described in Section 1.2 and Figure 3. In addition to knowing the exact conformal map from the *V-shape* domain to $\mathbb{D}$, we know that

$$g_1^{-1}(z) = \frac{4z^2+4z+1}{40} + \frac{50}{12z^2+25z+25} - \frac{3-5\mathrm{i}}{6}\sin\left(\frac{11}{10}z\right)\cos\left(\frac{z}{2}\right) + \log\left(\frac{20z^2-45+36\mathrm{i}}{30}\right)$$

is the exact 'ground truth' conformal map from the unit disk to the *blob* domain. Hence, we can compare the approximate conformal maps from the different approaches with the ground truth. While our result is visually identical to the latter, all other methods suffer from the numerical issues mentioned in Section 1.2. On the one hand, the image a_1 of the interior source constraint x_1 under the precomputed map g_1 is incorrect and too far from $\partial\mathbb{D}$. While this can be seen as an advantage, since it at least leads to a well-defined Möbius transform φ_1, even tiny inaccuracies in a_1 magnify due to the extreme crowding, and the final image $\tilde{f}(x_1)$ (yellow dot) is far from x_2 (green dot). On the other hand, g_1 maps several source boundary points to the same position on $\partial\mathbb{D}$, hence their final images on $\partial\Omega_2$ are identical, too, which is confirmed by the vertical 'jumps' in the reparameterisation plot.

Figure 13 shows the results for two domains that were discretised with high resolution meshes. While the Zipper mapping is basically identical to ours (see Figure 9, right), the BFF mapping is visually different and tends to trace out the boundary slower than ours. It is also the one with the highest area-weighted average error Q_{avg}. We do not show the SC Toolbox mapping, because SC Toolbox did not manage to generate a conformal map for the *Maple leaf* domain with 3200 boundary vertices in reasonable time (less than 12 h).

Table 1 lists the CPU time needed by the different methods for preprocessing the six domains used in the previous three examples. BFF turns out to be the fastest algorithm for computing an approximate conformal map to $\mathbb{D}$. It scales similarly to our method with n, which is not surprising, because it is also based on the classical FEM discretization of the Laplace equation and uses CHOLMOD for solving the corresponding linear system, but our runtime additionally depends (linearly) on m, because we have to solve this system for each of the m harmonic coordinates. Note that the time for computing Ψ for the *maple leaf* mesh is larger than suggested by the asymptotic runtime, because the matrices $\hat{L}$, its Cholesky factors, and Ψ are so big that they (slightly) exceed the available RAM, and so our machine started using virtual (disk) memory. The Zipper algorithm is also very fast, with the advantage of having an $O(m^2)$ runtime that does not depend on n. However, it only generates a conformal boundary map and additional time would have to be spent for preparing the precomputed result, such that the interior user constraint can quickly be mapped to the unit disk during the exploration phase. The SC Toolbox is by far the slowest, which can only be

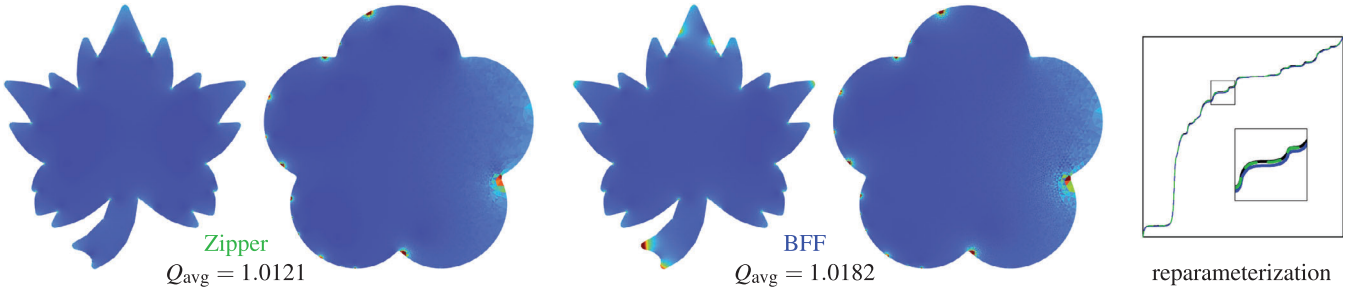

Figure 13: *Approximate conformal maps and corresponding reparameterisations from the* maple leaf *domain ($m_1 = 3200$, $N_1 = 693020$) to the* cinquefoil *domain ($m_2 = 1000$, $N_2 = 204968$) with the constraints used in the leftmost example in Figure 1. Compare to the results obtained by our method in the rightmost column of Figure 9.*

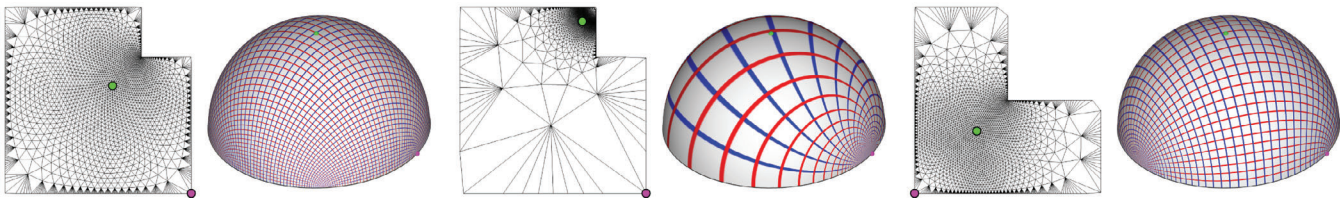

Figure 14: *Constrained conformal maps from the* hemisphere *mesh ($m = 900$, $N = 4800$) to L-shaped domains. The leftmost example reproduces the result that BFF [SC18] delivers when asked to map the hemisphere to a polygonal domain with six right-angled corners (five convex, one concave).*

partially attributed to the fact that the timings are based on MATLAB code, while we used compiled C++ code for the other methods.

Our method does not offer the fastest preprocessing, but we believe that this disadvantage is compensated for by the fact that it is the only method that can handle situations like the one in Figure 12. Note that Zipper (but not SC Toolbox) gives similarly accurate results, if it is used to compute the conformal maps $f_i = \varphi_i \circ g_i$ ($i = 1, 2$) that directly map x_i to 0 and y_i to 1, but according to the timings in Table 1, this is too slow for an interactive application. Moreover, Figure 15 shows that even in this setting, Zipper's boundary map, in contrast to ours, does not seem to converge to the ground truth. The plots show the maximum and the average distance

$$E_{\max} = \max_{k=1,\dots,m} \|f(v_k) - \tilde{f}(v_k)\|, \quad E_{\text{mean}} = \frac{1}{m}\sum_{k=1}^{m} \|f(v_k) - \tilde{f}(v_k)\|$$

between the images of the m boundary points of the *V-shape* domain under the exact conformal map f and the approximate conformal maps $\tilde{f}$ generated by Zipper and our tool, for increasing m.

Table 2 reports the CPU times during the exploration phase of our tool for the three examples above. We observe that updating the Poisson kernel at x_1 or x_2 by finding the triangle that contains x_1 or x_2 (the so-called 'point location' problem), computing ψ or ψ' as in Equation (10), and executing the matching algorithm comes at a negligible cost. Although we did not implement the equivalent step (computing $a_i = g_i(x_i)$ and $b_i = g_i(y_i)$, applying φ_i to the boundary points and matching them) for the other methods, we can safely assume that it would be similarly fast. The bottleneck is clearly the extension of the boundary map to the interior, but any other tool would have to perform this step, too. However, despite the additional cost for tracking the triangles that contain the interior constraints and actually rendering the mesh, our tool remains interactive, even for the high-resolution mesh used in Figure 13.

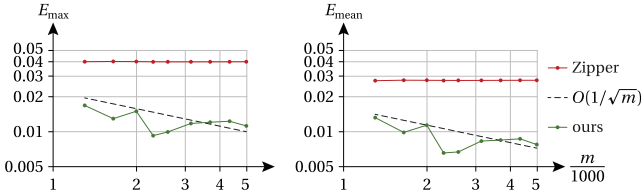

Figure 15: *Maximum and average distance between the exact images of the source boundary points and the images obtained by our method and Zipper for the example in Figure 12 with different m.*

5.3. Conformal parameterizations

The parameterization example in Figure 14 was inspired by Figure 15 in [SC18], which shows that BFF can find the unique conformal parameterization of a 3D hemisphere over an L-shaped domain and the domain itself, such that six equidistant boundary points correspond to the domain corners. Using this domain, one boundary and one interior correspondence from their result, we can reproduce this parameterization with our tool. We can further approximate all other conformal parameterizations over this and any other L-shaped domain with different edge lengths, but given the finite number of mesh boundary vertices, it is impossible to interactively find a map where all six domain corners correspond exactly to a mesh vertex.

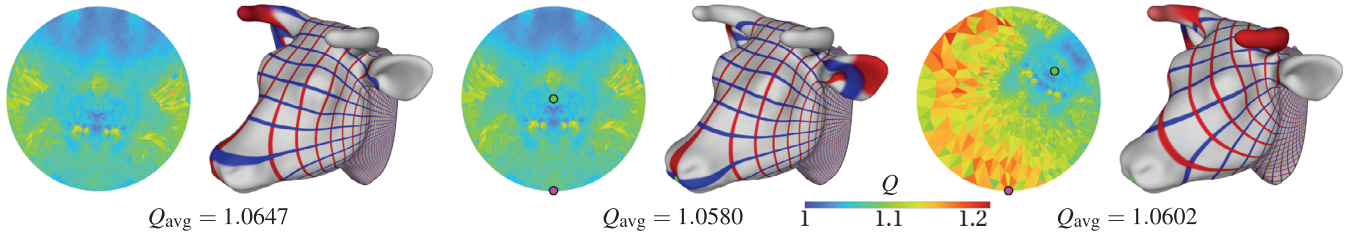

Figure 16: *Quasi-conformal error per triangle for different conformal maps from the* cow head *mesh ($m = 97$, $N = 29999$) to the unit disk. The average errors of the maps computed with our tool (centre and right) are similar to the average error of the map generated with BFF (left).*

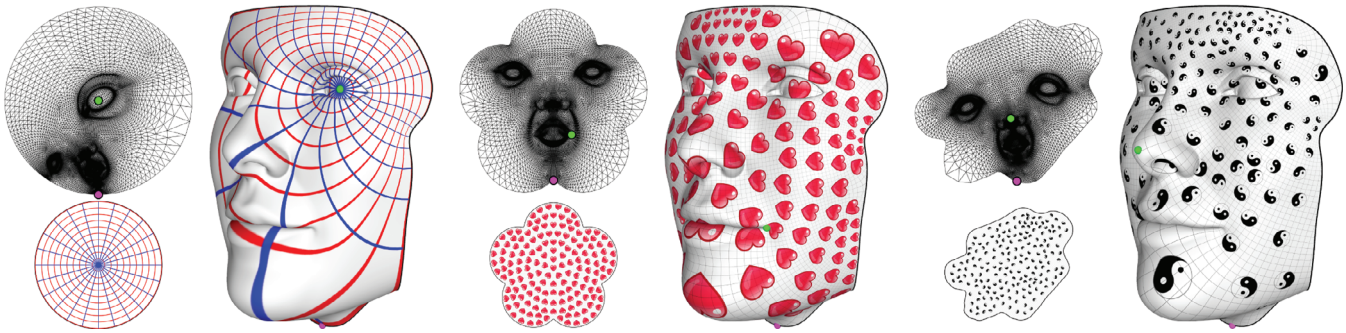

Figure 17: *With our tool we can conformally parameterize the* face *mesh ($m = 168$, $N = 34144$) over arbitrary planar domains and use this parameterization to conformally map a texture given inside these domains to the mesh surface.*

In Figure 16, we compare the quality (in terms of angle distortion) of our approximate conformal maps from a 3D mesh to the unit disk to the map obtained by BFF. Forcing the nose of the cow to be mapped to the centre of the disk gives a result that it very similar to BFF, but with a slightly smaller area-weighted average angle distortion. This average value grows only marginally if we move the interior constraint point closer to the boundary of the disk, even though this noticeably increases the quasi-conformal errors for some triangles near the opposite boundary, where the conformal factor of the boundary map is large.

More examples of conformal parameterizations over different planar domains, computed with our tool, can be seen in Figure 17 and the supplementary video.

6. Conclusion and Discussion

Computing conformal maps between arbitrary planar domains is a notoriously hard problem, but it seems that harmonic measure is just the right tool for approximating these maps with good accuracy at low computational cost. We have shown that harmonic measure can be derived very precisely from harmonic coordinates, which we compute in our preprocessing step. Once harmonic measure is available, a simple online matching algorithm very efficiently determines the boundary behaviour of the conformal map. In this context, a novel idea is the conversion of harmonic coordinates, which are a *discrete* approximation of the Poisson kernel, into a parametric and *continuous* approximation. The latter is crucial for the matching algorithm, because it needs to evaluate the harmonic measure (i.e., the integral of the Poisson kernel) at arbitrary parameter values, not only at those corresponding to the mesh boundary vertices. Once the conformal boundary map is known, it can easily be extended to the interior of the domain, and the fact that the precomputed harmonic coordinates can also be used for this is a fortunate coincidence.

Besides exploring planar conformal maps in real time, our interactive tool can also be used to conformally parameterize a genus-0 manifold 3D surface with boundary, given as a triangle mesh, over an arbitrary planar domain. The only existing method that addresses this problem is BFF [SC18], but it requires several iterations before converging to the desired shape and finds only *one* of the infinitely many conformal maps from the mesh to the parameter domain, while our tool allows to interactively compute and inspect *all* of them. In future work, it would be interesting to generalize it to an optimization method that finds the *best* conformal map with respect to some secondary criterion, for example, area distortion.

A potential limitation of using our tool for 3D mesh parameterization is because the cotangent weights of an arbitrary triangle mesh (in contrast to planar Delaunay triangulations) are not guaranteed to be positive. Consequently, the harmonic coordinates ϕ_{ik} can be negative and may result in approximate harmonic measures $\tilde{\omega}_x$ in Equation (11) that are not monotonically increasing, which in turn can cause the matching algorithm to not respect the ordering of the mapped boundary points. In practice, in all of our 3D examples we experienced no such problem in the mapping, despite the presence of many negative cotangent weights and some negative harmonic

coordinates. In any case, a possible solution to prevent this completely would be to use the cotangent weights derived from an alternative *intrinsic* Delaunay triangulation of the mesh vertices [BS07].

Acknowledgements

This work was supported by the Swiss National Science Foundation (SNF) under project number 188577. We thank the anonymous reviewers for their valuable comments and suggestions.

Open access publishing facilitated by Università della Svizzera italiana, as part of the Wiley - Università della Svizzera italiana agreement via the Consortium Of Swiss Academic Libraries.

Conflict of Interest Statement

We have no conflicts of interest to disclose.

Ethical Statement

We affirm that this research was conducted in accordance with ethical guidelines.

Appendix A: Harmonic Coordinates

The harmonic coordinates of the vertices of a mesh M stem from the standard finite element discretization of the Laplace equation with Dirichlet boundary conditions. We represent the harmonic coordinates $\phi_i = (\phi_{i1}, \dots, \phi_{im})$ of v_i as the i-th row of a matrix

$$\Phi = \begin{pmatrix} \bar{\Phi} \\ \hat{\Phi} \end{pmatrix}, \qquad \bar{\Phi} \in \mathbb{R}^{m\times m}, \quad \hat{\Phi} \in \mathbb{R}^{n\times m}.$$

While the harmonic coordinates of the boundary vertices $v_1, \dots, v_m$ are just the standard basis vectors of $\mathbb{R}^m$, that is, $\bar{\Phi}$ is just the $m \times m$ identity matrix, the harmonic coordinates of the interior vertices $\hat{\Phi}$ are obtained by solving the linear system

$$\hat{L}\hat{\Phi} = -\bar{L}, \tag{A.1}$$

where the non-zero off-diagonal elements of $\hat{L} \in \mathbb{R}^{n\times n}$ and the non-zero elements of $\bar{L} \in \mathbb{R}^{n\times m}$ are the *cotangent weights*

$$\hat{L}_{ij} = -\tfrac{1}{2}(\cot\alpha_{m+i,m+j} + \cot\alpha_{m+j,m+i}),$$
$$\bar{L}_{ik} = -\tfrac{1}{2}(\cot\alpha_{m+i,k} + \cot\alpha_{k,m+i})$$

for $i = 1, \dots, n$ and all $j \in \{1, \dots, n\}$ and $k \in \{1, \dots, m\}$ for which $[v_{m+i}, v_{m+j}]$ is an edge between two interior vertices of M and $[v_{m+i}, v_k]$ is an edge between an interior and a boundary vertex, and with α_{ab} denoting the angle of the triangle $[v_a, v_b, v_c]$ at v_c, hence

$$\cot\alpha_{ab} = \frac{\langle v_a - v_c, v_b - v_c\rangle}{\det(v_a - v_c, v_b - v_c)}.$$

Moreover, the diagonal elements of $\hat{L}$ are

$$\hat{L}_{ii} = -\sum_{j=1, j\neq i}^{n} \hat{L}_{ij} - \sum_{k=1}^{m} \bar{L}_{ik}$$

for $i = 1, \dots, n$ (see [CdGDS13, Section 6.2] for a derivation).

Appendix B: Approximating the Parametric Poisson Kernel

Denote the length of the k-th boundary edge $[v_k, v_{k+1}]$ by $e_k = \|v_{k+1} - v_k\|$ for $k = 1, \dots, m$ and their sum by $\tilde{\ell} = e_1 + \cdots + e_m$. Here, and whenever we work with indices related to boundary vertices of M, we treat these indices cyclically over $[1, \dots, m]$, that is, $v_{m+1} = v_1$ and $v_0 = v_m$. We further let $t_1 = 0$ and $t_{k+1} = t_k + e_k$ for $k = 1, \dots, m-1$ be the parameters of the boundary vertices with respect to the piecewise linear arc length parameterization $\tilde{p}$ of $\partial\tilde{\Omega}$, that is, $\tilde{p}(t_k) = v_k$. In addition, we let $t_{m+1} = t_m + e_m = \tilde{\ell}$.

For any vertex v_i of M, we then approximate P_{v_i} by the piecewise linear function $\tilde{P}_{v_i} : [0, \tilde{\ell}] \to \mathbb{R}$ with $\tilde{P}_{v_i}(t_k) = \psi_{ik}$ for $k = 1, \dots, m$ and $\tilde{P}_{v_i}(\tilde{\ell}) = \psi_{i1}$. To determine the unknowns ψ_{ik}, we recall that the k-th harmonic coordinate ϕ_{ik} of v_i approximates the value $h_k(v_i)$ of the harmonic function h_k that is linear along the boundary edges of M and zero at all boundary vertices, except v_k, where it is one, that is, $h(v_j) = \delta_{jk}$ for $j = 1, \dots, m$. Therefore, by the parametric equivalent of Equation (2),

$$\phi_{ik} \approx h_k(v_i) \approx \int_0^{\tilde{\ell}} \tilde{P}_{v_i}(t)\, h_k(\tilde{p}(t))\, \mathrm{d}t. \tag{B.1}$$

Since we assume $\tilde{P}_{v_i}$ to be piecewise linear, we can write the integral in terms of the unknowns ψ_{ik} as

$$\int_0^{\tilde{\ell}} \tilde{P}_{v_i}(t)\, h_k(\tilde{p}(t))\, \mathrm{d}t = \frac{e_{k-1}}{6}\psi_{i,k-1} + \frac{e_{k-1}+e_k}{3}\psi_{ik} + \frac{e_k}{6}\psi_{i,k+1}$$

and express Equation (B.1) compactly as $\Phi \approx \Psi E$, where $\Psi \in \mathbb{R}^{(n+m)\times m}$ is the matrix of all unknowns and $E \in \mathbb{R}^{m\times m}$ is the symmetric matrix with non-zero elements

$$E_{kk} = \frac{e_{k-1}+e_k}{3}, \qquad E_{k+1,k} = E_{k,k+1} = \frac{e_k}{6}$$

for $k = 1, \dots, m$. This suggests to set $\Psi = \Phi E^{-1}$, but this may lead to negative values ψ_{ik}. To avoid this problem, recall that $\psi_{i,k-1}$ and $\psi_{i,k+1}$ converge to $\psi_{i,k}$ with increasing boundary sample density, hence

$$\int_0^{\tilde{\ell}} \tilde{P}_{v_i}(t)\, h_k(\tilde{p}(t))\, \mathrm{d}t \approx \frac{e_{k-1}+e_k}{2}\psi_{ik}$$

and $\Phi \approx \Psi\tilde{E}$, where $\tilde{E} \in \mathbb{R}^{m\times m}$ is the diagonal matrix with elements $\tilde{E}_{kk} = (e_{k-1} + e_k)/2$. We therefore set the values of Ψ to

$$\psi_{ik} = \frac{2}{e_{k-1}+e_k}\phi_{ik}, \tag{B.2}$$

which is guaranteed to be positive. In the FEM literature, the process of replacing E with $\tilde{E}$ is known as 'mass lumping' with row sums [ZTZ13].

Appendix C: Pseudocode

To facilitate the implementation of our work, this appendix provides pseudocode for our interactive tool, as described in Section 4.

In the Main program, the routine LoadImage loads the texture image that is used during rendering. The vertices of mesh M_i $(i = 1, 2)$ are stored in a vector $v_i = (\bar{v}_i, \hat{v}_i)$, where $\bar{v}_i$ are the m_i boundary vertices and $\hat{v}_i$ are the n_i interior vertices of M_i. Likewise,

Algorithm C.1. MAIN($\partial\Omega_1, \partial\Omega_2, m_1, m_2$).

Input: Boundaries $\partial\Omega_1, \partial\Omega_2$ of source and target domain, and desired numbers (m_1, m_2) of boundary vertices.
Output: Exploration of all conformal maps from Ω_1 to Ω_2 by interactively changing the positions of the interior constraints x_1 and x_2 and the indices i_1 and i_2 of the boundary constraints y_{i_1} and y_{i_2}.

$I :=$ LOADIMAGE ▷ load texture image
$[M_1, e_1, \hat{G}_1, \bar{L}_1, \Psi_1] :=$ PREPROCESS($\partial\Omega_1, m_1$) ▷ prepare source
$[M_2, e_2, \hat{G}_2, \bar{L}_2, \Psi_2] :=$ PREPROCESS($\partial\Omega_2, m_2$) ▷ prepare target
repeat
  wait until user changes x_1, x_2, i_1, or i_2
  $\psi_1 :=$ UPDATE(x_1, M_1, Ψ_1) ▷ only if x_1 was changed
  $\psi_2 :=$ UPDATE(x_2, M_2, Ψ_2) ▷ only if x_2 was changed
  if forward mapping **then** ▷ *forward mapping*
    $\bar{w}_1 :=$ MATCHING($\bar{v}_1, e_1, e_2, \psi_1, \psi_2, i_1, i_2$) ▷ images of $\bar{v}_1$
    BACKSOLVE($\hat{L}_1, \hat{w}_1, -\bar{L}_1\bar{w}_1$) ▷ images of $\hat{v}_1$
    render M_1 with vertices w_1 and texture coordinates v_1
  else ▷ *backward mapping*
    $\bar{w}_2 :=$ MATCHING($\bar{v}_2, e_2, e_1, \psi_2, \psi_1, i_2, i_1$) ▷ images of $\bar{v}_2$
    BACKSOLVE($\hat{L}_2, \hat{w}_2, -\bar{L}_2\bar{w}_2$) ▷ images of $\hat{v}_2$
    render M_2 with vertices v_2 and texture coordinates w_2
until terminated by user

Algorithm C.2. PREPROCESS($\partial\Omega, m$).

Input: The boundary $\partial\Omega$ of a domain (given, e.g., as a dense polygon) and the desired number of m boundary vertices.
Output: A mesh M with boundary edge lengths e, the Cholesky factor $\hat{G}$ of the Laplace matrix, and the approximate Poisson kernels Ψ at the mesh vertices.

$\bar{v} :=$ UNIFORMLYSAMPLE($\partial\Omega, m$) ▷ generate boundary vertices
for $k = 1, \ldots, m$ **do**
  $e_i := \|v_{k+1} - v_k\|$ ▷ compute boundary edge lengths
$M :=$ DELAUNAY($\bar{v}$) ▷ Delaunay triangulation of $\bar{v}$ using *Triangle*
$[\hat{L}, \bar{L}] :=$ BUILDLAPLACE(M) ▷ system matrix and r.h.s. of Equation (A.1)
$\hat{G} :=$ CHOLESKYFACTOR($\hat{L}$) ▷ using CHOLMOD
BACKSOLVE($\hat{L}, \hat{\Phi}, -\bar{L}$) ▷ harmonic coord. of interior vertices
$\Phi := (I_m, \hat{\Phi})$ ▷ add harmonic coord. of boundary vertices
$\tilde{E} :=$ DIAG$\big((e_m + e_1), (e_1 + e_2), \ldots, (e_{m-1} + e_m)\big)/2$
$\Psi := \Phi\tilde{E}^{-1}$ ▷ scale Φ as in (B.2)
return $[M, e, \hat{G}, \bar{L}, \Psi]$

their images under the conformal map are stored as $w_i = (\bar{w}_i, \hat{w}_i)$. The CHOLMOD routine BACKSOLVE(A, x, b) solves the linear system $Ax = b$ by back substitution, using the Cholesky factor of A.

During the PREPROCESS phase, the routine UNIFORMLYSAMPLE generates m uniformly spaced vertices along $\partial\Omega$, which is given as a polygon itself in our implementation, but could also be given as a B-spline or some other curve. DELAUNAY computes a conforming Delaunay triangulation with the parameters stated in Section 4.1, the routine BUILDLAPLACE fills the matrices $\hat{L}$ and $\bar{L}$ with the *cotangent weights* and their sums as explained in the text following Equation (A.1) in Appendix A, the CHOLMOD routine CHOLESKYFACTOR computes the sparse Cholesky factor of the Laplace matrix $\hat{L}$, and DIAG generates a diagonal matrix.

Algorithm C.3. UPDATE(x, M, Ψ).

Input: A point x inside some triangle of mesh M and the approximate Poisson kernels Ψ at the mesh vertices.
Output: The approximate Poisson kernel ψ at x.

$T :=$ FINDTRIANGLE(x, M) ▷ $x \in T = [v_i, v_j, v_k]$
$[\varrho, \sigma, \tau] :=$ BARYCOORD(x, T) ▷ $x = \varrho v_i + \sigma v_j + \tau v_k$
$\psi := \varrho\psi_i + \sigma\psi_j + \tau\psi_k$ ▷ see (10)
return ψ

Algorithm C.4. MATCHING($v, e, e', \psi, \psi', k_0, l_0$).

Input: The source boundary vertices v, the source and target boundary edge lengths e and e', the approximate Poisson kernels ψ and ψ' of the source and target interior constraints, and the indices k_0 and l_0 of the source and target boundary constraints.
Output: The approximate conformal images w of the boundary vertices.

$A_1 := 0$
for $k = 1, \ldots, m$ **do**
  $A_{k+1} := A_k + e_{k+k_0-1}(\psi_{k+k_0-1} + \psi_{k+k_0})/2$
$B_1 := 0$
for $l = 1, \ldots, m'$ **do**
  $B_{l+1} := B_l + e'_{l+l_0-1}(\psi'_{l+l_0-1} + \psi'_{l+l_0})/2$
$l := 1$
for $k = 1, \ldots, m$ **do**
  while $A_k \geq B_{l+1}$ **do**
    $l := l + 1$
  $\Delta := \psi'_{l+1} - \psi_l$
  $\mu := (A_k - B_l)/(B_{l+1} - B_l)$
  $\xi := \psi'_l/\Delta$
  if $\Delta = 0$ **then**
    $\lambda := \mu$
  else
    $\lambda := \text{sign}(\Delta)\sqrt{\xi^2 + 2\mu\xi + \mu} - \xi$
  $w_{k+k_0-1} := (1 - \lambda)v'_{l+l_0-1} + \lambda v'_{l+l_0}$
return w

In the UPDATE routine, the FINDTRIANGLE procedure identifies the unique triangle T of M that contains x. We simply loop through the list of triangles until we find T, which turns out to be fast enough, but a more efficient point location method (e.g., a greedy walk from the previously identified triangle towards x) can be used. The routine BARYCOORD implements the standard formula for determining the normalized barycentric coordinates of x with respect to T as ratios of areas.

In the MATCHING algorithm, the variables m and m' represent the numbers of source and target boundary vertices, and they can be obtained, for example, by querying the sizes of e and e'. The pseudocode follows the explanation in Section 3.2, except that it does not assume the indices of the boundary constraints to be 1 and instead handles all necessary index shifts.

References

[Abi81] ABIKOFF W.: The uniformization theorem. *American Mathematical Monthly 88*, 8 (1981), 574–592. https://doi.org/10.1080/00029890.1981.11995320.

[BS07] Bobenko A. I., Springborn B. A.: A discrete Laplace–Beltrami operator for simplicial surfaces. *Discrete & Computational Geometry 38*, 4 (Dec. 2007), 740–756. https://doi.org/10.1007/s00454-007-9006-1.

[CdGDS13] Crane K., de Goes F., Desbrun M., Schröder P.: Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses* (New York, NY, July 2013), SIGGRAPH '13, Association for Computing Machinery. https://doi.org/10.1145/2504435.2504442.

[CDHR08] Chen Y., Davis T. A., Hager W. W., Rajamanickam S.: Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software 35*, 3 (Oct. 2008). https://doi.org/10.1145/1391989.1391995.

[CG17] Chen R., Gotsman C.: Approximating planar conformal maps using regular polygonal meshes. *Computer Graphics Forum 36*, 8 (Dec. 2017), 629–642. https://doi.org/10.1111/cgf.13157.

[DMA02] Desbrun M., Meyer M., Alliez P.: Intrinsic parameterizations of surface meshes. *Computer Graphics Forum 21*, 3 (Sept. 2002), 209–218. https://doi.org/10.1111/1467-8659.00580.

[DP93] DeLillo T. K., Pfaltzgraff J. A.: Extremal distance, harmonic measure and numerical conformal mapping. *Journal of Computational and Applied Mathematics 46*, 1–2 (June 1993), 103–113. https://doi.org/10.1016/0377-0427(93)90289-N.

[Dri25] Driscoll T. A.: Schwarz–Christoffel Toolbox. Github, 2025. Retrieved September 12, 2025. URL: https://github.com/tobydriscoll/sc-toolbox/releases/tag/v3.1.3.

[DSL19] Dym N., Slutsky R., Lipman Y.: Linear variational principle for Riemann mappings and discrete conformality. *Proceedings of the National Academy of Sciences 116*, 3 (Dec. 2019), 732–737. https://doi.org/10.1073/pnas.1809731116.

[DT02] Driscoll T. A., Trefethen L. N.: *Schwarz–Christoffel Mapping*, vol. *8* of Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, (2002).

[DV98] Driscoll T. A., Vavasis S. A.: Numerical conformal mapping using cross-ratios and Delaunay triangulation. *SIAM Journal on Scientific Computing 19*, 6 (1998), 1783–1803. https://doi.org/10.1137/S1064827596298580.

[For80] Fornberg B.: A numerical method for conformal mappings. *SIAM Journal on Scientific and Statistical Computing 1*, 3 (1980), 386–400. https://doi.org/10.1137/0901027.

[JMD*07] Joshi P., Meyer M., DeRose T., Green B., Sanocki T.: Harmonic coordinates for character articulation. *ACM Transactions on Graphics 26*, 3 (July 2007). https://doi.org/10.1145/1276377.1276466.

[Kak44] Kakutani S.: Two-dimensional Brownian motion and harmonic functions. *Proceedings of the Imperial Academy 20*, 10 (1944), 706–714. https://doi.org/10.3792/pia/1195572706.

[KLYY21] Kuo Y.-C., Lin W.-W., Yueh M.-H., Yau S.-T.: Convergent conformal energy minimization for the computation of disk parameterizations. *SIAM Journal on Imaging Sciences 14*, 4 (2021), 1790–1815. https://doi.org/10.1137/21M1415443.

[Kra16] Krantz S. G.: *The Theory and Practice of Conformal Geometry*, vol. *156* of Cambridge Tracts in Mathematics. Dover Publications, Mineola, NY, (2016).

[KSS06] Kharevych L., Springborn B., Schröder P.: Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics 25*, 2 (Apr. 2006), 412–438. https://doi.org/10.1145/1138450.1138461.

[Kyt19] Kythe P. K.: *Handbook of Conformal Mappings and Applications*. CRC Press, Boca Raton, FL, (2019). https://doi.org/10.1201/9781315180236.

[Lan99] Lang S.: *Complex Analysis*, 4th ed., vol. *103* of Graduate Texts in Mathematics. Springer, New York, NY, (1999). https://doi.org/10.1007/978-1-4757-3083-8.

[LLCO08] Lipman Y., Levin D., Cohen-Or D.: Green coordinates. *ACM Transactions on Graphics 27*, 3 (Aug. 2008). https://doi.org/10.1145/1360612.1360677.

[LPRM02] Lévy B., Petitjean S., Ray N., Maillot J.: Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics 21*, 3 (July 2002), 362–371. https://doi.org/10.1145/566654.566590.

[MR07] Marshall D. E., Rohde S.: Convergence of a variant of the zipper algorithm for conformal mapping. *SIAM Journal on Numerical Analysis 45*, 6 (2007), 2577–2609. https://doi.org/10.1137/060659119.

[MTAD08] Mullen P., Tong Y., Alliez P., Desbrun M.: Spectral conformal parameterization. *Computer Graphics Forum 27*, 5 (July 2008), 1487–1494. https://doi.org/10.1111/j.1467-8659.2008.01289.x.

[SBC16] Segall A., Ben-Chen M.: Iterative closest conformal maps between planar domains. *Computer Graphics Forum 35*, 5 (Aug. 2016), 33–40. https://doi.org/10.1111/cgf.12961.

[SC18] Sawhney R., Crane K.: Boundary first flattening. *ACM Transactions on Graphics 37*, 1 (Feb. 2018). https://doi.org/10.1145/3132705.

[SdS01] Sheffer A., de Sturler E.: Parameterization of faceted surfaces for meshing using angle-based flattening. *Engineering with Computers 17*, 3 (Oct. 2001), 326–337. https://doi.org/10.1007/PL00013391.

[She96] Shewchuk J. R.: Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied Computational*

Geometry: Towards Geometric Engineering, Lin M. C., Manocha D., (Eds.), vol. *1148* of Lecture Notes in Computer Science. Springer-Verlag, (May 1996), pp. 203–222.

[SLMB05] Sheffer A., Lévy B., Mogilnitsky M., Bogomyakov A.: ABF++: fast and robust angle based flattening. *ACM Transactions on Graphics 24*, 2 (Apr. 2005), 311–330. https://doi.org/10.1145/1061347.1061354.

[SSP08] Springborn B., Schröder P., Pinkall U.: Conformal equivalence of triangle meshes. *ACM Transactions on Graphics 27*, 3 (Aug. 2008). https://doi.org/10.1145/1360612.1360676.

[TZ22] Tan Z.-H., Zhang Z.: Stable discrete minimization of conformal energy for disk conformal parameterization, (2022). http://arxiv.org/abs/2210.09125.

[WBCG09] Weber O., Ben-Chen M., Gotsman C.: Complex barycentric coordinates with applications to planar shape deformation. *Computer Graphics Forum 28*, 2 (Apr. 2009), 587–597. https://doi.org/10.1111/j.1467-8659.2009.01399.x.

[Weg86] Wegmann R.: An iterative method for conformal mapping. *Journal of Computational and Applied Mathematics 14*, 1–2 (Feb. 1986), 7–18. https://doi.org/10.1016/0377-0427(86)90128-7.

[WG10] Weber O., Gotsman C.: Controllable conformal maps for shape deformation and interpolation. *ACM Transactions on Graphics 29*, 4 (July 2010). https://doi.org/10.1145/1778765.177881.

[ZLS07] Zayer R., Lévy B., Seidel H.-P.: Linear angle based parameterization. In *Proceedings of the Fifth Symposium on Geometry Processing* (Aire-la-Ville, July 2007), SGP '07, Eurographics Association, pp. 135–141. https://doi.org/10.2312/SGP/SGP07/135-141.

[ZTZ13] Zienkiewicz O. C., Taylor R. L., Zhu J. Z.: *The Finite Element Method: Its Basis and Fundamentals*, 7th ed. Butterworth-Heinemann, Oxford, (2013). https://doi.org/10.1016/C2009-0-24909-9.