Variational Surface Cutting

NICHOLAS SHARP and KEENAN CRANE, Carnegie Mellon University

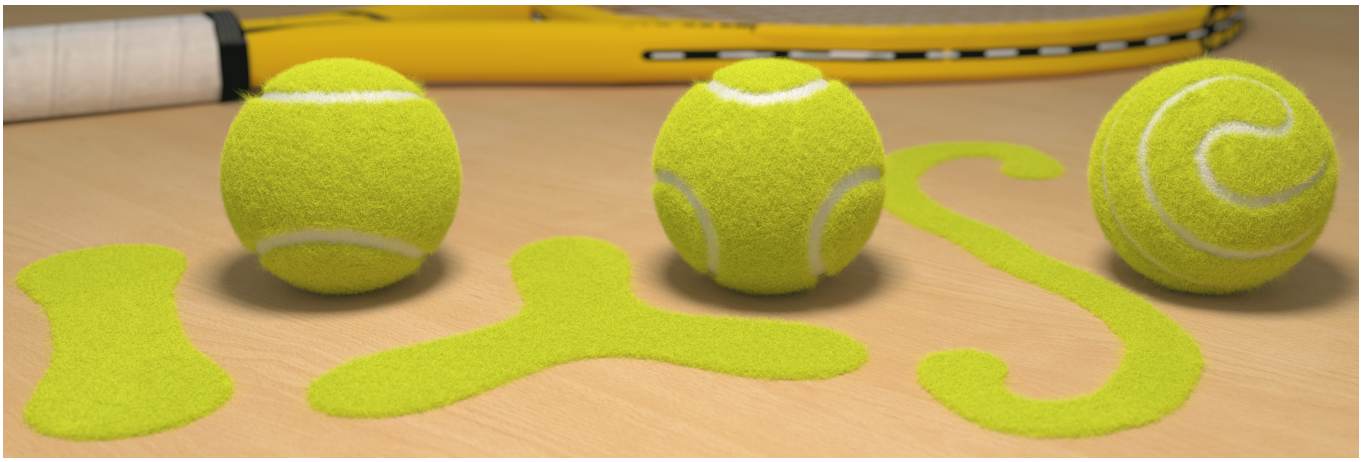


Fig. 1. Our flow evolves a given cut toward one that yields lower area distortion when the surface is flattened. Here we start with a cut through the equator and flow to cuts of three different lengths (visualized as seams on the tennis ball), resulting in progressively easier flattenings into the plane.

This paper develops a global variational approach to cutting curved surfaces so that they can be flattened into the plane with low metric distortion. Such cuts are a critical component in a variety of algorithms that seek to parameterize surfaces over flat domains, or fabricate structures from flat materials. Rather than evaluate the quality of a cut solely based on properties of the curve itself (*e.g.*, its length or curvature), we formulate a flow that directly optimizes the distortion induced by cutting and flattening. Notably, we do not have to explicitly parameterize the surface in order to evaluate the cost of a cut, but can instead integrate a simple evolution equation defined on the cut curve itself. We arrive at this flow via a novel application of *shape derivatives* to the *Yamabe equation* from conformal geometry. We then develop an Eulerian numerical integrator on triangulated surfaces, which does not restrict cuts to mesh edges and can incorporate user-defined data such as importance or occlusion. The resulting cut curves can be used to drive distortion to arbitrarily low levels, and have a very different character from cuts obtained via purely discrete formulations. We briefly explore potential applications to computational design, as well as connections to space filling curves and the problem of uniform heat distribution.

CCS Concepts: • **Computing methodologies** → **Mesh geometry models**; • **Mathematics of computing** → *Continuous optimization*;

Additional Key Words and Phrases: discrete differential geometry, geometry processing

Authors' address: Nicholas Sharp; Keenan Crane, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213.

0730-0301/2018/8-ART156 $15.00
https://doi.org/10.1145/3197517.3201356

ACM Reference Format:
Nicholas Sharp and Keenan Crane. 2018. Variational Surface Cutting. *ACM Trans. Graph.* 37, 4, Article 156 (August 2018), 13 pages. https://doi.org/10.1145/3197517.3201356

1 INTRODUCTION

Which cut allows a curved surface to be most easily flattened? In general, surfaces cannot be flattened without some distortion of areas or angles—a classic example is maps of the Earth, which invariably distort the relative shape or size of land masses. Broader interest in cutting and flattening curved surfaces arises in a variety of problems in digital geometry processing, scientific computing, and computational design, where distortion affects things like signal quality, numerical accuracy, and material stress during manufacturing. The general strategy for reducing distortion is to make cuts in the surface, much as a designer might accommodate curvature by incorporating seams into a pattern for a garment. However, the question of *where* to place cuts remains an ongoing challenge.

A common idea is to view cutting as an inherently discrete problem: find the best cut along edges of a triangulation. However, these edges may have no meaning in a context like physical manufacturing, where the triangulation is merely a proxy for a smooth surface. In this paper we instead start in the smooth setting, and show that the distortion of a cut can be minimized by integrating a remarkably simple flow. In particular, if n is the unit normal of a curve γ, then we simply need to evolve γ according to the equation

$$\boxed{\frac{d}{dt}\gamma = \left(\frac{\partial u}{\partial n}\right)^2 n,} \tag{1}$$

where u is the logarithm of the scale distortion induced by cutting the surface along γ and applying a conformal (*i.e.*, angle-preserving)

Fig. 2. A variational formulation of cutting makes it easy to explore low-distortion design alternatives. *Top:* By adjusting parameters in our flow, we obtain designs that look like either classic panelizations for furniture *(top left)* or designs more reminiscent of race car seats *(top right)*. Alternatively, we can explore how the changing geometry of a surface affects the cuts needed to easily upholster it *(bottom)*.

flattening. The scale factor u can in turn be obtained by solving the *Yamabe equation* $\Delta u = -K$, where K is the Gaussian curvature of the surface (Sec. 3). Since we never require an explicit map to the plane, this approach avoids many of the difficulties typically associated with mesh parameterization. We develop a numerical integrator for several variations of this flow, based on the machinery of *shape optimization* (Sec. 5).

The fact that we can apply shape optimization to the cutting problem is a very special property of working with *conformal* maps, rather than other types of surface flattenings. In the conformal setting, one can express a flattening indirectly in terms of the log scale factor u, rather than directly through a map $f : M \to \mathbb{R}^2$. The Yamabe equation then gives a direct relationship between u and the cut curve γ, allowing us to formulate the problem of minimal-distortion cutting as a standard PDE-constrained optimization problem. If one attempts the same approach for other common flattening energies, the resulting optimization problem is constrained by yet another optimization problem (consider for instance defining f as the minimizer of a nonlinear elastic energy).

As stated however, our problem is ill-posed, since distortion can always be reduced by making longer and longer cuts. We therefore penalize the length of the cut by adding a simple curvature term to the flow velocity, *à la* curve shortening flow (Sec. 3.2.1). By adjusting the influence of this term, we can trade off between more traditional cuts yielding small compact regions, and longer winding cuts that can drive area distortion to arbitrarily low levels (Fig. 3). After considering several variations of our flow in the smooth setting (Sec. 3) we develop an implicit Eulerian discretization and accompanying numerical integrator on triangle meshes (Sec. 5). The resulting algorithm yields cuts of a very different nature than those found in previous work, as explored in Sec. 7.

2 RELATED WORK

The problem of cutting a curved surface into pieces that can be easily flattened is as old as the problem of mapmaking itself. More recently, questions about algorithmically cutting and flattening general curved surfaces have received considerable attention.

Topological Cutting. Perhaps the most basic question is how to cut a given surface into one or more topological disks, without reference to the distortion of a subsequent flattening. Here, the quality of the cut can be measured via basic geometric quantities like total length. Even for this simple problem, finding the optimal cut is intractable: in particular, Erickson and Har-Peled [2004] show that finding the shortest collection of edges that cuts a polyhedral surface into a disk is NP-hard. For this reason, we should not expect that our method (or any method) will find a globally optimal solution. Restricted versions of this problem can however be efficiently solved [Erickson and Whittlesey 2005]. Another starting point, widely used in graph theory, are *Cheeger cuts*, which seek the shortest cut that partitions a domain into two pieces of near-equal size [Gotsman 2003], again with no consideration of distortion (or even topology).

Cutting for Flattening. A more challenging question is how to find cuts that yield low parametric distortion. A basic first question is: what kinds of cuts should one even consider? One idea is to partition the surface into many small compact regions, for example by greedy region growing or iterative clustering [Sander et al. 2001; Lévy et al. 2002; Sorkine et al. 2002; Zhou et al. 2004; Julius et al. 2005; Yamauchi et al. 2005]. Another way is to partition into strips with long winding boundaries, either by user-guided or automatic methods [Mitani and Suzuki 2004; Tang et al. 2016]. A third approach is to first identify a special set of vertices akin to the "darts" used in tailoring, then find a Steiner tree-like cut that connects these points [Sheffer and Hart 2002]—an important special case are *cone singularities* [Kharevych et al. 2006], which concentrate all the curvature into the special points rather than anywhere along the cut [Ben-Chen et al. 2008;

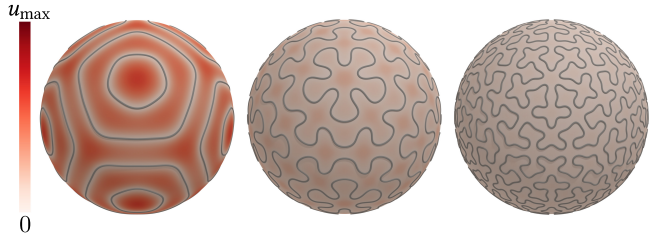


Fig. 3. Without penalizing length, a cut that seeks to minimize distortion will evolve into a curve reminiscent of space filling curves. Surface coloring indicates area distortion (here we explicitly enforce icosahedral symmetry).

Springborn et al. 2008; Soliman et al. 2018]. Each type of cut comes with its own advantages and disadvantages: compact regions are easy to pack into an atlas, but may not be optimal in terms of area distortion. Conversely, long winding cuts may yield lower distortion and lend themselves to easy fabrication [Schüller et al. 2017], at the cost of less compact layouts in the plane. Finally, cone singularities pave the way to globally seamless parameterizations with low average distortion, at the cost of extremely high distortion near cone points (which can be problematic for fabrication). We do not advocate for any one of these strategies, but rather observe that our flow naturally yields all three types of cuts depending on initial conditions. In particular: a network of patches will evolve into small compact regions; a single closed curve will evolve into long winding strips, and, interestingly enough, cone-like features will automatically emerge in situations where they are beneficial (see for example Fig. 2 and 16).

Cutting Algorithms. Algorithmically, how can one find cuts of low distortion? The type of cuts considered has motivated past approaches. Early algorithms used basic proxies for parameteric distortion, such as the total curvature of a patch [Yamauchi et al. 2005], later methods focused more directly on measuring *developability*, *e.g.*, by fitting primitives like cones [Julius et al. 2005] or considering the Gauss image [Decaudin et al. 2006]. Like our method, recent work by Poranne et al. [2017] directly uses the distortion of the flattening to guide optimization of the cut. In our case we completely subvert the need to work with an explicit parameterization, avoiding ensuing difficulties such as noninjectivity. Instead, the configuration space of our problem consists purely of the cut curve itself, from which the distortion can be directly determined.

Since our method can take any initial cut as input (including those generated by all previous work), we can always find cuts of smaller distortion (or a better distortion/length trade off) by simply running our flow for some amount of time. Moreover, most existing methods are inherently discrete, making the solution highly dependent on the input tessellation—which in a context like digital manufacturing is a superficial feature of discretization. In contrast, our cut flow depends primarily on the geometry of the underlying smooth surface, can be augmented to incorporate physically relevant terms, and generally produces cuts of a very different nature from traditional surface cuts (consider, for example, Fig. 1 and 6). These features open the door to a wide variety of possibilities beyond classic problems like texture atlas generation: it is unclear, for instance, what it would even mean to apply standard cutting algorithms to problems like Fig. 13, 14, or 18. Overall, the variational approach to cutting appears to be a powerful tool that both complements and broadens the scope of the existing surface cutting toolbox.

3 SMOOTH FORMULATION

In contrast to all previous work on cut optimization, which considers discrete paths along edges, we formulate our problem in the smooth setting, where we have a richer space of candidate solutions. Our starting point is a variational problem that penalizes distortion while trying to keep the cut length short. To solve this problem, we apply the machinery of *shape derivatives* (Sec. 3.3) to an optimization problem constrained by the *Yamabe equation* from conformal geometry (Sec. 3.2). Using the *Dirichlet energy* to measure area distortion leads to a particularly simple formulation (Sec. 3.4), though other energies can easily be incorporated into our framework (Sec. 3.5). An implicit formulation (Sec. 4) then leads naturally to our discrete algorithm (Sec. 5).

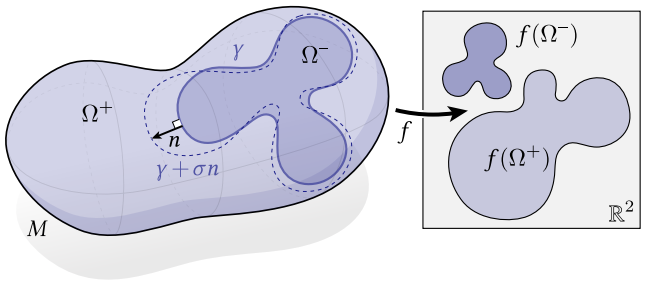


Fig. 4. We optimize a curve γ on a surface M to reduce the distortion of a conformal map $f : M \setminus \gamma \to \mathbb{R}^2$ while keeping length short. The key task is computing the evolution speed σ in the normal direction n.

The flows we obtain naturally relate to existing work on geometric PDEs. Perhaps the most classic example is *curve shortening flow* [Gage and Hamilton 1986; Grayson 1987], which we effectively use as a regularizer. The tension between length and distortion is reminiscent of the classic isoperimetric problem, *i.e.*, finding the largest area enclosed by a curve of fixed length. More recently there has been a fair bit of work on *nonlocal isoperimetric problems* (*e.g.*, [Alberti et al. 2009]); our flow provides an interesting link between such problems and the task of conformal flattening.

3.1 Background and Notation

Our main object is a surface M which is cut by a network of curves γ into one or more disk-like pieces, which constitute the *cut surface* M_γ (Fig. 4). Any such surface admits a *conformal flattening* $f : M_\gamma \to \mathbb{R}^2$, *i.e.*, an angle-preserving map to the Euclidean plane. Although angles are exactly preserved, lengths and areas may be significantly distorted. This distortion can be quantified via a function $u : M_\gamma \to \mathbb{R}$ called the *log conformal scale factor* (defined below); for a map with no distortion, $u \equiv 0$. Our goal is therefore to find an evolution of the cut γ that drives u toward zero.

More formally, let g be a Riemannian metric on a surface M of any topology, with or without boundary, and let γ be a network of curves cutting M into a collection of topological disks $M_\gamma := M \setminus \gamma$. A differentiable map $f : M_\gamma \to \mathbb{R}^2$ is *conformal* if at all points $p \in M_\gamma$ and for all tangent vectors $X, Y \in T_pM_\gamma$, we have $df_p(X) \cdot df_p(Y) = e^{2u(p)} g_p(X, Y)$ for some function $u : M_\gamma \to \mathbb{R}$, *i.e.*, if the inner product changes only by some positive scale factor e^{2u} that varies across the domain. In many situations, working directly with the logarithm u is more natural, since scaling is multiplicative. Throughout we use κ and K to denote the geodesic curvature of a curve and the Gaussian curvature of a surface (*resp.*). We use Δ to denote the positive semidefinite Laplace-Beltrami operator, and $\Delta_a := -\nabla \cdot a\nabla$ to denote an inhomogeneous Laplacian, where $a : M \to \mathbb{R}$ is a positive function.

3.2 Dirichlet Distortion Energy

How should one measure total area distortion? One natural approach is the *Dirichlet energy* of the conformal factor, which measures the overall variation in scaling across the domain:

$$E_D := \int_{M_\gamma} |\nabla u|^2 dA. \tag{2}$$

If E_D is zero, it means scaling is *constant*, *i.e.*, up to a global scale there is no area distortion. Springborn *et al.* [Springborn et al. 2008, Appendix E] show that for a *fixed* cut γ, E_D is minimized by any conformal map with uniform boundary scaling—in particular, we will assume that $u|_{\partial M} = 0$, corresponding to isometry (*i.e.*, no change in length). Crucially, this optimal scale factor can be obtained without explicitly computing the flattening f, but by instead solving the *Yamabe equation*

$$\begin{array}{rcll} \Delta u & = & -K & \text{on} \quad M_\gamma \\ u & = & 0 & \text{on} \quad \partial M_\gamma \end{array} \tag{3}$$

(see Aubin [2013, Chapter 5]). From a practical point of view, the fact that we do not have to explicitly compute the map f avoids many of the challenges typically associated with computing a conformal flattening with prescribed boundary scaling. Importantly, since the solution to Eqn. 3 already minimizes distortion for a fixed cut, the only way to further reduce E_D is to optimize the cut curve γ.

3.2.1 Length Regularization. As stated, the problem of finding the cut that minimizes parametric distortion is ill-defined (*i.e.*, has no minimizer), since one can always drive the distortion closer to zero by making longer and longer cuts. We therefore regularize this problem by penalizing the total cut length

$$E_L(\gamma) := \frac{1}{2} \int_\gamma ds, \tag{4}$$

where ds is the usual measure of arc length along the cut. Together, the parametric distortion and length regularization terms yield a combined energy that characterizes desirable cuts:

$$E(\gamma) := E_D(\gamma) + \alpha_L E_L(\gamma). \tag{5}$$

The coefficient $\alpha_L > 0$ determines the relative strength of length regularization, providing a trade off between the amount of distortion and the complexity of the cut.

3.3 Shape Optimization

To flow towards an optimal cut, we need an expression for the gradient of the energy with respect to variations of the cut. We first give a brief overview of general shape derivatives before applying this machinery to our particular problem.

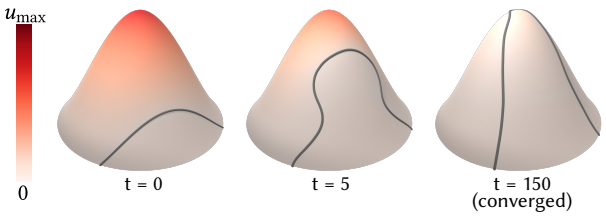


Fig. 5. Minimizing distortion evolves a given curve *(left)* toward a longer cut through regions of high curvature *(right)*.

3.3.1 Shape Derivatives. Consider a shape modeled as a subdomain Ω' of a larger domain Ω (Fig. 4). The *shape derivative* D_σ evaluates how a given objective changes with respect to the velocity $\sigma : \partial\Omega' \to \mathbb{R}$ along a canonical direction n normal to the boundary. (In the case of cutting, $\gamma = \partial\Omega'$.)

Fixed Functions. Consider first the total area of a region Ω' or the total length of its boundary; in this case, the shape derivatives with respect to a normal perturbation σ are (*resp.*)

$$\begin{aligned} D_\sigma \int_{\Omega'} dA &= \int_{\partial\Omega'} \sigma \, ds, \\ D_\sigma \int_{\partial\Omega'} ds &= \int_{\partial\Omega'} \kappa\sigma \, ds. \end{aligned} \tag{6}$$

The former reflects the fact that the area grows in proportion to the boundary velocity σ; the latter reflects the well-known fact that the rate of length variation is proportional to geodesic curvature κ. More generally, consider a function $\psi : \Omega \to \mathbb{R}$ which is independent of Ω'. Then

$$D_\sigma \int_{\Omega'} \psi \, dA = \int_{\partial\Omega'} \psi\sigma \, ds, \tag{7}$$

and

$$D_\sigma \int_{\partial\Omega'} \psi \, dA = \int_{\partial\Omega'} \left(\frac{\partial\psi}{\partial n} + \kappa\psi \right) \sigma \, ds. \tag{8}$$

The additional term $\partial\psi/\partial n$ in the second expression accounts for the fact that the value along the boundary is changing.

Céa's method. The rules above cannot be directly applied to a function ψ that depends on the shape of Ω' itself. In particular, taking the shape derivative becomes more challenging if ψ depends on the solution to a PDE over Ω' (such as the Yamabe equation). For such objectives, Céa [1986] provides a general framework for deriving the shape derivative, via the method of Lagrange multipliers.

In general, consider an objective of the form

$$J(\Omega') := \int_{\Omega'} j(u) \, dA, \tag{9}$$

where u is the solution to a Poisson equation on Ω', and j is a function that depends pointwise on u (and possibly its derivatives). More precisely, u solves the Poisson equation

$$\begin{array}{rcll} \Delta u & = & f & \text{on} \quad \Omega' \subset \Omega \\ u & = & 0 & \text{on} \quad \partial\Omega' \end{array} \tag{10}$$

for a fixed real-valued function f defined over all of Ω. The associated Lagrangian is

$$\mathcal{L}_{\Omega'}(u, p, \lambda) := \int_{\Omega'} j(u) \, dA + \int_{\Omega'} p(\Delta u - f) \, dA + \int_{\partial\Omega'} \lambda u \, ds, \tag{11}$$

where $p : \Omega' \to \mathbb{R}$ and $\lambda : \partial\Omega' \to \mathbb{R}$ are Lagrange multipliers. The key observation of Céa's method (which we will not justify here) is that the shape derivative of J can be obtained by taking the shape derivative of $\mathcal{L}$ at a critical point. The latter is easier to evaluate, since we can just apply the ordinary rules from Equations 7 and 8. To do so, one must first find a critical point by solving for the Lagrange multipliers—the *adjoint problems* that characterize this critical point depend on the particular choice of integrand $j(u)$.

Once u, p, and λ have been determined, applying Equations 7 and 8 to $\mathcal{L}$ yields a general form for the shape derivative of J:

$$D_\sigma J(\Omega') = D_\sigma \mathcal{L}(\Omega') = \int_{\partial\Omega'} \left(j(u) + \lambda \frac{\partial u}{\partial n} \right) \sigma \, ds. \tag{12}$$

Different energies will of course yield different shape derivatives—the only real change is the form of the adjoint problems, which can be obtained by taking variations of the Lagrangian with respect to u, p, and λ (and simplifying via Stokes' theorem). Plugging the solutions to these problems into Eqn. 12 yields a final expression for the shape derivative D_σ (Table 1 gives several examples).

3.3.2 Shape Derivative of Dirichlet Energy. For the energy $J = E_D$, several important simplifications occur. First, the adjoint problem for λ amounts to just $\lambda = 0$, *i.e.*, at a critical point the Lagrange multiplier for the constraint $u|_{\partial\Omega'} = 0$ is not active, in agreement with the observation by Springborn *et al.* that constant scaling along the boundary yields minimal Dirichlet energy. As a result, the second term in Eqn. 12 vanishes, and one does not have to solve an adjoint problem for p. Moreover, along $\partial\Omega' = \gamma$ we have

$$j(u) = |\nabla u|^2 = \left(\frac{\partial u}{\partial n} \right)^2 + \left(\frac{\partial u}{\partial n^\perp} \right)^2,$$

where $n^\perp$ denotes the direction tangent to the boundary. But since u is constant along the boundary, Eqn. 12 reduces to just

$$D_\sigma E_D(\gamma) = \int_\gamma \left(\frac{\partial u}{\partial n} \right)^2 \sigma \, ds. \tag{13}$$

3.4 Gradient Flow

For an arbitrary perturbation σ, Eqn. 6 and Eqn. 13 provide directional derivatives for the length and distortion terms, *resp.*, of the combined energy $E(\gamma)$ from Eqn. 5. Since we want to minimize distortion on *both* sides of the cut, the overall derivative is

$$D_\sigma E(\gamma) = \int_\gamma \Bigg(\underbrace{\left(\frac{\partial u_+}{\partial n} \right)^2}_{\sigma_{D+}} - \underbrace{\left(\frac{\partial u_-}{\partial n} \right)^2}_{\sigma_{D-}} + \underbrace{\alpha_L \kappa_\gamma}_{\sigma_L} \Bigg) \sigma \, ds, \tag{14}$$

where u_+ and u_- denote scale factors on either side of γ. The perturbation corresponding to the gradient of E is then

$$\sigma^* = \sigma_{D+} - \sigma_{D-} + \sigma_L, \tag{15}$$

where the terms $\sigma_{D\pm}$ and σ_L account for area distortion and cut length, *resp.* The cut is hence improved by integrating the flow

$$\frac{d}{dt}\gamma = -\sigma^* n. \tag{16}$$

3.5 Other Distortion Energies

We briefly consider several alternatives of the Dirichlet distortion energy which can be useful in applications; adjoint problems and shape derivatives for all energies are summarized in Table 1.

3.5.1 Rescaled Dirichlet Energy. The Dirichlet energy $E_D(\gamma)$ penalizes distortion uniformly over the surface; a more general spatially-varying penalty is provided by the energy

$$E_{D'}(\gamma) := \int_M a |\nabla u|^2 dA, \tag{17}$$

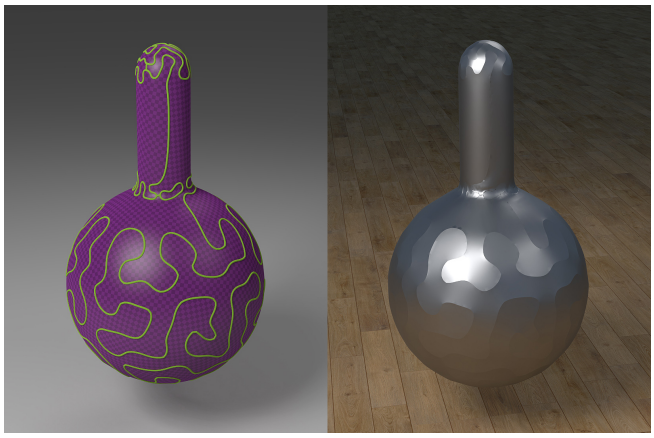


Fig. 6. *Left:* under a conformal cut flow, curves develop more oscillations in regions of high curvature, where the surface is hard to flatten. *Right:* since long cuts almost completely eliminate area distortion, the surface can be well-approximated by cutting the flattened shape from a sheet of inextensible material. To get a sense of approximation error, we here compute a 3D embedding with edge lengths that closely match the flattened mesh.

where the function $a : M \to \mathbb{R}_{>0}$ describes the rescaling at each point. The resulting gradient flow is then identical to Eqn. 16, except that the distortion terms on each side of the cut become

$$\tilde{\sigma}_{D\pm} = -a \left(\frac{\partial u_\pm}{\partial n} \right)^2 - \frac{\partial u_\pm}{\partial n} \frac{\partial p_\pm}{\partial n}, \tag{18}$$

and we must solve a separate adjoint problem for p (given in Table 1).

Curvature Based Rescaling. Without rescaling, the flow described by Eqn. 16 evolves much more quickly in regions of high curvature. Rescaling the energy by a local feature size $r : M \to \mathbb{R}_{>0}$ helps to reduce the time step restriction for surfaces with highly variable curvature. In practice, we use the intrinsic feature size $r := 1/(\sqrt{|K|} + r_\varepsilon^{-1})$, with a small numerical constant $r_\varepsilon > 0$. This feature size is small in tightly curved regions and large in flat regions. Rescaling by $a = r^3$ then yields a flow that is locally scale invariant, as revealed by a simple dimensional analysis.

3.5.2 Pointwise Energy. An alternative is to consider the integrand $b(u)$, where $b : \mathbb{R} \to \mathbb{R}$ is any function that depends pointwise on the value of u (but not its derivatives). We can again apply Céa's method to arrive at a shape derivative for this energy, as shown in the final row of Table 1.

Hencky Energy. An important special case is $b(u) = u^2$, yielding the energy

$$E_H(\gamma) := \int_M u^2 dA. \tag{19}$$

Up to material constants, this energy corresponds to the *true strain* or *Hencky strain* [Hencky 1928] from mechanics (this formulation is equivalent to the more traditional formulation of Hencky strain where u encodes the scaling from the flat reference domain to the curved surface—consider in particular that $\Delta = e^{-2u}\Delta_{\mathbb{R}^2}$.) This energy is useful for physical fabrication, as discussed in Sec. 7.2.

Table 1. Shape derivatives for the energies considered in this paper. In all cases the adjoint problem for p is subject to zero Dirichlet boundary conditions.

	Energy	Adjoint p	Boundary adjoint λ	Gradient	Comments
Dirichlet	$\int_{\Omega'} \lvert\nabla u\rvert^2 dA$	$\Delta p = -2\Delta u$	$\lambda = 0$	$\left(\frac{\partial u}{\partial n}\right)^2$	problem is self-adjoint
Scaled Dirichlet	$\int_{\Omega'} a\lvert\nabla u\rvert^2 dA$	$\Delta p = -2\Delta_a u$	$\lambda = -\frac{\partial p}{\partial n} - 2a\frac{\partial u}{\partial n}$	$-a\left(\frac{\partial u}{\partial n}\right)^2 - \frac{\partial u}{\partial n}\frac{\partial p}{\partial n}$	$\Delta_a := -\nabla \cdot a\nabla$
Pointwise	$\int_{\Omega'} b(u) dA$	$\Delta p = -b'(u)$	$\lambda = -\frac{\partial p}{\partial n}$	$b(u) - \frac{\partial u}{\partial n}\frac{\partial p}{\partial n}$	b' denotes the ordinary pointwise derivative

4 LEVEL SET FORMULATION

The cut flow from Sec. 3 could be directly evaluated on an explicit curve, but here we instead reformulate the evolution on an implicit signed distance function representation, as is common in shape optimization [Allaire et al. 2004]. Although this formulation cannot represent open curves, it enables easy changes in the topology of γ, as well as stable implicit integration. For simplicity, we here consider a single closed curve; multiple regions are discussed in App. A.

4.1 Cut Evolution

Any closed curve $\gamma \subset M$ can be represented as the zero level set of its signed distance function $\phi : M \to \mathbb{R}$; by convention, we assume that the gradient of ϕ points in the direction n of the outward unit normal of the curve, *i.e.*, $\nabla\phi = n$. In general, the advection equation for a scalar function ϕ in a velocity field X can be expressed as

$$\frac{d}{dt}\phi = -X \cdot \nabla\phi. \tag{20}$$

Along γ in our flow, this velocity is given by the shape gradient $-\sigma^* n$ (Eqn. 16), yielding an evolution

$$\frac{d}{dt}\phi = -X \cdot n = \sigma^* n \cdot n = \sigma^*. \tag{21}$$

Away from γ, it is difficult to explicitly express the change in the signed distance ϕ induced by the motion of γ. However, to track the evolution of γ, we only need to accurately integrate ϕ in the immediate vicinity of γ. We therefore harmonically interpolate the scalar velocity $d\phi/dt$ (Eqn. 21) over the interior by solving a standard Laplace equation using σ^* as Dirichlet boundary conditions; after integration, ϕ is projected back onto a signed distance function (see Sec. 6.4). An attractive feature of this formulation is that it involves only ordinary scalar functions; we entirely avoid the need to represent tangent vector fields, as well as the curve normal n.

4.2 Length Regularization

For a signed distance function ($|\nabla\phi| = 1$), the curve shortening component of the flow is easily expressed via the well-known relationship $\Delta\phi = -\kappa_\phi$, where $\kappa_\phi : M \to \mathbb{R}$ denotes the curvature of the level sets of ϕ. Applying the advection equation (Eqn. 20) to the velocity field $X = -\kappa_\phi n$ then yields the evolution

$$\frac{d}{dt}\phi = -\Delta\phi, \tag{22}$$

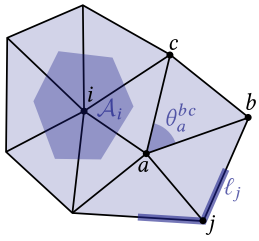
which is modulated by the length regularization parameter α_L. This formulation easily leads to stable implicit integration (Sec. 6.3).

5 DISCRETE FORMULATION

In principle one can use any numerical technique to discretize the cut flow; we consider an Eulerian discretization on triangle meshes using standard techniques from discrete differential geometry [Crane et al. 2013], as outlined below. For simplicity we first consider the case where γ is a single closed curve; the more general case of multiple regions is detailed in App. A. The final algorithm is summarized in Fig. 9.

Note that at no point do we explicitly compute a map to the plane, avoiding many of the usual pitfalls of mesh parameterization (such as concerns about local injectivity). Flattenings shown in figures are purely for visualization; any conformal flattening algorithm that guarantees minimal area distortion (such as CETM [Springborn et al. 2008] or BFF [Sawhney and Crane 2017]) could be used to produce these results from the given cut.

5.1 Discrete Surfaces



We discretize the surface M as a manifold triangle mesh $\mathsf{M} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$, possibly with boundary. We will use $\mathsf{B} \subset \mathsf{V}$ to denote the set of boundary vertices, and $\mathsf{I} := \mathsf{V} \backslash \mathsf{B}$ for the interior vertices. A sequence of k indices specifies a $(k-1)$-simplex; for instance, $ijk \in \mathsf{F}$ is a triangle with vertices i, j, and k. An important convention is that indices appearing on both sides of an equality implicitly restrict the terms in a sum; for instance, $a_i = \sum_{ijk} b_{ijk}$ denotes a sum over triangles containing vertex i. We use $\ell_{ij} \in \mathbb{R}_{>0}$ to denote the length of edge $ij \in \mathsf{E}$, and $\mathcal{A}_{ijk} \in \mathbb{R}_{>0}$ to denote the area of triangle ijk. For any vertex i, the area of the associated barycentric dual cell is one-third the area of all incident triangles: $\mathcal{A}_i := \frac{1}{3}\sum_{ijk} \mathcal{A}_{ijk}$, and for three consecutive boundary vertices $i, j, k \in B$, the barycentric dual length at j is $\ell_j := \frac{1}{2}(\ell_{ij} + \ell_{jk})$ (see inset figure). Finally, we will use θ_i^{jk} to denote the interior angle at vertex i of triangle ijk.

5.2 Discrete Curvature

Gaussian curvature K is most typically discretized as the deviation of the vertex angle sum $\Theta_i := \sum_{ijk} \theta_i^{jk}$ from the Euclidean angle sum of 2π, which corresponds to the integral of K over the vertex neighborhood. Dividing by the vertex area $\mathcal{A}_i$ then gives us the

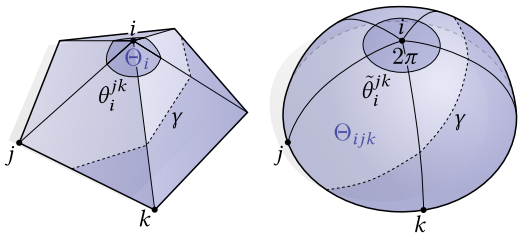


Fig. 7. If we discretize Gaussian curvature as a value Θ_i per vertex *(left)*, it always belongs entirely to one side of the cut γ. We therefore define a Gaussian curvature Θ_{ijk} per face *(right)*, which can be distributed proportionally to regions on both sides of the cut.

pointwise Gaussian curvature

$$\mathsf{K}_i := (2\pi - \Theta_i)/\mathcal{A}_i. \tag{23}$$

For our flow, this discretization causes undesirable artifacts: if curvature is concentrated at vertices, the curvature in a region will suddenly "jump" when the cut curve passes across a vertex. We therefore associate Gaussian curvature with *triangles*, in the spirit of Knöppel et al. [2013, Equation 13]. In particular, let

$$\tilde{\theta}_i^{jk} := 2\pi\theta_i^{jk}/\Theta_i,$$

so that the augmented angles $\tilde{\theta}$ sum to 2π around each vertex, effectively pushing all curvature into faces (see Fig. 7, *right*). The total curvature of a face $ijk \in \mathsf{F}$ is then just the deviation from the angle sum for a Euclidean triangle:

$$\Theta_{ijk} := \pi - \tilde{\theta}_i^{jk} - \tilde{\theta}_j^{ki} - \tilde{\theta}_k^{ij}. \tag{24}$$

Dividing by area gives the pointwise value $\mathsf{K}_{ijk} := \Theta_{ijk}/\mathcal{A}_{ijk}$. The total curvature of a partial face (obtained by multiplying K_{ijk} by the partial area) then varies continuously with respect to variations of the cut curve, helping to prevent numerical artifacts when solving the discrete Yamabe equation (Sec. 5.5).

5.3 Discrete Poisson Equation

We discretize the Laplace-Beltrami operator Δ via the usual postive-semidefinite *cotan matrix* $\mathsf{L} \in \mathbb{R}^{|\mathsf{V}|\times|\mathsf{V}|}$ with nonzero entries

$$\mathsf{L}_{ij} = \begin{cases} -\frac{1}{2}\sum_{ijk} \cot\theta_k^{ij}, & i \neq j, \\ -\sum_{p\neq i} \mathsf{L}_{pi}, & i = j. \end{cases} \tag{25}$$

For an inhomogeneous Laplacian Δ_a with vertex scale factors $a : \mathsf{V} \to \mathbb{R}$, we simply multiply the cotans by $(a_i + a_j)/2$ to yield a matrix L_a (diagonal entries are again given by row sums).

We consider two different mass matrices: a diagonal *vertex mass matrix* $\mathsf{W}_\mathsf{V} \in \mathbb{R}^{|\mathsf{V}|\times|\mathsf{V}|}$ with nonzero entries $(\mathsf{W}_\mathsf{V})_{ii} = \mathcal{A}_i$, and a rectangular *face mass matrix* $\mathsf{W}_\mathsf{F} \in \mathbb{R}^{|\mathsf{V}|\times|\mathsf{F}|}$ with nonzero entries $(\mathsf{W}_\mathsf{F})_{i,ijk} := \mathcal{A}_{ijk}/3$ for each face ijk containing a vertex i. These matrices map piecewise constant functions on dual cells and primal triangles (*resp.*) to their integrated values on dual cells.

To formulate boundary conditions, we partition matrices into blocks corresponding to interior vertices I and boundary vertices B. (On the cut mesh defined below, B includes both vertices on the cut and vertices along the domain boundary.) A Poisson equation $\Delta u = f$ with Neumann data $\partial u/\partial n = v$ along ∂M can then be expressed as

$$\begin{bmatrix} \mathsf{L}_{\mathsf{II}} & \mathsf{L}_{\mathsf{IB}} \\ \mathsf{L}_{\mathsf{IB}}^\mathsf{T} & \mathsf{L}_{\mathsf{BB}} \end{bmatrix} \begin{bmatrix} \mathsf{u}_\mathsf{I} \\ \mathsf{u}_\mathsf{B} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathsf{v} \end{bmatrix} = \mathsf{W}\mathsf{f}, \tag{26}$$

where for each boundary vertex $i \in \mathsf{B}$ the values $\mathsf{v}_i \in \mathbb{R}$ represent the integral of v over the corresponding dual boundary edge. The mass matrix W on the right-hand side will be either or W_V or W_F, depending on whether the source term is encoded by a column vector $\mathsf{f} \in \mathbb{R}^{|\mathsf{V}|}$ of values on vertices, or $\mathsf{f} \in \mathbb{R}^{|\mathsf{F}|}$ of values on faces.

5.3.1 Discrete Harmonic Interpolation. Any quantity defined at boundary vertices can be interpolated to interior vertices via *harmonic interpolation*, *i.e.*, by solving a discrete Laplace equation with Dirichlet boundary conditions. Explicitly, if $\mathsf{u}_\mathsf{B} \in \mathbb{R}^{|\mathsf{B}|}$ are the boundary values, then the interior values $\mathsf{u}_\mathsf{I} \in \mathbb{R}^{|\mathsf{I}|}$ are obtained via

$$\mathsf{L}_{\mathsf{II}}\mathsf{u}_\mathsf{I} = -\mathsf{L}_{\mathsf{IB}}\mathsf{u}_\mathsf{B}. \tag{27}$$

5.4 Cut Mesh

Our discrete cut curve is represented implicitly via a piecewise linear signed distance function ϕ interpolating values $\phi_i \in \mathbb{R}$ at vertices $i \in \mathsf{V}$. The zero set γ of this function is a piecewise linear curve, which is easily extracted by identifying zeros on the edges of each triangle (see App. A.2 for discussion of multiple regions). Explicitly cutting along γ and triangulating the resulting faces yields two *submeshes* M^+ and M^- corresponding to the regions where ϕ is nonnegative and nonpositive, *resp.*; both submeshes include all vertices inserted along the cut. Since each vertex of the original mesh is contained in either M^+ or M^-, values computed on either submesh can be easily copied back to M. Steps (2)–(5) of the algorithm (as enumerated in Fig. 9) are evaluated on both M^+ and M^-; all other steps can be evaluated on the original mesh M.

Note that γ does not necessarily cut the surface into disk-like regions—in general, one might also have nonsimply connected regions with holes or handles. To obtain the final flattening, such regions can be further cut into disks at the conclusion of the flow (*e.g.*, using the method of Erickson and Whittlesey [2005]).

5.5 Discrete Yamabe Equation

To discretize the distortion-minimizing term in our flow, we need the boundary derivatives $\partial u/\partial n$ of the solution to the Yamabe equation (Eqn. 3). Remembering that we have zero-Dirichlet boundary conditions ($\mathsf{u}_\mathsf{B} = 0$), we can use the discrete Poisson equation (Eqn. 26) to obtain a linear system for the unknown values of $\mathsf{u}_\mathsf{I} \in \mathbb{R}^{|\mathsf{I}|}$ on the interior, and unknown normal derivatives $\mathsf{v} \in \mathbb{R}^{|\mathsf{B}|}$ on the boundary:

$$\begin{bmatrix} \mathsf{L}_{\mathsf{II}} & 0 \\ \mathsf{L}_{\mathsf{IB}}^\mathsf{T} & I \end{bmatrix} \begin{bmatrix} \mathsf{u}_\mathsf{I} \\ \mathsf{v} \end{bmatrix} = \mathsf{W}_\mathsf{F}\mathsf{K}. \tag{28}$$

Here I is the $|\mathsf{B}| \times |\mathsf{B}|$ identity matrix, and $\mathsf{K} \in \mathbb{R}^{|\mathsf{F}|}$ contains the per-face Gaussian curvature values defined following Eqn. 24. Since L_{II} is invertible, we can first solve a smaller, positive-definite system for u_I, and then simply evaluate v using the known values of u_I.

5.6 Discrete Adjoint Equation

Except when using the Dirichlet energy, we also need the boundary derivatives $\partial p/\partial n$, which can be obtained by solving an equation

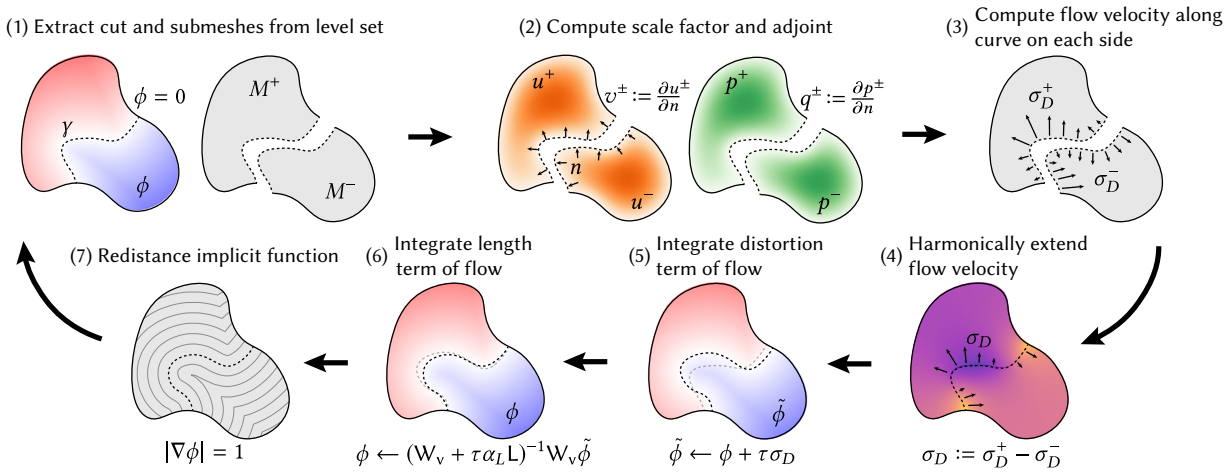


Fig. 8. Overview of the steps to evaluate our flow.

Input: A manifold triangle mesh $\mathsf{M} = (\mathsf{V}, \mathsf{E}, \mathsf{F})$ and a function $\phi^0 : \mathsf{V} \to \mathbb{R}$ whose zero set specifies the initial cut.
Output: A locally optimal cut γ^* as a piecewise linear path on M.

- (Optional:) Remesh to desired output resolution (Sec. 6.1).
- Until convergence:
  (1) Extract cut γ^t and submeshes $\mathsf{M}^+, \mathsf{M}^-$ from ϕ^{t-1} (Sec. 5.4).
  (2) Compute values of v, q along γ^t (Sections 5.5, 5.6).
  (3) Evaluate σ_D (Sec. 6.2).
  (4) Harmonically extend σ_D (Sec. 5.3.1).
  (5) Integrate distortion term: $\widetilde{\phi}^t \leftarrow \widetilde{\phi}^t + \tau\sigma_D$ (Sec. 6.2).
  (6) Integrate length term: $(\mathsf{W_V} + \tau\alpha_L \mathsf{L})\phi^t = \mathsf{W_V}\widetilde{\phi}^t$ (Sec. 6.3).
  (7) Redistance ϕ^t with respect to γ^{t+1} (Sec. 6.4).
- Extract cut γ^* from ϕ^t (Sec. 5.4).

Fig. 9. The key steps used to integrate our conformal cut flow. Steps (2)–(5) are performed for both M^+ and M^-; all other steps are performed on M.

nearly identical to Eqn. 28. In particular, we replace $\mathsf{u_I}$ with a vector $\mathsf{p_I} \in \mathbb{R}^{|\mathsf{I}|}$ representing interior values of p; v with a vector $\mathsf{q} \in \mathbb{R}^{|\mathsf{B}|}$ representing (integrated) normal derivatives $\partial p/\partial n$; and use a different source term, which depends on the choice of energy.

Source Term. For a pointwise energy $b(u)$, we get the source term by just evaluating $b'(\mathsf{u}_i)$ at each vertex $i \in \mathsf{V}$. For the rescaled Dirichlet energy, one easily obtains the right hand side

$$-2\mathsf{W_V}\left(\mathsf{L_a u} + \begin{bmatrix} 0 \\ \mathsf{y} \end{bmatrix}\right),$$

where $\mathsf{y} \in \mathbb{R}^{|\mathsf{B}|}$ has entries $\mathsf{y}_i = \mathsf{a}_i \mathsf{v}_i$, and $\mathsf{a}_i = \mathsf{r}_i^3$ (as discussed in Sec. 3.5.1). To obtain a local feature size r_i at each vertex, we let

$$\mathsf{r}_i^0 := 1/\sqrt{|\mathsf{K}_i| + r_\varepsilon^{-1}},$$

where K_i is the curvature at vertex i (Eqn. 23), and the term $r_\varepsilon > 0$ avoids degeneracy in flat regions. Since pointwise estimates of curvature are typically quite noisy (even for nice triangulations), we smooth r^0 using a single step of heat flow via backward Euler:

$$\left(\mathsf{W_V} + c\mathsf{L}\right)\mathsf{r} = \mathsf{W_V}\mathsf{r}^0.$$

This system is positive definite, and the parameter $c > 0$ controls the amount of smoothing (we use $c = 0.02$ in all experiments).

6 ALGORITHM

To advect the signed distance function ϕ we adopt a standard operator splitting approach where the length term σ_L and distortion terms σ_D (Eqn. 15) are separately integrated using an explicit and implicit strategy (*resp.*); the signed distance constraint $|\nabla\phi| = 1$ is enforced via projection. We use a fixed time step $\tau > 0$, though since the flow minimizes an energy (Eqn. 5) one could easily apply a more sophisticated line search strategy.

6.1 Preprocessing

In an Eulerian discretization, the output resolution of the cut is of course limited by resolution of the mesh—one can therefore optionally remesh the input surface M. For highly oscillatory cuts, we adapt the edge length to the magnitude $|K|$ of the Gaussian curvature, since we expect more oscillations in highly curved regions (Fig. 6). Any standard remeshing strategy can be used here (*e.g.*, [Dunyach et al. 2013]); we simply perform local subdivision until the curvature K_{ijk} in each face falls below a user-specified bound.

6.2 Distortion Minimization

To integrate the distortion term σ_D, we first compute the normal derivatives v and (possibly) q for each submesh $\mathsf{M}^+, \mathsf{M}^-$ (as described in Sections 5.6 and 5.5). These derivatives can be used to evaluate any of the distortion gradients in Table 1—for example, in the case of Dirichlet energy we compute values $(\sigma_D^\pm)_i = (\mathsf{v}_i^\pm/\ell_i)^2$ at each vertex i for both sides of the cut. Division by the dual boundary length ℓ_i is needed to convert the integrated values of v_i to pointwise values (and likewise for q_i). The overall distortion term is then given by $(\sigma_D)_i = (\sigma_D^+)_i - (\sigma_D^-)_i$.

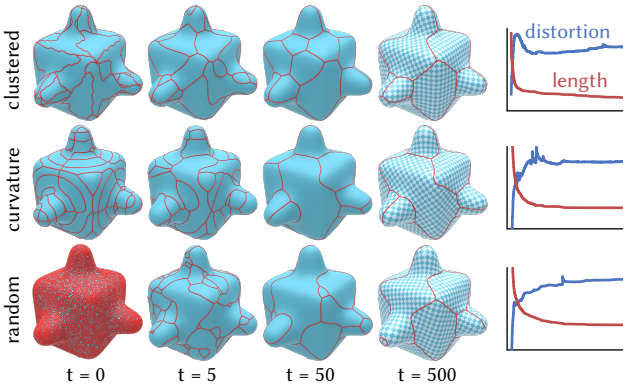


Fig. 10. Although our flow finds only local minima, different initializations can still lead to very similar results. Here we initialize using normal clustering *(top)*, level sets of mean curvature *(middle)*, and random values *(bottom)*. In each case, the initial cut is driven to a much shorter, low-distortion cut. Plots show rescaled Dirichlet distortion energy and total cut length.

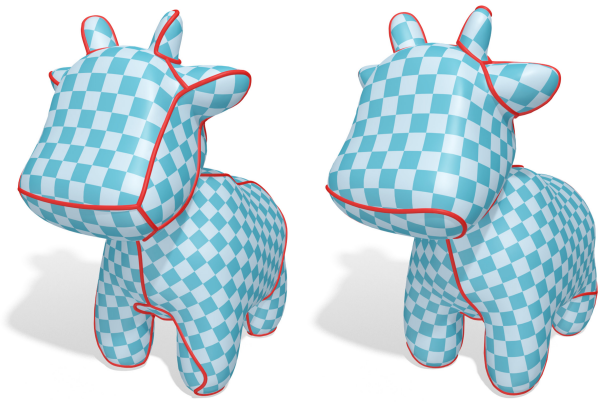


Fig. 11. Adding a penalty in visible regions causes the flow to automatically hide the cut as much as possible, while avoiding excessive distortion. *Left:* no penalty. *Right:* with penalty.

To mitigate discretization artifacts, we apply 1D Laplacian smoothing at vertices of small sliver triangles (aspect ratio less than a fixed constant $\rho_0 := 5$); values at other vertices remain unchanged. The smoothed function is then harmonically interpolated, *à la* Sec. 5.6. Finally, to perform advection, we take a single explicit Euler step

$$\phi_i^{t+1} = \phi_i^t + \tau(\sigma_D)_i.$$

6.3 Length Regularization

One way to numerically integrate the length term σ_L would be to numerically estimate the curvature κ of the cut curve and add this term to the overall flow velocity—however, this strategy is notoriously unstable [Wu and Tai 2010]. We instead apply the unconditionally stable backward Euler scheme to Eqn. 22, yielding the implicit update

$$(\mathsf{I} + \tau\alpha_L \mathsf{L})\phi^{t+1} = \mathsf{W}_\mathsf{V}\phi^t. \tag{29}$$

6.4 Redistancing

Following advection (Sections 6.3 and 6.2), ϕ may no longer be a signed distance function. We therefore update ϕ by computing the distance to the new cut curve. The distance at vertices adjacent to the cut (*i.e.*, connected by an edge) is not updated to avoid introducing a numerical drift of the curve; at all other vertices one can apply any numerical method for computing geodesic distance—we used the fast marching method [Kimmel and Sethian 1998].

7 EXTENSIONS AND APPLICATIONS

To demonstrate the robustness and utility of our flow, we consider several aspirational examples and possible extension from geometry processing and computational design. We implemented the flow in C++, using *SuiteSparse* [Chen et al. 2008] for numerical linear algebra. Computational cost is dominated by solving a small number of linear systems; since many factorizations can be re-used and the submeshes partition the domain, the total cost per time step is roughly the same as factoring a single $|\mathsf{V}|\times|\mathsf{V}|$ Laplace matrix. Fig. 1, 3, 5, 6, 12, 13, 17 and 18, used the standard Dirichlet energy; Figures 10, 11 and 16 used the rescaled Dirichlet energy to handle more variable curvature, and Figures 2, 14 and 15 used the Hencky energy, which is better suited to fabrication. Fig. 10 shows the flow has little dependence on initialization; for all other multi-region examples, we initialized the cut via greedy clustering of vertex normals. In Fig. 3 and Fig. 2 we imposed explicit symmetry constraints by constraining symmetric vertices to identical values of ϕ; for meshes that do not exhibit per-vertex symmetry, one could likewise use a functional notion of symmetry [Ovsjanikov et al. 2013].

7.1 Penalty Functions

In addition to penalizing length and distortion, we can use any user-defined penalty function $\eta : M \to \mathbb{R}_{\geq 0}$ to drive the cut toward or away from application-relevant regions (*e.g.*, in regions where seams might reduce structural stability). To do so, we add a term

$$E_\eta(\gamma) := \int_\gamma \eta(x)\, ds \tag{30}$$

to the overall objective E (Eqn. 5). Since this energy is effectively a reweighted version of the curve length (Eqn. 4), its shape derivative is just $\eta\kappa$. Hence, we need only replace the constant length penalty α_L with the variable weight $\alpha_L + \eta$, *i.e.*, in Eqn. 29 we replace α_L with a diagonal matrix with nonzeros $\mathsf{D}_{ii} := \alpha_L + \eta_i$.

As one concrete example, we can discourage cuts from appearing where they are easily visible. In particular, we let $\eta_i := \nu_i n_i \cdot E_i$; here $\nu : \mathsf{V} \to [0, 1]$ is a smoothed binary function indicating visibility, n_i is the vertex normal, and E_i is the unit vector from vertex i toward the camera location. Fig. 11 shows one example.

7.2 Computational Design

The fact that our cuts are highly regular and induce extremely low area distortion makes them useful for designing patterns that will be cut from flat, nearly inextensible materials such as cloth, sheet metal, paper, or plywood, which are then sewn, welded, or otherwise joined together. During this process, the residual area distortion roughly predicts the strain experienced by a slightly extensible material (such as leather), or alternatively, the approximation error exhibited

Fig. 13. Our flow can be used in conjunction with other design tools, or to optimize existing designs. Here we use a classic volleyball pattern *(left)* to define our initial cut, and flow to a new design *(right)* that has both smaller scale distortion (hence more uniform material stress) *and* smaller cut length.

by a perfectly inextensible material (such as paper). Optionally, we can exchange the Dirichlet energy E_D (Eqn. 2) for the more physically meaningful Hencky energy, as discussed in Sec. 3.5.2. Unlike Dirichlet energy, no rescaling is needed since Hencky energy has consistent flow rates even on surfaces of highly varying curvature.

Here we briefly explore the application of our method to computational design problems that demand judicious placement of cuts. First, we consider examples from the world of sports, where seam placement and mechanical stress play an important role in game dynamics [Hong and Asai 2014]. Fig. 1 shows how the standard design for a tennis ball naturally emerges from our flow. Here we initialize with a circle around the equator and do not explicitly enforce symmetry; weakening the length penalty yields alternative designs with lower distortion. Fig. 12 shows a similar result where explicit constraints are used to enforce icosahedral symmetry. Fig. 13 illustrates how our flow can be used to improve an existing design, initializing with the standard pattern for a volleyball and flowing to a pattern with both lower distortion *and* shorter cut length. In Fig. 2 we use Hencky distortion and a bilateral symmetry constraint to design low-distortion panels for leather chairs. Finally, we can minimize distortion on just one side of the cut to optimize the shape of a patch on a surface—for instance, in Fig. 14 we design bandages custom tailored to a particular patient physiology, which reduces stretching and helps dressings remain on the body [Fletcher 2005]. Here, a small patch is initialized around a wound; an additional area term encourages it to grow outward. Such patches could be readily cut from a flat sheet of bandage material.

Fig. 12. A symmetric cut on an inconvenient tennis ball.

Developable Approximation. Rather than viewing scale distortion as the strain in a pliable material such as rubber, we can view it as the geometric approximation error induced by fabrication via

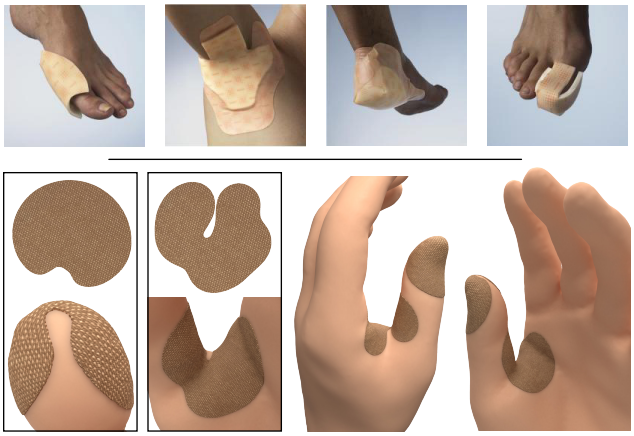


Fig. 14. To design a small patch on the surface, we can simply penalize distortion on only one side of the cut. Here for instance, we design bandages adapted to a particular patient or wound (*bottom*), mimicking real medical dressings adapted to particular body parts (*top*, courtesy of [Fletcher 2005]).

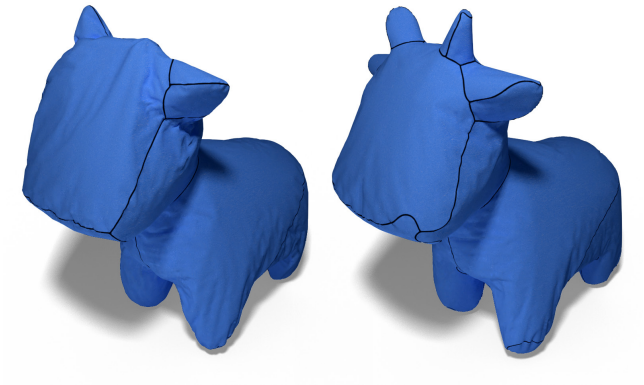


Fig. 15. To verify that we obtain near-isometric flattenings, we reconstruct a surface using the edge lengths from the 2D domain. Wrinkling and crumpling indicate excessive scale distortion *(left)*; similar behavior will occur if such a piece is fabricated from developable material. Even a moderate reduction in distortion can significantly improve manufacturability *(right)*.

inextensible material such as sheet metal. Such *piecewise developable* designs are typically obtained via user interaction [Tang et al. 2016; Rabinovich et al. 2018] or deformation of the input geometry itself [Wang and Tang 2004; Kilian et al. 2008]; we instead simply run our flow until the resulting flattening is near-isometric. Fig. 15, *right* verifies that even fairly short cuts can yield developable patches that nicely approximate the original shape. (To produce this visualization we simply minimize the deviation of the 3D edge lengths from the flattened 2D edge lengths to obtain a near-isometric embedding; a small bending term avoids severe crumpling.) Fig. 6, *right* shows a nearly perfect developable approximation obtained via a long winding cut.

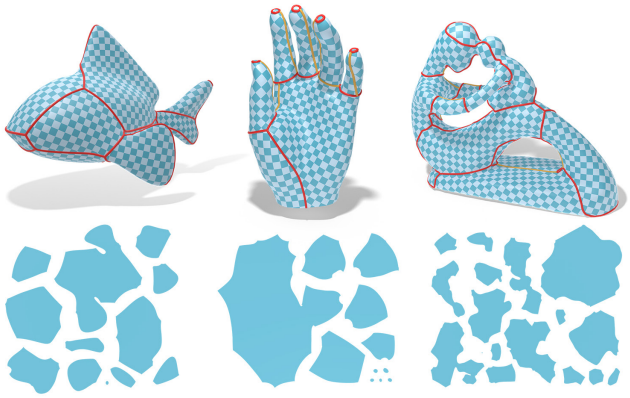


Fig. 16. Though not the primary focus of this work, our flow yields a low-distortion texture atlas when strong length regularization is used. *Top:* uniform checkerboards indicate near-isometric flattening. *Bottom:* flattened patches. All three examples used identical parameters; feature alignment emerges naturally, even without an explicit alignment term. (Orange indicates cuts added as a post-process, in order to obtain disk topology.)

7.3 Texture Mapping

The problem of cutting a surface into patches for texture mapping has been extensively studied, with many highly specialized algorithms (see Sec. 2). Though our main interest is exploring cutting problems beyond traditional texture mapping, we can nonetheless generate a low-distortion texture atlas by running our flow with strong length regularization (Fig. 16). Our method could also be used to optimize the length/distortion trade off in any existing atlas. One interesting feature of our flow is that it can develop small loops that resemble cone singularities (*e.g.*, the fingertips in Fig. 16, *middle*); this behavior is atypical of traditional methods, which look for either a cut or for cones, but not both in conjunction.

7.4 Space-Filling Curves

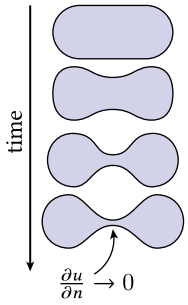
As the length penalty goes to zero, our flow tends toward space filling curves. The general problem of finding space-filling curves or *ropes* (*i.e.*, curves of finite thickness) on surfaces has been analyzed for simple geometries like the sphere [Gerlach and von der Mosel 2011], and investigated via manual design [Delp and Thurston 2011], but few computational algorithms are known. Existing methods are mainly targeted at tasks like error diffusion [Alexa and Kyprianidis 2015] or generating coherent memory layouts [Vo et al. 2012; Isenburg and Lindstrom 2005], which do not require geometric regularity or even global continuity. Pedersen and Singh [2006] generate smooth space-filling curves via regularized Brownian motion, though it is not clear how to incorporate nonlocal terms (such as distortion) into such a framework. Some simple modifications to our flow yield smooth space-filling curves with user-specified density, or which incorporate nonlocal terms.

Fig. 17. A decorative space-filling curve on a ceramic vase.

Uniform Density Curves. In our flow, recall that the scale distortion u is governed by the Yamabe Equation $\Delta u = -K$ (Eqn. 3); the density of cuts will hence be roughly proportional to $|K|$. Replacing K with any fixed constant instead yields curves of uniform density, which are still adapted to the shape since Δ depends on the original metric. In Fig. 17, we initialize γ with a small circle, then gradually decrease the length penalty as the curve evolves. To avoid spurious topological merging/splitting events (as discussed in Sec. 8), we pick a time step that ensures the cut never moves by more than a constant factor of the smallest edge length.

Thermal Element Design. The Poisson equation $\Delta u = -K$ relating scaling and curvature can also of course be interpreted as the governing equation for a steady-state temperature distribution u. From this perspective, our method can be used to design a thermal element which is short but effectively cools or heats a given surface. Fig. 18 shows a toy example where K is set to a larger value around the engine block, where more rapid cooling is required. Such an approach might facilitate computational design of heat pipes for mechanical and electronic components [Maydanik 2005].

8 LIMITATIONS, FUTURE WORK, AND CONCLUSION



Topological Changes. In the smooth setting our conformal cut flow will not cause two regions to merge, nor a single region to split in two: in such situations, the thin "neck" between regions is easy to flatten isometrically, hence the velocity $(\frac{\partial u}{\partial n})^2$ would tend toward zero during the event. However, since we adopt an Eulerian discretization, such pinch-offs can still occur. Though adaptive time stepping can help mitigate such artifacts (as mentioned in Sec. 7.4), a Lagrangian formulation might provide stronger guarantees. On the other hand, even our smooth flow is currently unaware of potentially desirable changes in the curve topology, such as adding a new component to the cut that models a cone singularity not captured by an existing loop. These changes could be considered via a *topological derivative* [Sokolowski and Zochowski 1999; He et al. 2007].

Discretization. In this work we explicitly extract cut meshes on which we compute necessary quantities using traditional techniques. However, the *extended finite element method* could be used to evaluate these quantities on cut domains without an explicit mesh [Sukumar et al. 2001]. We are optimistic that such techniques will yield more efficient and robust numerical integration of the flow.

Optimality. As discussed in Sec. 2, the problem of finding globally optimal cuts is generally intractable. Accordingly, our approach yields only locally optimal solutions: the energies we consider are not convex, and thus the result of our flow is dependent on the initialization. However, as we have seen (*e.g.*, in Fig. 10) even local

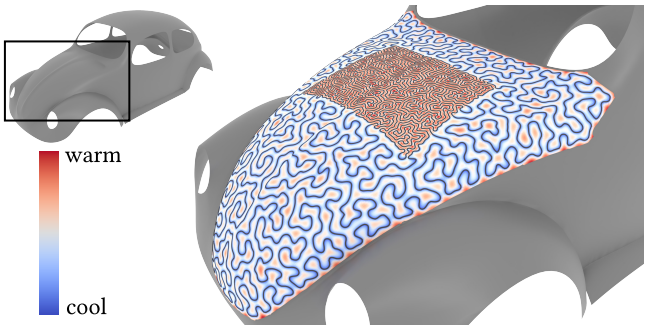


Fig. 18. We can also find space-filling curves of variable density. Here we adapt our flow to design a cooling element on the hood of a car, where an engine block generates 10x more heat than the rest of the hood. Colors indicate steady-state temperature.

minima can look quite similar for very different initializations. Likewise, examples initialized starting with nothing more than a simple circle, or greedy normal clustering, show that good results can be achieved without careful initialization.

Looking forward, we are optimistic that the variational viewpoint will provide new insights into the problem of surface cutting, and open the door to new applications not previously considered.

ACKNOWLEDGEMENTS

Thanks to the anonymous reviewers, Christian Santangelo for early discussions about conformal cutting, and Dejan Slepčev for pointers to nonlocal isoperimetric problems. This work was sponsored in part by NSF Award 1717320 and Graduate Research Fellowship Program Grants DGE 1252522 and DGE 1745016, and gifts from Autodesk Research and Adobe Research.

REFERENCES

Giovanni Alberti, Rustum Choksi, and Felix Otto. 2009. Uniform energy distribution for an isoperimetric problem with long-range interactions. *Journal of the American Mathematical Society* (2009).

Marc Alexa and Jan Eric Kyprianidis. 2015. Error diffusion on meshes. *Computers & Graphics* (2015).

Grégoire Allaire, François Jouve, and Anca-Maria Toader. 2004. Structural optimization using sensitivity analysis and a level-set method. *Journal of computational physics* (2004).

Thierry Aubin. 2013. *Some nonlinear problems in Riemannian geometry.* Springer Science & Business Media.

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

Jean Céa. 1986. Conception optimale ou identification de formes, calcul rapide de la dérivée directionnelle de la fonction coût. *RAIRO-Modélisation mathématique et analyse numérique* (1986).

Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. 2008. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. (2008).

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

Philippe Decaudin, Dan Julius, Jamie Wither, Laurence Boissieux, Alla Sheffer, and Marie-Paule Cani. 2006. Virtual garments: A fully geometric approach for clothing design. In *Computer Graphics Forum.*

Kelly Delp and Bill Thurston. 2011. Playing with Surfaces: Spheres, Monkey Pants, and Zippergons. In *Proceedings of the 14th Annual Bridges Conference.*

Marion Dunyach, David Vanderhaeghe, LoÃŕc Barthe, and Mario Botsch. 2013. Adaptive Remeshing for Real-Time Mesh Deformation. In *Eurographics 2013 - Short Papers.*

Jeff Erickson and Sariel Har-Peled. 2004. Optimally cutting a surface into a disk. *Discrete & Computational Geometry* (2004).

Jeff Erickson and Kim Whittlesey. 2005. Greedy optimal homotopy and homology generators. In *Proceedings of the sixteenth annual ACM-SIAM symposium on discrete algorithms.*

Jacqui Fletcher. 2005. Dressings: cutting and application guide. *World Wide Wounds.* (2005).

Michael Gage and Richard S Hamilton. 1986. The heat equation shrinking convex plane curves. *Journal of Differential Geometry* (1986).

Henryk Gerlach and Heiko von der Mosel. 2011. On sphere-filling ropes. *Amer. Math. Monthly* (2011).

Craig Gotsman. 2003. On graph partitioning, spectral analysis, and digital mesh processing. In *Shape Modeling International, 2003.*

Matthew A Grayson. 1987. The heat equation shrinks embedded plane curves to round points. *Journal of Differential geometry* (1987).

Lin He, Chiu-Yen Kao, and Stanley Osher. 2007. Incorporating topological derivatives into shape derivatives based level set methods. *J. Comput. Phys.* (2007).

Heinrich Hencky. 1928. Über die Form des Elastizitätsgesetzes bei ideal elastischen Stoffen. *Zeitschrift für technische Physik* (1928).

Sungchan Hong and Takeshi Asai. 2014. Effect of panel shape of soccer ball on its flight characteristics. *Scientific reports* (2014).

Martin Isenburg and Peter Lindstrom. 2005. Streaming meshes. In *Visualization, 2005. VIS 05. IEEE.*

Dan Julius, Vladislav Kraevoy, and Alla Sheffer. 2005. D-Charts: Quasi-Developable Mesh Segmentation. *Comput. Graph. Forum* (2005).

Liliya Kharevych, Boris Springborn, and Peter Schröder. 2006. Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics (TOG)* (2006).

Martin Kilian, Simon Flöry, Zhonggui Chen, Niloy J Mitra, Alla Sheffer, and Helmut Pottmann. 2008. Curved folding. In *ACM Transactions on Graphics (TOG).* ACM.

R Kimmel and J A Sethian. 1998. Computing geodesic paths on manifolds. *Proc. Natl. Acad. Sci. U. S. A.* (1998).

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Globally optimal direction fields. *ACM Trans. Graph.* (2013).

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least squares conformal maps for automatic texture atlas generation. In *ACM transactions on graphics (TOG).*

Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. 2006. Multiple interacting liquids. In *ACM SIGGRAPH 2006 Papers.*

Yu F Maydanik. 2005. Loop heat pipes. *Applied Thermal Engineering* (2005).

Jun Mitani and Hiromasa Suzuki. 2004. Making Papercraft Toys from Meshes Using Strip-based Approximate Unfolding. *ACM Trans. Graph.* (2004).

Maks Ovsjanikov, Quentin Mérigot, Viorica Patraucean, and Leonidas Guibas. 2013. Shape matching via quotient spaces. In *Computer Graphics Forum.*

Hans Pedersen and Karan Singh. 2006. Organic labyrinths and mazes. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering.*

Roi Poranne, Marco Tarini, Sandro Huber, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Autocuts: Simultaneous Distortion and Cut Optimization for UV Mapping. *ACM Transactions on Graphics* (2017).

Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2018. Discrete Geodesic Nets for Modeling Developable Surfaces. *ACM Trans. Graph.* (2018).

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

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

C. Schüller, R. Poranne, and O. Sorkine-Hornung. 2017. Shape Representation by Zippable Ribbons. *ArXiv e-prints* (2017).

Alla Sheffer and John C Hart. 2002. Seamster: inconspicuous low-distortion texture seam layout. In *Proceedings of the conference on visualization '02.*

Jan Sokolowski and Antoni Zochowski. 1999. On the topological derivative in shape optimization. *SIAM journal on control and optimization* (1999).

Yousuf Soliman, Dejan Slepčev, and Keenan Crane. 2018. Optimal Cone Singularities for Conformal Flattening. *ACM Trans. Graph.* (2018).

Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. 2002. Bounded-distortion piecewise mesh parameterization. In *Proceedings of the conference on Visualization'02.*

Boris Springborn, Peter Schröder, and Ulrich Pinkall. 2008. Conformal equivalence of triangle meshes. In *ACM SIGGRAPH 2008 Papers.*

Natarajan Sukumar, David L Chopp, Nicolas Moës, and Ted Belytschko. 2001. Modeling holes and inclusions by level sets in the extended finite-element method. *Computer methods in applied mechanics and engineering* (2001).

Chengcheng Tang, Pengbo Bo, Johannes Wallner, and Helmut Pottmann. 2016. Interactive design of developable surfaces. *ACM Transactions on Graphics (TOG)* (2016).

Huy T Vo, Claudio T Silva, Luiz F Scheidegger, and Valerio Pascucci. 2012. Simple and efficient mesh layout with space-filling curves. *Journal of Graphics Tools* (2012).

Charlie CL Wang and Kai Tang. 2004. Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches. *The Visual Computer* (2004).

Chunlin Wu and Xuecheng Tai. 2010. A level set formulation of geodesic curvature flow on simplicial surfaces. *IEEE Trans. Vis. Comput. Graph.* (2010).

Hitoshi Yamauchi, Stefan Gumhold, Rhaleb Zayer, and Hans-Peter Seidel. 2005. Mesh segmentation driven by Gaussian curvature. *The Visual Computer* (2005).

Kun Zhou, John Synder, Baining Guo, and Heung-Yeung Shum. 2004. Iso-charts: stretch-driven mesh parameterization using spectral analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing.*

A MULTIPLE REGIONS

For simplicity of exposition we initially described the flow on a surface partitioned into two regions, but our flow applies to general configurations of N regions, as seen in Figs. 2, 6, 11, 13 and 16; here we provide key implementation details. The shape derivative of the Yamabe equation (Sec. 3.3) applies to the general case without modification, though one must be careful to maintain a consistent orientation for the normal n along each segment (and corresponding choice of sign in Eqn. 16).

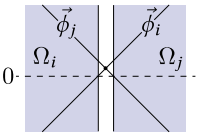
To represent more general curve networks via level sets (Sec. 4), we essentially follow the treatment of Losasso et al. [2006]. Consider a partition $\mathcal{P} \coloneqq \{\Omega_1, \ldots, \Omega_N\}$ of the surface into N disjoint regions (not necessarily simply connected). The curve network γ is then the union of the region boundaries:

$$\gamma = \bigcup_k \partial\Omega_k.$$

The regions themselves are expressed as a vector-valued function $\vec{\phi} : M \to \mathbb{R}^N$, where each $\vec{\phi}_k$ encodes the signed distance to the boundary of Ω_k; by convention the sign of $\vec{\phi}_k$ is negative inside the region. At any point $p \in M$, the minimum entry of $\vec{\phi}(p)$ is then the distance to the nearest point on γ, and γ is therefore the set of points where any component of $\vec{\phi}$ is zero. Since these regions need not be simply connected, one can represent complex cuts as the boundary of a small number of regions (consider the map coloring problem); we find that $N = 10$ is more than sufficient in practice. To evaluate the implicit evolution equation (Eqn. 21) one now simply needs to compute σ_D on either side of the cut, then solve a separate interpolation problem for each $\vec{\phi}_k$, with Dirichlet boundary conditions imposed only at points where $\vec{\phi}_k = 0$.

A.1 Discretization

In the discrete setting, the signed distance functions $\vec{\phi}$ are represented via N piecewise linear functions interpolating values $\phi_i \in \mathbb{R}^N$ at vertices $i \in \mathsf{V}$. Due to discretization error, the zero set of $\vec{\phi}$ no longer provides a reliable definition for the cut network γ: two components of $\vec{\phi}$ may have zeros at slightly different locations (as shown in the inset). We instead define γ as the set of points where $\vec{\phi}$ has more than one component of smallest magnitude. (In the smooth setting this set coincides with the zero level set, but in the discrete setting it provides a more robust definition—see [Losasso et al. 2006, Section 3] for further discussion.) The resulting set γ is still a network of piecewise linear curves, though possibly with multiple segments inside a single triangle.



A.2 Cut Mesh Extraction

Consider an edge from vertex a to vertex b, parameterized by a value $t \in [0, 1]$; let $i \coloneqq \operatorname{argmin}_k \phi_a^k$ and $j \coloneqq \operatorname{argmin}_k \phi_b^k$ be the regions associated with the two endpoints. If $i \neq j$, then the cut crosses the edge at the point where the identity of the most negative entry changes, given by the parameter value

$$t_c = \frac{\phi_a^i - \phi_a^j}{\phi_a^i - \phi_a^j + \phi_b^j - \phi_b^i} \in [0, 1]. \tag{31}$$

In this case we insert a new vertex at the location of the crossing. If all three edges of a triangle contain a crossing, then the triangle contains a triple point, and we insert a vertex at the centroid of the crossings. This strategy ensures that there is at most a single cut through each edge, and a single triple point within each face.

A.3 Numerical Integration

Numerical integration of multiple regions is largely the same as in the two-region case. Poisson problems are solved independently on each submesh $\widehat{M}_k$. The harmonic extension of the speed function σ_D is slightly more involved, as $\dot{\phi}_k$ is known only at the cut segments that bound $\widehat{M}_k$, yet must be interpolated to the entire surface. In principle this increases the computational cost of each iteration, requiring N interpolation problems over the entire surface with a distinct Laplace matrix factorization for each. In practice, however, these factorizations are not needed: since an accurate implicit function is needed only in a narrow band around the cut, we find it sufficient to substitute $\dot{\phi}_j \coloneqq -\dot{\phi}_k$ for unknown values of $\dot{\phi}_j$ along the boundary of region k, then solve per-submesh interpolation problems as in the two-region case. With this strategy we need only one $|\mathsf{V}| \times |\mathsf{V}|$ Laplace factorization per iteration; all multi-region examples above use this scheme. Length regularization and redistancing are applied independently for each distance function. When redistancing vertices adjacent to the cut, we use the projection of Losasso et al. [2006, Section 3.1] to avoid numerical drift and ensure exactly one entry of $\vec{\phi}$ is negative at each vertex (corresponding to the region containing that vertex).