Developability of Triangle Meshes

ODED STEIN and EITAN GRINSPUN, Columbia University
KEENAN CRANE, Carnegie Mellon University

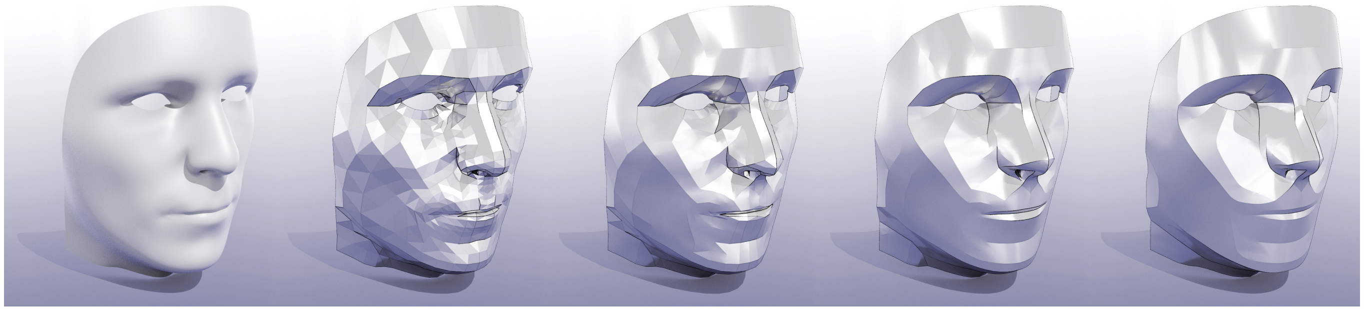


Fig. 1. By encouraging discrete developability, a given mesh evolves toward a shape comprised of flattenable pieces separated by highly regular seam curves.

Developable surfaces are those that can be made by smoothly bending flat pieces without stretching or shearing. We introduce a definition of developability for triangle meshes which exactly captures two key properties of smooth developable surfaces, namely flattenability and presence of straight ruling lines. This definition provides a starting point for algorithms in developable surface modeling—we consider a variational approach that drives a given mesh toward developable pieces separated by regular seam curves. Computation amounts to gradient descent on an energy with support in the vertex star, without the need to explicitly cluster patches or identify seams. We briefly explore applications to developable design and manufacturing.

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

Additional Key Words and Phrases: developable surface modeling, discrete differential geometry, digital geometry processing

ACM Reference Format:
Oded Stein, Eitan Grinspun, and Keenan Crane. 2018. Developability of Triangle Meshes. *ACM Trans. Graph.* 37, 4, Article 77 (August 2018), 14 pages. https://doi.org/10.1145/3197517.3201303

1 INTRODUCTION

Fabrication from developable pieces provides an enticing paradigm for manufacturing: flat sheet materials such as plywood or sheet metal are easy to cut, ship, and store; surfaces comprised of developable pieces also reduce cost and improve quality in computer controlled milling [Harik et al. 2013]. To date, however, there are few tools for automatic conversion of curved surfaces into developable pieces—most industrial applications still rely on manual interaction and designer expertise [Chang 2015].

The goal of this paper is to develop mathematical and computational foundations for developability in the simplicial setting, and show how this perspective inspires new approaches to developable design. Our starting point is a new definition of developability for triangle meshes (Section 3). This definition is motivated by the behavior of developable surfaces that are twice differentiable rather than those that are merely continuous: instead of just asking for flattenability, we also seek a definition that naturally leads to well-defined *ruling lines*. Moreover, unlike existing notions of discrete developability, it applies to general triangulated surfaces, with no special conditions on combinatorics.

These features make our definition a natural starting point for algorithms that seek to design developable surfaces. In this paper, we investigate a global variational approach that encourages the discrete developability of each vertex (Section 4.1). An interesting observation is that this local optimization naturally tends toward surfaces that are *piecewise* developable: in practice, curvature concentrates onto a sparse collection of seam curves (Section 4.1.3) which are themselves highly regular (Section 4.2).

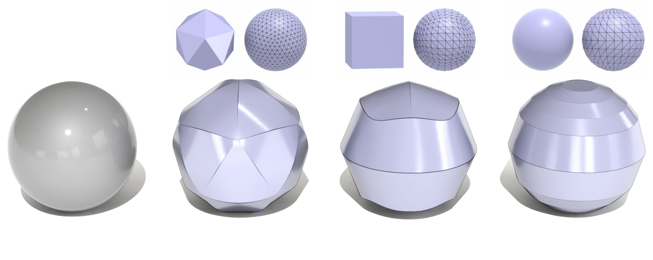


Fig. 2. How should one approximate a perfect sphere by developable pieces? Due to symmetry, there is no canonical "best" answer. Instead, we guide the surface toward a desired design by choosing an initial tessellation (shown at top) that breaks this symmetry.

Authors' addresses: Oded Stein; Eitan Grinspun, Columbia University, 530 West 120th Street, New York, NY, 10027; Keenan Crane, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213.

0730-0301/2018/8-ART77 $15.00
https://doi.org/10.1145/3197517.3201303

In general, there is no "best" approximation of a given smooth surface, since one can find successively closer approximations by smaller and smaller developable pieces, akin to a wrinkled shirt or a crumpled piece of paper. In our method, the final design is largely guided by the input tessellation (Figure 2) as well as how we choose to penalize non-developability (Figure 10). The final algorithms are straightforward to implement (Section 4.3); we explore how they can be used for developable design in Section 5.

2 RELATED WORK

2.1 Discrete Developability

Recently there has been an interest in *mimetic* notions of developability that exactly preserve key properties of developable surfaces even on coarse meshes, in the spirit of *discrete differential geometry* [Bobenko 2008; Crane and Wardetzky 2017]. For triangle meshes, a seemingly natural definition is the vanishing of *angle defect*, but this definition allows highly irregular geometry devoid of *ruling lines*, as discussed in Section 3.2. Other notions of discrete developability are based primarily on regular quadrilateral nets rather than triangulations, requiring a global quad layout that cannot easily adapt to changes in geometry. For instance, planar quadrilateral (PQ) strips provide a natural analogue of developable surfaces, since they can be isometrically flattened and have well-defined ruling lines [Liu et al. 2006; Sauer 1970]; Pottmann et al. [2008] consider *semi-discrete* developability, based on this same perspective. Solomon et al. [2012] present a framework where ruling directions are freely variable, but the global mesh layout must still be determined *a priori*. Recently, Rabinovich et al. [2018] propose a definition based on orthogonal geodesics, also requiring a global quad layout. Like the angle defect condition, meshes that exactly satisfy this definition can still be highly "crumpled"; ruling lines are defined locally at each vertex, but are not in general globally coherent (Figure 5). To provide editing of regular developable surfaces, this definition can be augmented with an auxiliary smoothness term. Our notion of discrete developability (Definition 3.1) is simultaneously compatible with both the zero angle defect notion (ensuring locally flattenability) as well as the PQ definition (providing global ruling lines); it also places no conditions on mesh combinatorics, making it suitable for general-purpose developable design.

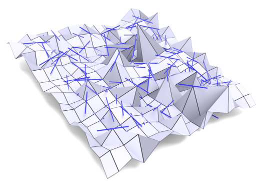


Fig. 5. Like triangulations with zero angle defect, discrete geodesic nets *à la* Rabinovich et al. [2018] can be highly crumpled—blue lines indicate local vertex rulings.

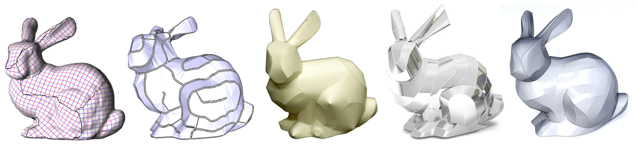


Fig. 3. A variety of methods have been developed to approximate a given surface by easily flattenable pieces. *Left to right:* Julius et al. [2005], Mitani and Suzuki [2004], Shatz et al. [2006], Tang et al. [2016] (which requires manual editing), and our method.

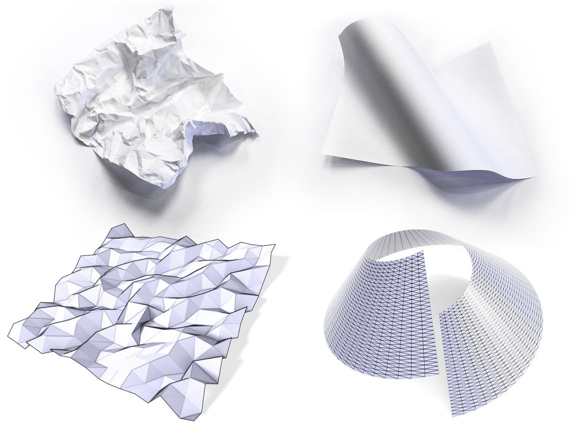


Fig. 4. Flattenability alone is not enough to ensure that a surface is easy to fabricate. For instance, both the crumpled piece of paper *(top left)* and noisy triangle mesh *(bottom left)* can be exactly flattened into the plane, but would be difficult to actually assemble from stiff material. In contrast, the smooth piece of paper *(top right)* and triangle mesh *(bottom right)* are both exactly flattenable *and* have straight *ruling lines* passing through each point. We seek surfaces of the latter kind.

2.2 Developable Design

Several methods approximate a given mesh by (near-)developable pieces. Wang and Tang [2004] directly optimize angle defect, which yields the crumpling behavior discussed in Section 3.2. Mitani and Suzuki [2004] generate *triangle strips* which can be trivially unfolded into the plane but lack clear ruling lines; Shatz et al. [2006] instead fit strictly conical regions which can have interior cone points; Decaudin et al. [2006] iteratively perform local fitting and projection. Massarwi et al. [2007] partition surfaces into *ruled* pieces, which become developable only upon triangulation. Rather than augment the geometry, Julius et al. [2005] find regions that can be flattened with low distortion. All of these methods apply some sequence of mesh processing operations (clustering, fitting, remeshing, *etc.*) involving parameters that can be difficult to understand and control; moreover, they are not suitable for the type of coarse form finding we explore in Section 5.2, and most do not provide clear ruling directions (which help facilitate manufacturing).

Other methods focus on user-guided *design* rather than automatic approximation—for instance, Kilian et al. [2008] explore *curved folding*, Tang et al. [2016] consider user-driven spline networks, and Rose et al. [2007] interpolate sketched boundaries; methods for simulating thin sheets [Narain et al. 2013; Schreck et al. 2015] might also be used for design exploration. In terms of output quality, there are no universally accepted criteria for what makes a "good" developable design; we show the results of several algorithms in Figure 3. On the whole, our approach produces models that are at least comparable in quality to previous work, and exhibit some nice features not exhibited by other methods such as (i) no requirement to partition the surface into disk-like pieces, (ii) automatic smoothing of feature lines, and (iii) natural emergence of ruling directions. These features arise naturally from our variational approach, *i.e.*, minimization of an energy supported in the star of each vertex.

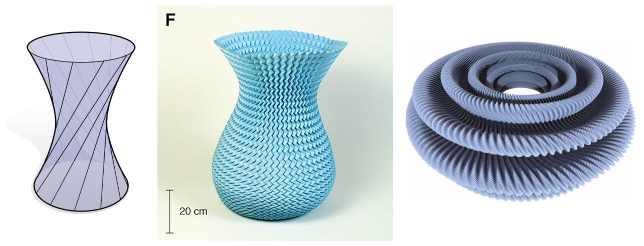
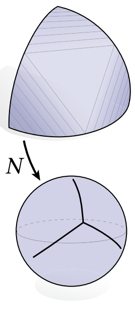


Fig. 6. Merely requiring a surface to have ruling lines does not ensure it can be flattened, such as the hyperboloid of revolution *(left)* which has negative Gaussian curvature. Conversely, a flattenable surface is not automatically ruled unless it is sufficiently regular—consider the C^0 *Miura-ori* origami pattern *(center)*, or the isometric embedding of the flat torus *(right)* which is highly wrinkled even though it exhibits C^1 continuity. We seek a discretization that captures both local flattenability and regularity. (Images from Dudte et al. [2016] and Borrelli et al. [2012], used with permission).

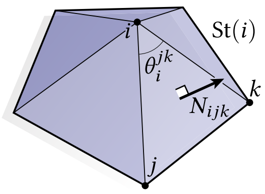
3 DISCRETE DEVELOPABILITY



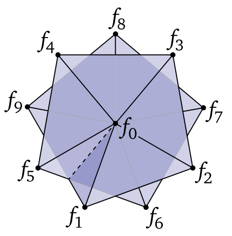
To develop our discrete definition, we first recall some basic facts about developable surfaces. For clarity we distinguish between a **flattenable** surface, which is locally isometric to the Euclidean plane, and a **developable** surface, which is also a twice differentiable (C^2) immersion. Equivalently, a developable surface is a C^2 immersion with zero Gaussian curvature ($K = 0$). The additional regularity of developable surfaces precludes pathological behavior that can occur when a surface is merely flattenable. For instance, a flattenable surface can look like a crumpled piece of paper or a corrugated piece of origami, whereas a developable surface must be a nice, smooth surface like a cone or cylinder, with a straight *ruling line* passing through each point (Figure 4). Under the Gauss map N, a developable surface therefore degenerates to a network of curves meeting at extrinsically flat regions where $\kappa_1 = \kappa_2 = 0$ (see inset). Note that not every ruled surface is developable (Figure 6, *left*), and flattenable surfaces that are C^1 but not C^2 may not be ruled (Figure 6, *right*).

Taking a cue from these definitions, our notion of discrete developability deviates from the usual notion that a triangle mesh is developable so long as it can be isometrically unfolded into the plane, instead demanding that it also have discrete *ruling lines*. This condition provides additional regularity, helping to mitigate the "crumpling" behavior discussed above. This definition provides the starting point for our variational formulation in Section 4.

3.1 Background and Notation



Throughout we use V, E, F to denote the vertices, edges, and faces of a simplicial surface M (*i.e.*, a manifold triangle mesh). An oriented simplex is expressed as an ordered list of vertex indices—for instance, ijk denotes a face with vertices $i, j, k \in \mathsf{V}$, and oriented edges ij, jk, ki along its boundary. Expressions of the form $u_i = \sum_{ij \in \mathsf{E}} v_{ij}$ mean that a quantity u_i associated with vertex i is obtained by summing a quantity v_{ij} over all edges containing i. We use St(i) to denote the *star* of a vertex i, *i.e.*, the collection of simplices containing i. The geometry is given by a simplicial map $f : \mathsf{M} \to \mathbb{R}^3$ interpolating coordinates f_i at each vertex $i \in \mathsf{V}$. Such a map is **nondegenerate** if all triangles have nonzero area; it is a **simplicial immersion** if it is locally injective, or equivalently, if every vertex star is embedded, *i.e.*, triangles intersect only at shared edges [Cervone 1996, Lemma 2.2]. The latter condition is stronger than the former: consider for instance the inset figure where the map f is not injective at 0, even though no triangle is degenerate. For any nondegenerate triangulation we use N_{ijk} to denote the unit normal of triangle ijk, and θ_i^{jk} for the interior angle at corner i of triangle ijk. The *angle defect* of a vertex $i \in \mathsf{V}$ is the sum $\Omega_i := 2\pi - \sum_{ijk \in \mathsf{F}} \theta_i^{jk}$, corresponding to the integral of Gaussian curvature over a small neighborhood around i.



3.2 Developability of Triangle Meshes

What does it mean for a triangle mesh to be developable? As in the smooth case, a natural idea is to require zero Gaussian curvature:

Definition. *A nondegenerate simplicial map $f : \mathsf{M} \to \mathbb{R}^3$ is* ***discrete flattenable*** *if the angle defect Ω_i at every vertex $i \in \mathsf{V}$ is zero.*

This condition ensures that the mesh can be locally flattened in the plane, since the angles around each vertex make a full 2π. Yet flattenability alone is not sufficient to characterize surfaces that are easily manufactured—consider for instance Figure 4, *(bottom left)*, which, despite its noisy appearance, has exactly zero angle defect at each vertex. This surface could in principle be constructed from an idealized flat sheet, but perhaps not from real physical materials like sheet metal. Moreover, flattenability alone does not ensure *normal convergence*: as shown by Thibert et al. [2005], a flattenable mesh may exhibit undesirable behavior even when inscribed in a smooth developable surface (such as poor approximation of surface area).

These observations motivate the need for a stronger condition, namely that (as in the smooth setting) a developable surface should not merely be flattenable, but also come with some kind of regularity that avoids degenerate or pathological behavior. In the smooth setting regularity is provided by C^2 differentiability; in the discrete setting (where we have at most one weak derivative) we replace this analytical condition with a geometric one. In particular:

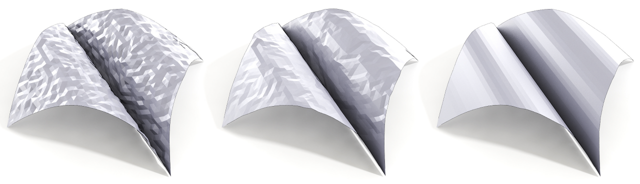


Fig. 7. Denoising a developable sheet *(left)* by simply minimizing angle defect leads to a flattenable but noisy surface *(center)*, whereas encouraging discrete developability yields a smoother ruled surface *(right)*.

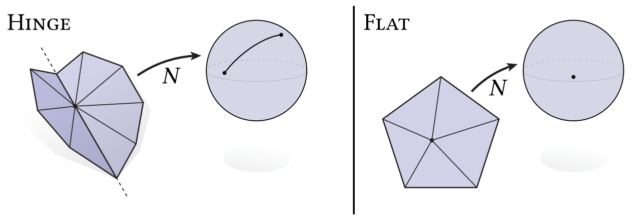
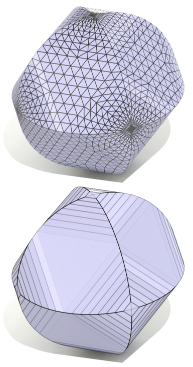


Fig. 8. A vertex is discrete developable if it looks like a HINGE or is FLAT.

Definition 3.1. A vertex star $\mathrm{St}(i)$ of a simplicial immersion $f : \mathsf{M} \to \mathbb{R}^3$ is a **hinge** if the triangles $ijk \in \mathrm{St}(i)$ can be partitioned into two edge-connected regions over which the normals N_{ijk} are constant; f is **discrete developable** if every vertex star is a hinge. Any vertex that is not a hinge is a **seam vertex**.

Figure 8 illustrates the prototypical configuration for a HINGE: two planar regions intersecting in a pair of antiparallel edges. A special case is a FLAT configuration (all vertices in a common plane), which admits many partitions into two flat regions. In addition to being locally flattenable, any non-flat discrete developable surface is (by Proposition A.2) *discrete ruled*:

Definition 3.2. A simplicial immersion $f : \mathsf{M} \to \mathbb{R}^3$ is **discrete ruled** if every vertex $i \in \mathsf{V}$ is contained in a path of parallel edges $i_0 i_1, i_1 i_2, \ldots, i_{n-1} i_n \in \mathsf{E}$ with endpoints i_0, i_n on $\partial\mathsf{M}$.



Importantly, however, not all discrete ruled surfaces are discrete developable: the latter implies that the triangles between two rulings are all contained in a common plane. Definition 3.1 is therefore compatible with the standard notion of discrete developability for quad meshes [Liu et al. 2006; Sauer 1970]: if edges with zero dihedral angle are removed, what remains is a collection of planar quad (PQ) strips. For instance, the inset shows a developable triangulation where edges shaded according to their dihedral angle, revealing a PQ mesh (Figures 14 and 26 are also rendered this way).

Normal Convergence. Normal convergence of a mesh inscribed in a smooth developable surface ensures convergence of other properties, such as the *development* (*i.e.*, unfolding) into the plane—see [Morvan and Thibert 2006, Theorem 1]. Discrete developability seems to automatically imply good normal approximation: for instance, if the normals of a sequence of triangulations approach the normals of an inscribing smooth developable surface (which approach a small arc on the Gauss map), then the triangulations will, by Theorem 3.3, approach discrete developability. Conversely, one might also argue that if a sequence of discrete developable triangulations are inscribed in a (nowhere flat and sufficiently regular) smooth developable surface, its ruling lines will eventually be contained in ruling lines of the smooth surface, ensuring normal convergence (though more rigorous arguments are needed here).

3.3 Alternative Characterization

Though geometrically straightforward, the notion of discrete developability (Definition 3.1) can be difficult to optimize directly. We therefore consider an alternative characterization:

THEOREM 3.3. *An embedded vertex star* $\mathrm{St}(i)$ *forms a hinge if and only if all its triangle normals* N_{ijk} *are contained in a common plane.*

PROOF. By definition a hinge has at most two distinct normals, which are always contained in a common plane. Conversely, if all normals are contained in a plane P, then the cross product of two distinct normals $N_{ijk} \neq N_{jil}$ of triangles sharing an edge ij will be parallel to both the edge vector $e_{ij} := f_j - f_i$ and the normal v of P. Hence, if the normal changes across more than two edges, more than two edges will be parallel to v. But since all edges emanate from the same vertex i, such a $\mathrm{St}(i)$ cannot be embedded. Likewise, if $\mathrm{St}(i)$ has exactly two distinct normals, they can differ across only two edges; if it has only one normal it is trivially a hinge. □

This result is fairly surprising: merely asking that all normals lie in a common plane forces them to "bifurcate" into two distinct directions, corresponding to the two planes of the hinge. The first condition (embeddability) plays a role in this bifurcation by preventing the kind of degenerate cases illustrated in Figure 9. An easy corollary of Theorem 3.3 is that a vertex is a hinge if and only if the minimum width of its Gauss image is zero—a fact that we will exploit in our variational formulation (Section 4.1).

Corner Cases. Configurations shown in Figure 9 help to further understand Definition 3.1. The SPIKE, NEEDLE, and FIN approach configurations where normals are coplanar and yet the vertex star is not embedded—as in the smooth setting, the condition that f must be a (simplicial) immersion provides additional regularity. The DOUBLECOVER further motivates the need to be immersed rather than merely nondegenerate: this configuration has a zero-width Gauss map but is not locally injective and hence fails to be a hinge.

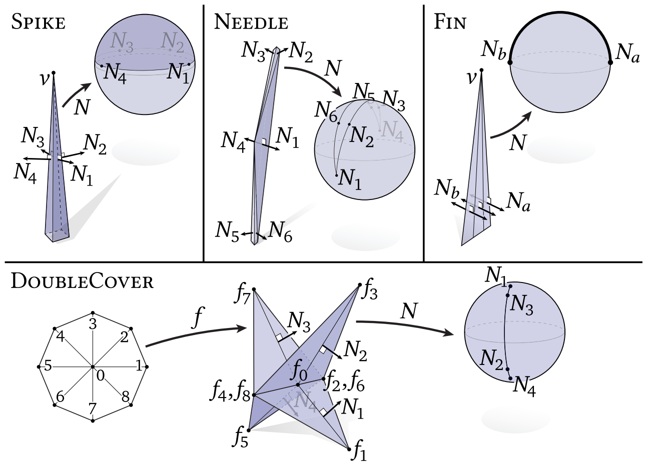


Fig. 9. As in the smooth setting, the requirement that the surface be immersed avoids degenerate configurations such those pictured above.

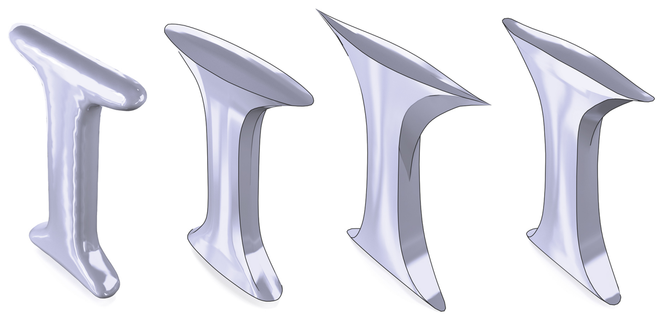


Fig. 10. A given surface *(left)* evolves toward a piecewise developable approximation using the combinatorial energy *(center left)* or the covariance energy *(center right)*. Modifying the covariance energy reduces the formation of sharp spikes *(right)*.

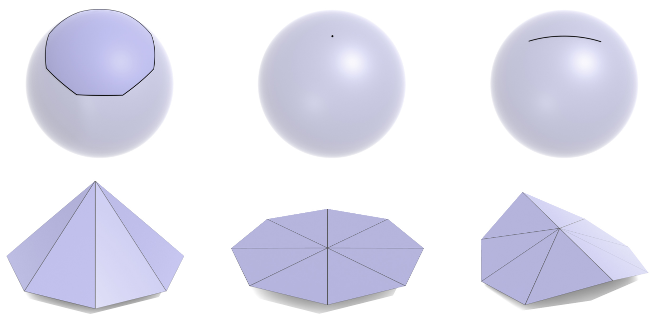


Fig. 12. *Bottom:* For a single vertex *(left)*, minimizing the squared angle defect produces a flat patch *(center)*, whereas our energy ($\mathcal{E}^\lambda$) yields a hinge *(right)*. *Top:* image of the vertex star under the Gauss map.

4 DEVELOPABLE APPROXIMATION

The definition of discrete developability (Section 3) can be used as the starting point for algorithms that seek to design or approximate developable surfaces. Here we consider a simple variational approach: formulate an energy that measures the developability of each vertex, and apply numerical descent. One might also use this definition as the starting point for other algorithms—some enticing ideas are to use discrete developability as a constraint for shape space exploration [Yang et al. 2011] or to seek a developable triangulation that interpolates points sampled from a smooth developable surface [Peternell 2004; Thibert et al. 2005], though we do not pursue those directions here.

4.1 Energy

The basic idea behind our variational formulation is to penalize the width of the Gauss image associated with each vertex star, *i.e.*, the polygon on the unit sphere made by consecutive triangle normals. There is no canonical way to measure this width—our only hard requirement is that the energy of hinge vertices must be zero. We therefore consider two possibilities, namely (i) a simple but computationally expensive combinatorial energy based on a direct interpretation of Definition 3.1, and (ii) a less expensive energy based on Theorem 3.3 which measures the covariance of the triangle normals. Gradient descent on either energy yields an evolution that exhibits a key property needed for developable design: empirically, a given triangle mesh tends toward one that is discrete developable away from a sparse collection of seam curves, where all Gaussian curvature is concentrated (see Figure 22). Interestingly enough, this behavior does *not* arise from the use of a sparsity-inducing norm, but rather from the simple geometric fact that seam curves are not energetically significant (Section 4.1.3)—in fact, using an L^1 norm in this context yields surfaces that are piecewise *flat*, rather than piecewise developable, as demonstrated in Figure 11. Finally, we do not explicitly encourage pairs of edges to be antiparallel, since flat regions need not be discrete ruled—in fact, flat regions connecting curved pieces may *require* a triangulation with no antiparallel pairs.

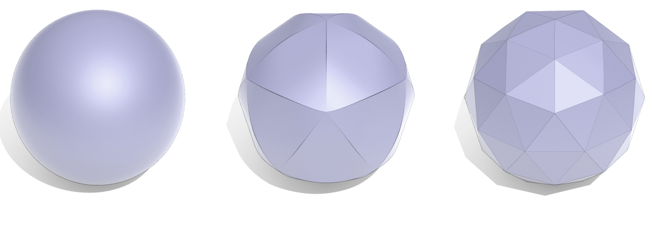


Fig. 11. Minimization of our energy tends to concentrate the Gaussian curvature of a given smooth surface *(left)* onto a sparse collection of seam curves *(center)*. Using a sparsity-inducing L^1 norm to achieve a similar effect results in regions that are flat rather than developable *(right)*.

4.1.1 Combinatorial Width. Suppose we partition the faces of a given vertex star $\mathrm{St}(i)$ into two edge-connected regions $F_1, F_2 \subset \mathsf{F}$ of cardinality n_1, n_2 (*resp.*). Letting $\overline{N}_p := \frac{1}{n_p}\sum_{ijk\in F_p} N_{ijk}$ be the average of the triangle normals N_{ijk} in region F_p, the degree to which the partition $P := \{F_1, F_2\}$ looks like a hinge can be quantified by the deviation of the normals in each region from their mean:

$$\begin{aligned}\pi(P) \quad &:= \quad \textstyle\sum_{p=1,2} \frac{1}{n_p} \sum_{\sigma\in F_p} |N_\sigma - \overline{N}_p|^2 \\ &= \quad \textstyle\sum_{p=1,2} \frac{1}{n_p^2} \sum_{\sigma_1,\sigma_2\in F_p} |N_{\sigma_1} - N_{\sigma_2}|^2.\end{aligned} \tag{1}$$

Here N_σ denotes the unit normal of a triangle $\sigma \in \mathsf{F}$. Vertex i is then a hinge if and only if there is a bipartition P for which $\pi(P)$ is zero. Letting $\mathcal{P}_i$ denote the set of all edge-connected bipartitions of $\mathrm{St}(i)$, we therefore define a local energy

$$\mathcal{E}_i^{\mathcal{P}} := \min_{P\in\mathcal{P}_i} \pi(P). \tag{2}$$

The *combinatorial energy* $\mathcal{E}^{\mathcal{P}}$ is then the sum of $\mathcal{E}_i^{\mathcal{P}}$ over all vertices. This energy is expensive to evaluate, since for a vertex of valence k we must sum $O(k^2)$ terms for each of $O(k^2)$ partitions. Nonetheless, it is a piecewise differentiable function of the vertex positions f_i, and can hence be optimized via standard subgradient methods as described in Section 4.3.

4.1.2 Covariance. Alternatively, consider the characterization given in Theorem 3.3, which says that an embedded vertex star $\mathrm{St}(i)$ forms a hinge if and only if all of its triangle normals are contained in a common plane. To quantify how hinge-like a vertex is, we can therefore measure the average alignment of the normals with the

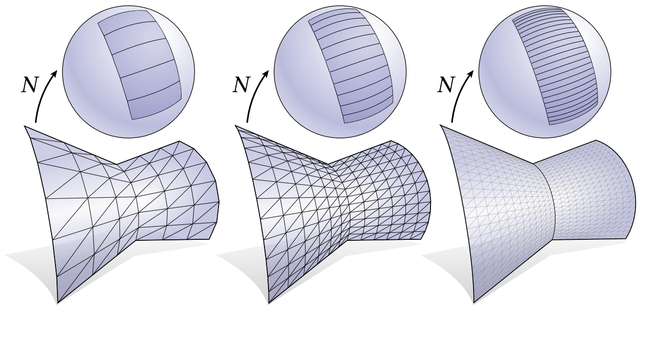


Fig. 13. For a discrete developable triangulation inscribed in a piecewise developable surface, energy is nonzero only at seam vertices, where the Gauss map is nondegenerate. This energy goes to zero under regular refinement.

unit normal $u \in \mathbb{R}^3$ of the best-fit plane:

$$\lambda_i := \min_{|u|=1} \sum_{ijk \in \mathsf{F}} \theta_i^{jk} \langle u, N_{ijk} \rangle^2. \tag{3}$$

Angle weights ensure that the energy does not change if we consider a different tessellation of the same piecewise linear surface; note that this quantity is also scale invariant. Since Equation 3 is just the variational form of an eigenvalue problem, λ_i can also be expressed as the smallest eigenvalue of the 3×3 *normal covariance matrix*

$$A_i := \sum_{ijk \in \mathsf{F}} \theta_i^{jk} N_{ijk} N_{ijk}^T. \tag{4}$$

The normal covariance matrix also arises in the context of surface descriptors [Berkmann and Caelli 1994] and quadric-based mesh simplification [Garland and Heckbert 1997], where it can be shown that its eigenvalues are related to the squares of the principal curvatures—see for instance [Garland 1999, Section 4.4]. This fact provides some intuition for the behavior of this energy: for instance, it makes less aggressive changes to the shape of the vertex star than penalizing the discrete Gaussian curvature Ω_i, since on the Gauss sphere it corresponds to penalizing smallest width rather than area (see Figure 12).

The developability of the whole mesh is measured via

$$\mathcal{E}^\lambda := \sum_{i \in \mathsf{V}} \lambda_i.$$

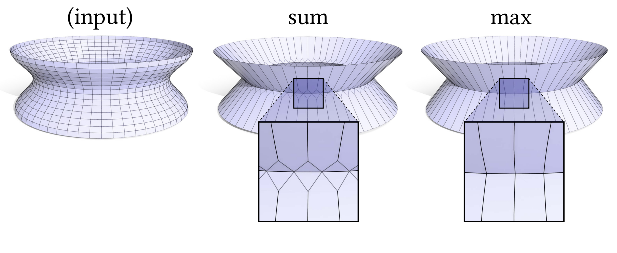
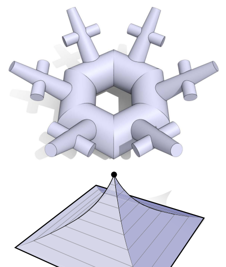


Fig. 14. Minimizing our covariance energy naturally pushes the input *(left)* toward a piecewise developable surface, but ruling lines may branch into "V" shapes along seams *(center)*. Replacing summation with maximization yields straight ruling lines *(right)*.

As with the energy $\mathcal{E}^\mathcal{P}$, the only degrees of freedom are the vertex positions f_i. Away from repeated eigenvalues this energy is smooth and can be minimized using standard techniques; the main cost is computing the eigenvalues λ_i and their associated eigenvectors (see Section 4.3 for details). It may however encourage Spike-like vertices, whose normals approach a common plane (Figure 9). A simple remedy is to measure the same energy *intrinsically* via the exponential map on the sphere—see Appendix B.5 for details.

4.1.3 Piecewise Developable Surfaces. An interesting behavior of the energies $\mathcal{E}^\mathcal{P}$ and $\mathcal{E}^\lambda$ is that they tend to encourage not only globally developable surfaces, but also *piecewise* developable surfaces. This behavior can be understood by considering the regular refinement sequence depicted in Figure 13, where a mesh is inscribed in a pair of developable pieces meeting along a non-developable seam. Since these meshes are discrete developable away from the seam, energy is nonzero only at seam vertices. However, since $\mathcal{E}^\mathcal{P}$ and $\mathcal{E}^\lambda$ effectively measure the *square* of the smallest width of polygons on the Gauss sphere, the energy contributed by such a seam goes to zero under regular subdivision—consider that the sum of squared widths is a vanishingly small fraction of the sum of widths, which is roughly constant. As a result, surfaces made of many developable pieces (inset, *top*) can approach zero energy in the limit of refinement, though energy will still be nonzero at points where several seams meet (inset, *bottom*).



4.1.4 Branching. At seams where two developable pieces meet, both energies can produce ruling lines that branch into two (Figure 14, *center*). A simple example is shown in Figure 15, where a branched configuration (*left*) and a perfectly ruled configuration (*right*) yield Gauss images of equal minimum width. However, the branched configuration will have lower energy, since both $\mathcal{E}^\lambda$ and $\mathcal{E}^\mathcal{P}$ take a *sum* of terms, thereby providing an average notion of polygon width. We can avoid this behavior by simply replacing summation with maximization, *i.e.*, $\pi(P)$ from Equation 1 becomes

$$\pi^{\max}(P) := \max_{F_k \in P} \max_{N_1, N_2 \in F_k} (N_2 - N_1)^2, \tag{5}$$

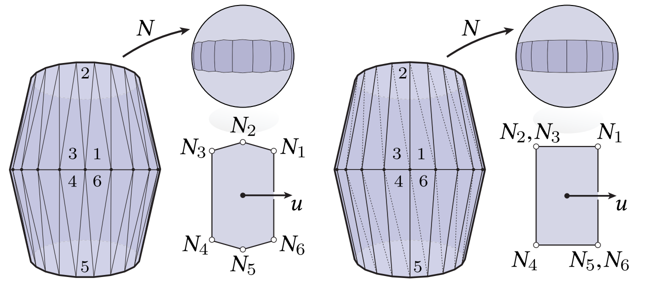


Fig. 15. Measuring the average deviation from planarity can result in "branching" behavior at seam vertices where two developable pieces meet *(left)*; minimizing the worst alignment with any normal encourages straight rulings *(right)*.

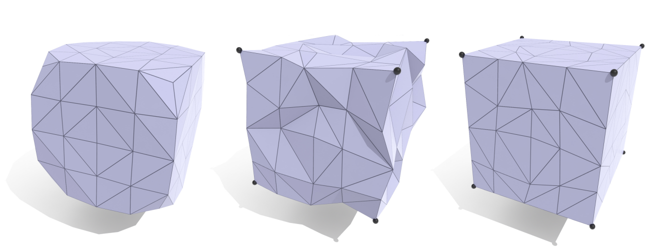


Fig. 16. Valence-3 vertices occur at points where three seams meet; in order to be hinge-like such vertices must be flat, causing rounding of features like corners of the cube *(left)*. We therefore omit the contribution of valence-3 vertices from the overall energy, enabling for instance a noisy cube *(center)* to become a piecewise developable cuboid *(right)*.

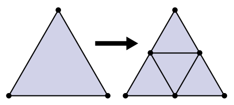
and the objective in Equation 3 becomes instead

$$\lambda_i^{\max} := \min_{|u|=1} \max_{ijk\in F} \langle u, N_{ijk} \rangle^2. \tag{6}$$

(Angle weights can be omitted since we no longer consider a sum.) In the case of Figure 15, the two configurations for the seam vertices now have identical energy, and the remaining vertices in the mesh can freely evolve toward clean ruling lines. Though more expensive to optimize, this strategy can be used to produce cleaner ruling lines, as demonstrated in Figure 14, *right*.

Triple Points. In a piecewise developable surface, seams generically meet at a triple point—this observation fits together well with the fact that valence-3 hinge vertices must be flat (Proposition A.3). Rather than force such vertices to be developable, we simply omit their contribution to the energy, allowing them to serve as triple points. Figure 16 shows one example.

4.2 Regularity



In the limit of refinement, seams contribute nothing to the energy (Section 4.1.3). For any finite mesh, however, seam vertices have nonzero energy—minimization of this energy provides a natural smoothing of seam curves, even though we do nothing to explicitly detect or extract these curves (see Figure 17). Since the magnitude of energy shrinks as the seam area goes to zero, we can use length scale of the mesh to control the degree of regularization: first, we minimize the energy on an initial coarse mesh to get the basic shape. Once the norm of the energy gradient is below a given threshold (or the overall design is simply satisfactory) we apply regular 4-1 subdivision to all triangles (see inset) and continue minimizing in order to refine seams and improve developability. In practice we tend to start with fairly coarse meshes, and use no more than two or three rounds of subdivision. In Figure 1 for instance we start with a mesh of about 1k triangles and subdivide twice; notice the natural smoothing of contours in the face.

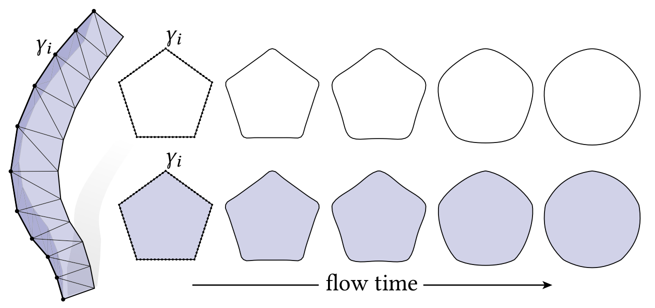


Fig. 17. Minimizing our energy automatically smooths out seam curves. To compare with planar curvature flow, we construct a surface mesh around a given planar curve *(left)*. On this mesh we obtain an evolution *(bottom)* nearly identical to standard elastic curve flow in the plane *(top)*.

4.3 Numerical Optimization

Energy minimization can be performed via any standard numerical technique for nonsmooth optimization; explicit expressions for gradients are given in Appendix B. We experimented with a variety of methods and found that L-BFGS using the line search of Lewis and Overton [2013] yields the best results; to get a basic implementation up and running, one can also use standard (sub)gradient descent with Armijo-Wolfe line search [Nocedal and Wright 2006, Chapter 3.1]. In practice we also found that small interior angles can adversely affect the performance of line search—for triangles with two small angles we therefore perform edge flips; for triangles with a single small angle we simply perform an edge collapse (see inset). Otherwise, we do not perform any remeshing. All calculations were performed in double precision; for the energy $\mathcal{E}^\lambda$ we found that an accurate 3×3 eigensolver [Kopp 2008] is needed to achieve convergence.

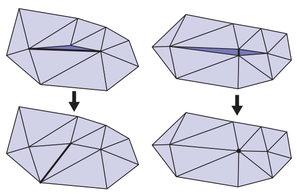


Fig. 18. To improve numerical stability we flip *(left)* or collapse *(right)* triangles with small angles; otherwise, we do not perform any special remeshing to encourage developability.

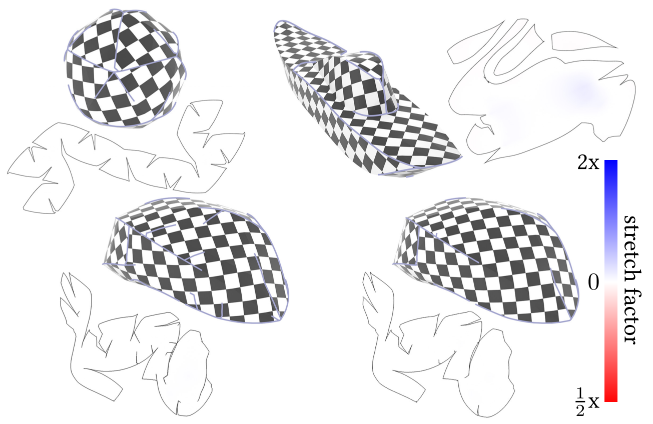
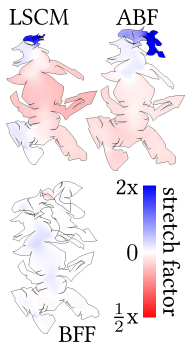


Fig. 19. Since we do not partition surfaces into patches, they can often be cut and flattened into a single contiguous piece *(top left)*, though global overlaps may need to be resolved by cutting into additional pieces *(top right)*. *Bottom:* gluing together tiny "cracks" yields even nicer layouts. (Mostly white coloring indicates that flattening yields virtually zero metric distortion.)

4.4 Cutting and Flattening

During minimization, we need not identify which vertices are developable and which are seams. For fabrication or subsequent processing, one may however wish to cut the model into explicit developable pieces (as in Figure 19). We use a simple, automatic strategy: first identify vertices with energy above a user-specified tolerance $\varepsilon > 0$, then compute a cut passing through all such vertices via the method of Erickson and Har-Peled [2004] (for surfaces with spherical topology, just a minimal spanning tree). The tolerance can be viewed as a very rough proxy for material stiffness: materials with strong membrane stiffness (like paper) require one to cut more aggressively through non-flat vertices than materials that stretch more easily (like leather).

To get aesthetically pleasing cuts we set edge weights to a linear combination of (i) length and (ii) the dot product between the edge direction and the smaller eigenvector of the matrix A_i at each endpoint (Equation 4), thereby encouraging cuts to align with principal directions (though such weights are not strictly required for flattening). Since the cut surface has almost no Gaussian curvature, a conformal flattening should in principle produce a near-isometric map to the plane. In practice, however, methods such as least-squares conformal maps (LSCM) or angle based flattening (ABF) yield large area distortion (see inset), since pieces connected by thin regions can be scaled independently with little penalty. We instead use the boundary first flattening (BFF) algorithm [Sawhney and Crane 2017], which allows one to enforce isometry along the boundary. Figure 20 illustrates the effect of the tolerance ε on the cut, and shows that we obtain simpler cuts over time. In all figures, we quantify scale distortion via the log conformal factor of the flattening, indicated by the red–blue scale. In practice, we glue together small "cracks" in parameter space (which have little effect on the final layout), and repeat the flattening (see Figure 19, *bottom*). Global overlaps are avoided by segmenting the flattening by hand (as in Figure 19, *top right*), though automatic tools could of course be used instead [Sorkine et al. 2002]. An interesting question for future work is whether we can automatically avoid global overlap, perhaps using recent ideas about automatic texture chartification [Poranne et al. 2017].



4.5 Discussion

Empirically, different choices of energy, surface tessellation, or even numerical descent strategy have an effect on the resulting piecewise developable approximation, as shown in Figure 10. This fact is not at all surprising in light of the ill-posed nature of the problem itself: in general, there is not one clear "best" way to approximate a smooth surface by developable pieces. As illustrated in Figure 2, the effect is particularly pronounced in near-spherical regions (κ_1 close to κ_2), where there is no clear preferred direction for ruling lines. Especially in the intial stage, the choice of mesh and energy therefore has a pronounced effect on the final design; at finer levels of tessellation, all energies tend to have very similar behavior.

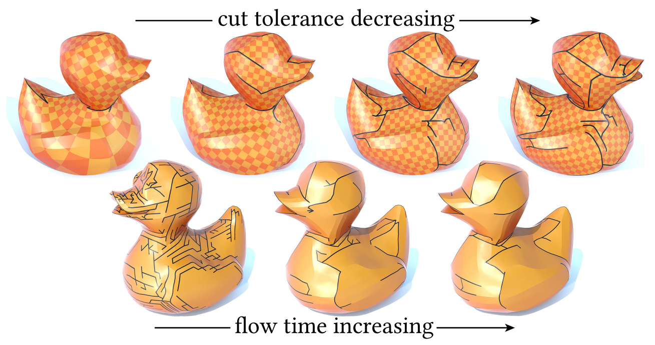


Fig. 20. *Top row:* Increasing the cut tolerance ε accommodates progressively stiffer materials; here setting ε = .01, .005, .002, .001 results in a progressively lower average scale distortion of 2.4, .09, .006, .0004, *resp.* (also indicated by the checkerboard pattern). *Bottom row:* for a fixed tolerance ε, we obtain shorter and shorter cuts over time.

5 RESULTS

Here we perform numerical experiments and explore preliminary applications to surface approximation and developable design (Section 5.2). We also take a brief forward look toward manufacturing (Sections 5.3 and 5.4).

5.1 Validation

Several experiments indicate that optimization of discrete developability exhibits the expected behavior. Figure 22 confirms that curvature concentrates onto seams, becoming highly flattenable everywhere else (as seen in Figure 19). Figure 7 confirms that merely minimizing the squared angle defect Ω (*i.e.*, the discrete Gaussian curvature) yields a surface that is perfectly flattenable yet highly "crumpled" like a piece of paper—in contrast, we obtain a smoother surface with clear ruling lines. Figure 17 confirms that seams are automatically smoothed out, as discussed in Section 4.2. Figure 2 emphasizes that for many surfaces (such as the sphere), there may be no clear "best" piecewise developable approximation. Instead, one can guide the solution toward a desired result by picking an initial triangulation that roughly suggests important geometric features.

Figure 26 shows the effect of tessellation on surfaces that are already close to developable. Here we minimize the covariance energy $\mathcal{E}^{\lambda}$; no preprocessing is performed, and no constraints or projections keep the mesh close to the initial surface. For these simple examples, the tessellation has little effect on the overall shape, but still encourages a ruling line parameterization. Note also that the choice of mesh will affect the *number* of ruling lines: for instance, the cone at center has far fewer vertices on the top boundary component than the bottom—since (by Proposition A.2) ruling lines must have endpoints on the boundary, there cannot be a discrete developable surface with a ruling line passing through all bottom boundary vertices. For a similar reason, local adaptivity provides little value, since not every vertex in the "fine" region can belong to its own ruling.

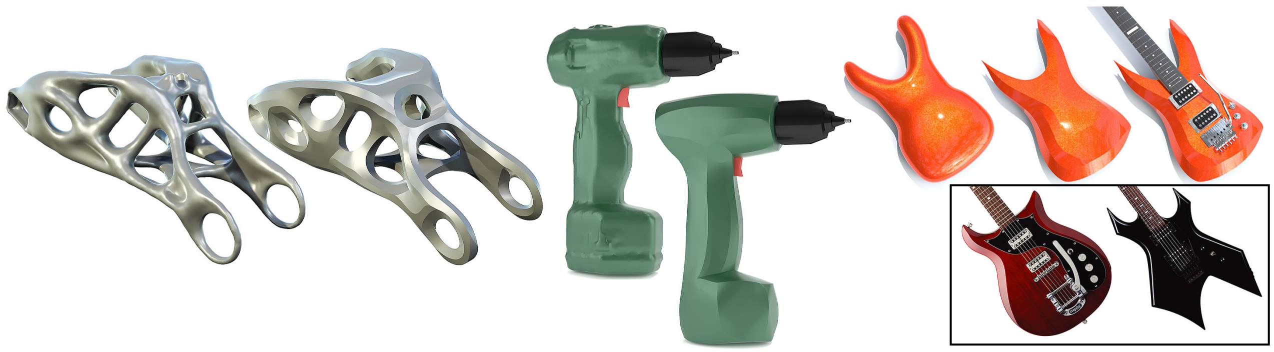


Fig. 21. Here we approximate a given shape (leftmost in each image) with a surface that is developable away from seam curves. Unlike methods that partition the surface into individual pieces, seams can blend organically into the design. *Left to right:* a *swingarm* model produced via topological optimization (courtesy Autodesk); the handle of a drill; and a guitar body, which naturally yields features appearing in real guitar designs *(bottom right)*.

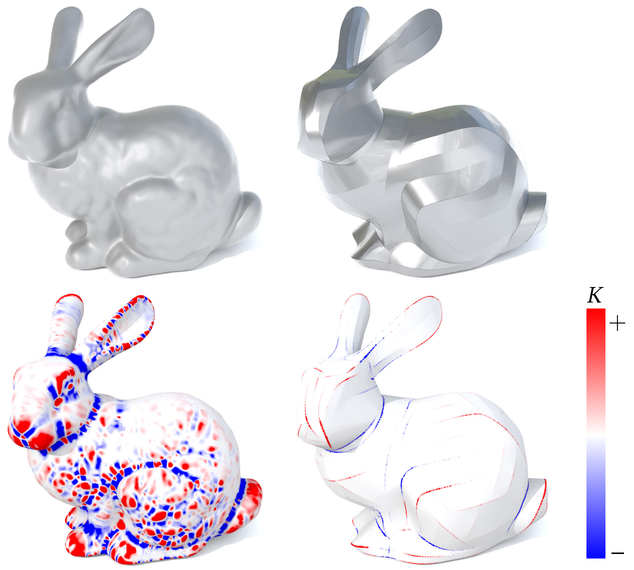


Fig. 22. On the Stanford bunny *(top row)*, Gaussian curvature K concentrates onto a sparse collection of curves *(bottom row)*.

5.2 Developable Design

A long-standing challenge in digital manufacturing is automatic approximation of general curved surfaces by high-quality ruled developable pieces. Our method represents the first stage in such a pipeline, taking any unstructured mesh to one or more piecewise developable designs. A more complete pipeline might entail, *e.g.*, conversion into a clean network of developable splines (as considered in Tang et al. [2016]), which in turn facilitates tool path planning for cylindrical flank milling [Chu and Chen 2005]. We ran our method on a variety of models, including those pictured in Figures 1, 22 and 21.

A top-down description of our algorithm is as follows. The input to our algorithm is most typically a coarse mesh of the target surface, obtained either via coarsening (as described below) or by "sketching out" a rough target design, akin to modeling a coarse subdivision cage. Given this initial mesh, we perform energy minimization (Section 4.3) until the gradient is sufficiently small, or we are satisfied with the rough shape. We then apply regular 4-1 subdivision (Section 4.2) and this process is repeated until we are satisfied with the quality of the fine mesh. Refinement steps are currently executed "by hand" based on aesthetic judgements that are part of the design process; one might also implement automatic refinement based on the norm of the gradient.

Examples are shown in Figure 23, where each model captures the basic design intent, but also suggests design possibilities not originally conceived by the user. Since there are many different piecewise developable approximations of a given surface, the choice of input mesh will influence the final result. This effect is most pronounced for coarse meshes, where different tessellations can have a significant effect on global geometry (Figure 27); for finer meshes they effectively act as different parameterizations, and have a less significant geometric effect (Figure 26). To obtain coarse input for Figures 3 *far right*, 10, 21, 22, 23, 25, 24 we ran the freely-available meshing tool of Jakob et al. [2015], using a field with triangular symmetry and extrinsic alignment. Since this tool aims for uniform element size, we also ran Willmore flow [Bergou et al. 2006] on Figure 21 *left*, to capture the thin handles on the front of the swingarm.

As illustrated in Figure 10, different energies will yield different solutions; we used the combinatorial energy $\mathcal{E}^{\mathcal{P}}$ for Figures 1, 2, and 25, and the covariance energy $\mathcal{E}^{\lambda}$ for Figures 11 *center*, 7; in Figures 20, 21, and 22 we used the intrinsic version of this energy described in Appendix B.5.1. In Figures 23 and 24 we used $\mathcal{E}^{\mathcal{P}}$ in the coarse phase and $\mathcal{E}^{\lambda}$ in the fine phase. Though different energies can be used to tweak the design, we find that in general just using one of the energies (say, $\mathcal{E}^{\lambda}$) will produce reasonable default designs for most models.

Runtimes for all examples were on the order of seconds to a few minutes on a 3.4GHz Intel Core i7 laptop with 32GB of RAM; we did not use multithreading, though gradient calculations could easily be parallelized. Especially during the initial coarse phase, the mesh rapidly evolves toward a shape that looks much like the final design; subsequent refinement takes no more than a few minutes, especially with L-BFGS and an appropriate line search (Section 4.3).

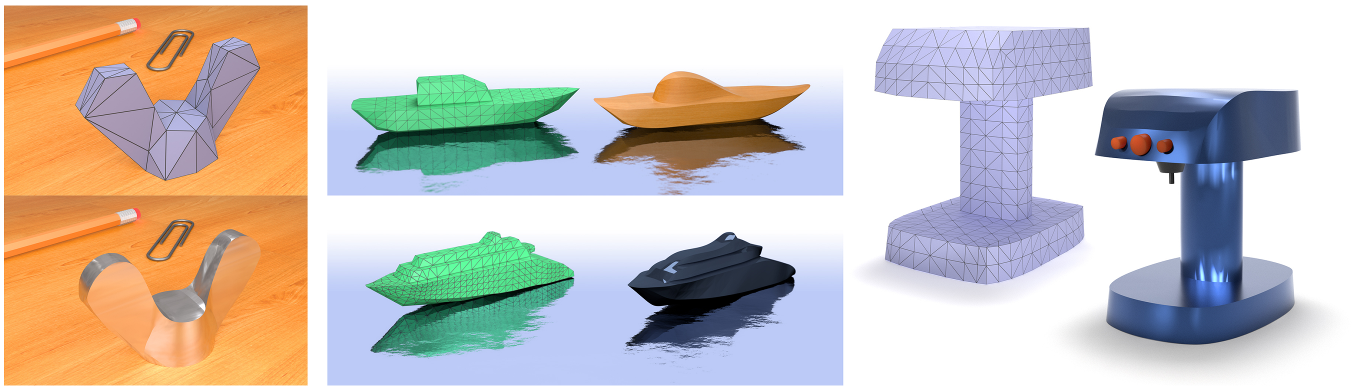


Fig. 23. Beyond the traditional task of shape approximation, our method enables developable shape exploration by starting with a coarse mesh and optimizing/refining toward a piecewise developable surface with organic flowlines. (Input mesh in each figure is shown in wireframe.)

5.3 Paper Craft

As a simple test of physical feasibility we fabricated a piecewise developable surface using a consumer cutting plotter (Silhouette CAMEO), shown in Figure 24. Here additional cuts had to be made to avoid overlap in the flattened domain. An interesting direction for future work is to optimize surfaces toward those with *curved folds* [Kilian et al. 2008], so that they can be formed from a single sheet of material; one possibility here is to penalize only the angle defect Ω_i along curved folds (rather than $\mathcal{E}^{\mathcal{P}}$ or $\mathcal{E}^{\lambda}$).

5.4 Flank Milling

Though our method provides only one small piece in a larger pipeline, we can still get a glimpse of how our designs might facilitate manufacturing via *flank milling*. As opposed to traditional *point milling*, which carves out material using only the tip of a cutting tool, flank milling sweeps a cylindrical bit *along* the surface, allowing for faster cuts and higher-quality surfaces [Harik et al. 2013]. By starting with a piecewise developable design, one can ensure that material is not erroneously added nor subtracted to the manufactured piece due to out-of-plane twisting of the bit along the toolpath. A common practice is to first cast or 3D print the bulk *near net shape* (thereby avoiding excess waste), then use flank milling to obtain accurate mating surfaces, threads, *etc.*, which must come into precise contact. Figure 25 prototypes such a process starting with our method, followed by manual extraction of PQ strips in a polygon mesh editor, and subsequent toolpath generation in *Autodesk Fusion 360*. Milling was performed using via a *Pocket NC V1*, a USD $5k hobbyist-level 5-axis mill. Though each individual contour is easily machined, extracting a global spline network, as well as planning a globally feasible toolpath, remains an interesting challenge for future work.

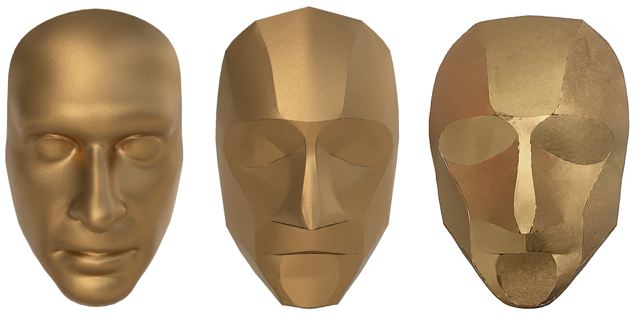


Fig. 24. Simple test of physical feasibility: an initial mesh of a face *(left)* is optimized to obtain a piecewise developable approximation *(center)*, which is then cut from paper, glued together, and spray painted to obtain a physical model *(right)*.

6 LIMITATIONS AND FUTURE WORK

Though we have laid some of the basic foundations, many questions still remain. On the theoretical side, there is still the fundamental question of how to formulate the task of finding the "best" piecewise developable approximation as a well-posed problem: simply asking for the closest approximation (*e.g.*, in the Hausdorff sense) is not meaningful since one can find "crumpled" solutions that are arbitrarily close to a given surface. The variational approach provides an enticing framework for thinking about such questions. In fact, this work was originally inspired by thinking about a gradient flow on the smooth energy

$$\mathcal{E}(f) := \int_M \kappa_1^2 \, dA,$$

which penalizes the smaller (in magnitude) principal curvature κ_1 of a surface $f : M \to \mathbb{R}^3$. Exploring the connections between this smooth energy and our discrete variational formulation is therefore a natural topic for future work.

At a more practical level, our current optimization strategy does not produce perfectly straight rulings except on fairly simple meshes; a better understanding of this issue could lead to meshes with a cleaner PQ structure (perhaps by incorporating more sophisticated remeshing). More broadly, our method is only the first step in a fully automated pipeline for taking a given input surface all the way through the process of developable approximation, decomposition into clean developable pieces (such as PQ strips or spline developables), and final manufacturing via roll bending or flank milling. For milling, a complete solution would also need to accommodate mechanical constraints such as collision avoidance, possibly via

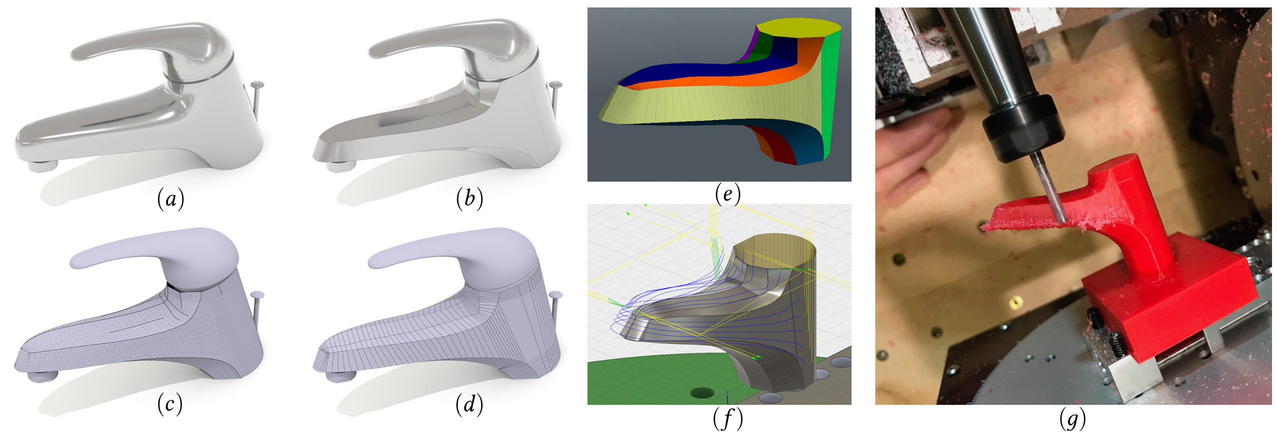


Fig. 25. Prototype of fabrication via flank milling. An originally smooth faucet design (a) evolves into a piecewise developable surface (b) with fairly clear seam curves and ruling lines (c) which are extracted by hand (d) in a polygon mesh editor and partitioned into PQ strips (e). These strips are then semi-automatically converted into tool paths (f) via NURBS patches. A contour of the final piece is then flank milled using a hobbyist 5-axis CNC mill (g).

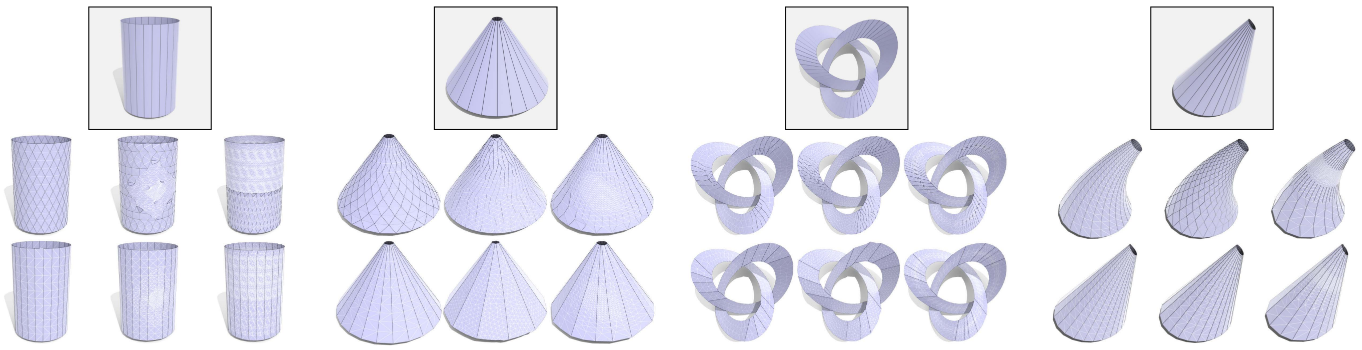


Fig. 26. Here we examine the effect of the tessellation on surfaces that are developable (*far left, center left, center right*) or nearly developable (*far right*). In each case the middle row shows the input and the bottom row shows the result of optimization; a smooth developable surface is shown at top for reference. Dark edges indicate large dihedral angles, and white edges indicate zero dihedral angles.

joint optimization of the geometry and the tool path. In general we are hopeful that a notion of discrete developability for general triangle meshes, as well as the variational point of view, will provide fertile soil for future work in developable surface processing, approximation, and design.

ACKNOWLEDGEMENTS

Thanks to Timothy Sun for preliminary discussion and investigation, and to Ryan Schmidt for performing the toolpath extraction and milling in Figure 25. This work was sponsored in part by NSF Awards 1717320 1409286, 1717268, and 1319483, and gifts from Adobe, Autodesk Research, Pixar, and SideFX Software.

REFERENCES

Miklos Bergou, Max Wardetzky, David Harmon, Denis Zorin, and Eitan Grinspun. 2006. A Quadratic Bending Model for Inextensible Surfaces. In *Symposium on Geometry Processing*, Alla Sheffer and Konrad Polthier (Eds.). The Eurographics Association. DOI: http://dx.doi.org/10.2312/SGP/SGP06/227-230

J. Berkmann and T. Caelli. 1994. Computation of Surface Geometry and Segmentation using Covariance Techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16, 11 (1994), 1114–1116.

Alexander Bobenko. 2008. *Discrete Differential Geometry*. Springer London, Limited.

Vincent Borrelli, Saïd Jabrane, Francis Lazarus, and Boris Thibert. 2012. Flat tori in three-dimensional space and convex integration. *Proc. Natl. Acad. Sci. U. S. A.* 109, 19 (2012), 7218–7223.

Davide P Cervone. 1996. Tight immersions of simplicial surfaces in three space. *Topology* 35, 4 (1996), 863 – 873. DOI: http://dx.doi.org/10.1016/0040-9383(95)00047-X

Lian Chang. 2015. The Software Behind Frank Gehry's Geometrically Complex Architecture. (2015). https://priceonomics.com/the-software-behind-frank-gehrys-geometrically/ Online (Priceonomics.com); posted May 12, 2015.

Chih-Hsing Chu and Jang-Ting Chen. 2005. Tool path planning for five-axis flank milling with developable surface approximation. *The International Journal of Advanced Manufacturing Technology* 29, 7 (12 Oct 2005), 707. DOI: http://dx.doi.org/10.1007/s00170-005-2564-6

Keenan Crane and Max Wardetzky. 2017. A Glimpse Into Discrete Differential Geometry. *Notices of the American Mathematical Society* 64, 10 (November 2017), 1153–1159.

Philippe Decaudin, Dan Julius, Jamie Wither, Laurence Boissieux, Alla Sheffer, and Marie-Paule Cani. 2006. Virtual Garments: A Fully Geometric Approach for Clothing Design. *Computer Graphics Forum* (2006). DOI: http://dx.doi.org/10.1111/j.1467-8659.2006.00982.x

Levi H Dudte, Etienne Vouga, Tomohiro Tachi, and L Mahadevan. 2016. Programming curvature using origami tessellations. *Nature materials* (2016).

Jeff Erickson and Sariel Har-Peled. 2004. Optimally Cutting a Surface into a Disk. *Discrete & Computational Geometry* 31, 1 (2004), 37–59. DOI: http://dx.doi.org/10.1007/s00454-003-2948-z

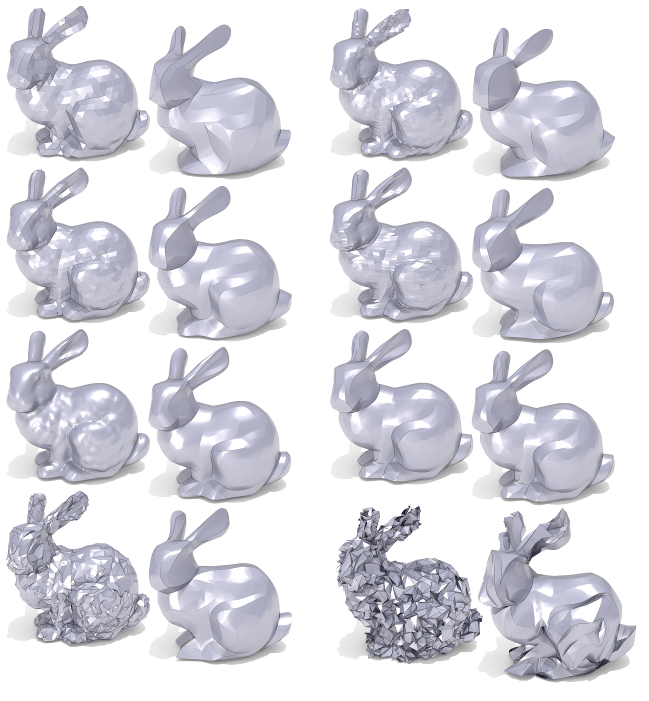


Fig. 27. Different initial meshes lead to different piecewise developable approximations, though the bulk shape typically remains quite close to the input. Overall, fine meshes tend to yield a closer approximation by a larger number of pieces; edges roughly aligned with principal curvatures tend to better preserve geometric features. Moderate noise is typically smoothed out; large amounts of noise can affect the overall shape.

Michael Garland. 1999. *Quadric-Based Polygonal Surface Simplification.* Ph.D. Dissertation. Carnegie Mellon University.

Michael Garland and Paul S. Heckbert. 1997. Surface Simplification Using Quadric Error Metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97).* ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 209–216. DOI: http://dx.doi.org/10.1145/258734.258849

Ramy F. Harik, Hu Gong, and Alain Bernard. 2013. Review: 5-axis Flank Milling: A State-of-the-art Review. *Comput. Aided Des.* 45, 3 (March 2013), 796–808. DOI: http://dx.doi.org/10.1016/j.cad.2012.08.004

Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Instant Field-aligned Meshes. *ACM Trans. Graph.* 34, 6, Article 189 (Oct. 2015), 15 pages. DOI: http://dx.doi.org/10.1145/2816795.2818078

Dan Julius, Vladislav Kraevoy, and Alla Sheffer. 2005. D-Charts: Quasi-Developable Mesh Segmentation. *Computer Graphics Forum* (2005). DOI: http://dx.doi.org/10.1111/j.1467-8659.2005.00883.x

Martin Kilian, Simon Flöry, Zhonggui Chen, Niloy J. Mitra, Alla Sheffer, and Helmut Pottmann. 2008. Curved Folding. *ACM Trans. Graph.* 27, 3, Article 75 (Aug. 2008), 9 pages. DOI: http://dx.doi.org/10.1145/1360612.1360674

J. Kopp. 2008. Efficient Numerical Diagonalization of Hermitian 3×3 Matrices. *International Journal of Modern Physics C* 19 (2008), 523–548. DOI: http://dx.doi.org/10.1142/S0129183108012303

Adrian S. Lewis and Michael L. Overton. 2013. Nonsmooth optimization via quasi-Newton methods. *Math. Program.* A, 141 (2013), 135–163.

Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. 2006. Geometric Modeling with Conical Meshes and Developable Surfaces. *ACM Trans. Graph.* 25, 3 (July 2006), 681–689. DOI: http://dx.doi.org/10.1145/1141911.1141941

Fady Massarwi, Craig Gotsman, and Gershon Elber. 2007. Papercraft Models using Generalized Cylinders. In *Proceedings of the Pacific Conference on Computer Graphics and Applications, Pacific Graphics 2007, Maui, Hawaii, USA, October 29 - November 2, 2007.* 148–157. DOI: http://dx.doi.org/10.1109/PG.2007.16

Jun Mitani and Hiromasa Suzuki. 2004. Making Papercraft Toys from Meshes Using Strip-based Approximate Unfolding. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 259–263. DOI: http://dx.doi.org/10.1145/1015706.1015711

Jean-Marie Morvan and Boris Thibert. 2006. Unfolding of Surfaces. *Discrete & Computational Geometry* 36, 3 (01 Oct 2006), 393–418. DOI: http://dx.doi.org/10.1007/s00454-006-1255-x

Rahul Narain, Tobias Pfaff, and James F. O'Brien. 2013. Folding and Crumpling Adaptive Sheets. *ACM Transactions on Graphics* 32, 4 (July 2013), 51:1–8. http://graphics.berkeley.edu/papers/Narain-FCA-2013-07/ Proceedings of ACM SIGGRAPH 2013, Anaheim.

Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization* (2nd ed.). Springer.

Martin Peternell. 2004. Developable Surface Fitting to Point Clouds. *Comput. Aided Geom. Des.* 21, 8 (Oct. 2004), 785–803. DOI: http://dx.doi.org/10.1016/j.cagd.2004.07.008

Roi Poranne, Marco Tarini, Sandro Huber, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Autocuts: Simultaneous Distortion and Cut Optimization for UV Mapping. *ACM Trans. on Graphics - Siggraph Asia 2017* 36, 6 (2017). http://vcg.isti.cnr.it/Publications/2017/PTHPS17

Helmut Pottmann, Alexander Schiftner, Pengbo Bo, Heinz Schmiedhofer, Wenping Wang, Niccolo Baldassini, and Johannes Wallner. 2008. Freeform Surfaces from Single Curved Panels. *ACM Trans. Graph.* 27, 3, Article 76 (Aug. 2008), 10 pages. DOI: http://dx.doi.org/10.1145/1360612.1360675

Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2018. Discrete Geodesic Nets for Modeling Developable Surfaces. *ACM Trans. Graph.* 37, 2, Article 16 (Feb. 2018), 17 pages. DOI: http://dx.doi.org/10.1145/3180494

Kenneth Rose, Alla Sheffer, Jamie Wither, Marie-Paule Cani, and Boris Thibert. 2007. Developable Surfaces from Arbitrary Sketched Boundaries. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing (SGP '07).* Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 163–172. http://dl.acm.org/citation.cfm?id=1281991.1282014

Robert Sauer. 1970. *Differenzengeometrie.* Springer.

Rohan Sawhney and Keenan Crane. 2017. Boundary First Flattening. *CoRR* abs/1704.06873 (2017). arXiv:1704.06873 http://arxiv.org/abs/1704.06873

Camille Schreck, Damien Rohmer, Stefanie Hahmann, Marie-Paule Cani, Shuo Jin, Charlie C.L. Wang, and Jean-Francis Bloch. 2015. Non-smooth developable geometry for interactively animating paper crumpling. *ACM Transactions on Graphics* 35, 1 (Dec. 2015), 10:1–10:18. DOI: http://dx.doi.org/10.1145/2829948

Idan Shatz, Ayellet Tal, and George Leifman. 2006. Paper craft models from meshes. *The Visual Computer* 22, 9-11 (2006), 825–834.

Justin Solomon, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2012. Flexible Developable Surfaces. *Computer Graphics Forum* (2012). DOI: http://dx.doi.org/10.1111/j.1467-8659.2012.03162.x

Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. 2002. Bounded-distortion Piecewise Mesh Parameterization. In *Proceedings of the Conference on Visualization '02 (VIS '02).* IEEE Computer Society, Washington, DC, USA, 355–362. http://dl.acm.org/citation.cfm?id=602099.602154

Chengcheng Tang, Pengbo Bo, Johannes Wallner, and Helmut Pottmann. 2016. Interactive Design of Developable Surfaces. *ACM Trans. Graph.* 35, 2, Article 12 (Jan. 2016), 12 pages. DOI: http://dx.doi.org/10.1145/2832906

Boris Thibert, Jean-Pierre Gratier, and Jean-Marie Morvan. 2005. A direct method for modeling and unfolding developable surfaces and its application to the Ventura Basin (California). *Journal of Structural Geology* 27, 2 (2005), 303–316. DOI: http://dx.doi.org/10.1016/j.jsg.2004.08.011

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* 20, 8 (2004), 521–539.

Yong-Liang Yang, Yi-Jun Yang, Helmut Pottmann, and Niloy J. Mitra. 2011. Shape Space Exploration of Constrained Meshes. *ACM Trans. Graph.* 30, 6, Article 124 (Dec. 2011), 12 pages. DOI: http://dx.doi.org/10.1145/2070781.2024158

A PROPERTIES OF DISCRETE DEVELOPABLE TRIANGULATIONS

As described in Definition 3.1, a vertex i is a *hinge* if $\mathrm{St}(i)$ is embedded and its triangles can be partitioned into two edge-connected flat regions; it is *flat* if the normals of all its triangles are parallel.

Proposition A.1. *If an interior vertex i is a non-flat hinge vertex, then it is contained in a pair of antiparallel edges $ia, ib \in \mathrm{St}(i)$.*

Proof. Let N_1, N_2 be the normals of the two flat regions of $\mathrm{St}(i)$; since these regions are edge-connected, there will be exactly two edges ia, ib that share both normals. Since i is not flat, the normals must be distinct ($N_1 \neq N_2$); since it is embedded, they must not be antiparallel ($N_1 \neq -N_2$). Hence, the cross products $N_1 \times N_2 = -N_2 \times N_1$ yield nonzero vectors parallel to the two edges ia, ib. □

Proposition A.2. *Consider a discrete developable immersion f with no flat vertices. Then f is discrete ruled.*

Proof. By Proposition A.1, any interior vertex i must have a pair of antiparallel edges ia, ib; let N_1, N_2 be the distinct normals determining the edge directions. Since $\mathrm{St}(a)$ and $\mathrm{St}(b)$ each share a pair of triangles with normals N_1, N_2, they will each contain a pair of antiparallel edges along the same line (or a single edge in the case of boundary vertices). □

Proposition A.3. *Any valence-3 hinge vertex $i \in \mathsf{V}$ is necessarily flat.*

Proof. Suppose i were not flat. Then by Proposition A.1 it would have a pair of antiparallel edges va, vb. But since i has valence 3, va and vb must be edges of the same triangle, *i.e.*, i, a, and b are collinear. Hence $\mathrm{St}(i)$ is not a hinge, since it is not embedded. □

B ENERGY AND GRADIENT EVALUATION

This section provides explicit expressions for evaluating the energies described in Section 4.1 and their gradients.

B.1 Derivatives of Basic Quantities

Our energies depend only on the triangle areas $\mathcal{A}_{ijk}$, triangle normals N_{ijk} and interior angles θ_i^{jk}, which have the following gradients with respect to vertex positions f:

$$\nabla_{f_i}\mathcal{A}_{ijk} = \tfrac{1}{2}N_{ijk} \times (f_k - f_j), \tag{7}$$

$$\nabla_{f_i}N_{ijk} = \frac{1}{\mathcal{A}_{ijk}}((f_k - f_j) \times N_{ijk})N_{ijk}^{\mathsf{T}}, \tag{8}$$

$$\begin{array}{rcl} \nabla_{f_j}\theta_i^{jk} &=& N_{ijk} \times (f_i - f_j)/|f_i - f_j|, \\ \nabla_{f_k}\theta_i^{jk} &=& N_{ijk} \times (f_k - f_i)/|f_k - f_i|, \\ \nabla_{f_i}\theta_i^{jk} &=& -(\nabla_{f_j}\theta_i^{jk} + \nabla_{f_k}\theta_i^{jk}). \end{array} \tag{9}$$

Since these quantities depend only on the positions of vertices i, j, and k, the gradients with respect to any other vertex are zero.

B.2 Combinatorial Energy

To evaluate the gradient of the combinatorial energy $\mathcal{E}_i^{\mathcal{P}}$ associated with vertex i, we first identify the partition P minimizing $\pi(P)$ (Equation 1). The gradient of a single term in this sum with respect to the position f_p of any vertex $p \in \mathsf{V}$ can then be expressed via

$$\nabla_{f_p}|N_{\sigma_1} - N_{\sigma_2}|^2 = 2\langle N_{\sigma_1} - N_{\sigma_2}, \nabla_{f_p}N_{\sigma_1} - \nabla_{f_p}N_{\sigma_2}\rangle,$$

where the normal gradient is given in Equation 8. The energy gradient is then the sum over all such terms. In the case where there are two or more partitions of equal energy, the gradient of any of them will be a subgradient of the piecewise smooth energy $\mathcal{E}^{\lambda}$, which is still suitable for the first-order descent strategy outlined in Section 4.3. To avoid branching (Section 4.1.4), the gradient of any maximal term provides a subgradient for Equation 5.

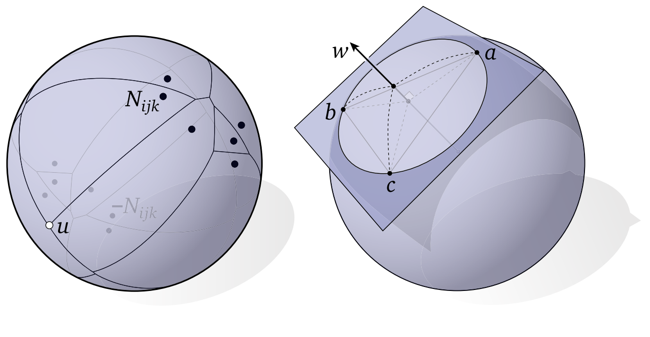


Fig. 28. The maximal covariance energy is easily evaluated by sampling the objective at the vertices of a spherical Voronoi diagram *(left)*, which are simply the unit normals of the triangles formed by three points on the unit sphere *(right)*.

B.3 Covariance Energy

At any vertex $i \in \mathsf{V}$, let λ be an eigenvalue of the matrix $A_i := \sum_{ijk \in \mathsf{F}} \theta_i^{jk} N_{ijk}N_{ijk}^{\mathsf{T}}$ with associated eigenvector x. Then the gradient of λ with respect to the position $f_p \in \mathbb{R}^3$ of any vertex $p \in \mathsf{V}$ is

$$\nabla_{f_p}\lambda = \sum_{ijk \in \mathsf{F}} (x^T N_{ijk})^2 \nabla_{f_p}\theta_i^{jk} + 2\theta_i^{jk}(x^T N_{ijk})(\nabla_{f_p}N_{ijk})^T x, \tag{10}$$

where we have simply applied the chain rule and the identity $\nabla_A \lambda = xx^T$. Expressions for $\nabla_{f_p}N_{ijk}$ and $\nabla_{f_p}\theta_i^{jk}$ are given in Equations 8 and 9, *resp.*

B.4 Maximal Covariance

Energy. To evaluate the energy given by Equation 6, let

$$\phi(u) := \max_{ijk \in \mathsf{F}} \langle u, N_{ijk}\rangle^2.$$

This function is piecewise smooth over spherical Voronoi cells associated with the unit normals N_{ijk} and their antipodes $-N_{ijk}$ (see Figure 28, *left*). Its minimum is therefore found at a vertex of the spherical Voronoi diagram, which will be the spherical centroid of some triple of sites. Since ϕ achieves a minimum at a Voronoi vertex, minimizing ϕ over *all* triples necessarily yields the optimal value $\lambda_i^{\max}$. From the perspective of performance and numerical stability, simply evaluating ϕ for all triples is more attractive than explicitly building the Voronoi diagram, especially since the number of distinct triples is typically very small. To compute the spherical centroid of three unit vectors a, b, c, note that the geodesic circumcenter of a spherical triangle coincides with the unit normal of the plane containing the triangle's vertices (Figure 28, *right*). The location of the site is therefore just $w = (b - a) \times (c - a)/|(b - a) \times (c - a)|$. To avoid a zero denominator we simply omit redundant sites.

Subgradient. Since ϕ is a maximum over a collection of convex differentiable functions, the gradient of any maximizing term provides a subgradient that can be used for optimization (Section 4.3). In particular, let v be the unit vector minimizing ϕ, let M be the maximizing normal, and let $a, b, c \in \mathsf{F}$ be the triple of triangles whose normals define v. Then the subgradient $\nabla_{f_p}\lambda_i^{\max}$ with respect to

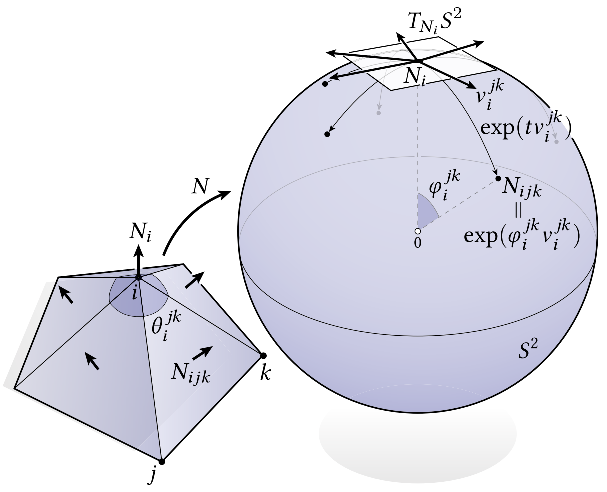


Fig. 29. For vertex stars with a large Gauss image (such as a Spike) one can quantify the width of the spherical polygon as the width of a 2D polygon whose vertices v_i^{jk} are taken to the triangle normals $\widetilde{N}_{ijk}$ by the exponential map exp at the vertex normal N_i.

the position f_p of a vertex $p \in \mathsf{V}$ can be expressed as

$$2\langle v, M\rangle\left(\frac{\left\langle v, e_p \times M\right\rangle}{2A_M}M + \sum_{\sigma\in\{a,b,c\}}\frac{\left\langle v, e_{\sigma|p}\times N_\sigma\right\rangle\langle e_\sigma \times v, M\rangle}{4A_{abc}A_\sigma}N_\sigma\right),$$

where N_σ is the unit normal of triangle $\sigma \in \mathsf{F}$, A_M and A_p are the areas of triangles with normals M and p (*resp.*), A_{abc} is the Euclidean area of a triangle with vertices a, b, c, $e_p \in \mathbb{R}^3$ is the edge vector opposite vertex p in the triangle with normal M (or zero if p is not contained in this triangle), and $e_{\sigma|p}$ is the edge across from vertex p in triangle σ.

B.5 Intrinsic Width

Energy. The energy $\mathcal{E}^\lambda$ (Section 4.1) quantifies the width of a polygon on the sphere via the covariance of extrinsic unit vectors $N_{ijk} \in \mathbb{R}^3$, which can lead to artifacts (*e.g.*, Spikes) for large polygons. An intrinsic notion of width is obtained by instead expressing this polygon in terms of the exponential map at the center of the polygon (Figure 29). In particular, if N_i is the area-weighted vertex normal at a vertex $i \in \mathsf{V}$ (*i.e.*, the unit vector in the direction $\sum_{ijk\in\mathsf{F}} A_{ijk}N_{ijk}$) and φ_i^{jk} is the angle from N_i to some triangle normal N_{ijk} in $\mathrm{St}(i)$, then the triangle normal itself can be expressed as

$$N_{ijk} = \exp_{N_i}(\varphi_i^{jk} v_i^{jk})$$

for some unit tangent vector v, where $\exp_p$ denotes the exponential map at a point p on the 2-sphere S^2 (see Figure 29). More explicitly, this vector can be obtained by simply projecting N_{ijk} onto the plane of N_i and normalizing:

$$\begin{aligned}\tilde{v}_i^{jk} &:= N_{ijk} - \langle N_{ijk}, N_i\rangle N_i,\\ v_i^{jk} &:= \tilde{v}_i^{jk}/|\tilde{v}_i^{jk}|.\end{aligned}$$

Letting $\tilde{N}_i^{jk} := \varphi_i^{jk} v_i^{jk}$, the width of the spherical polygon can then be quantified via the smallest eigenvalue of the 2×2 matrix

$$\tilde{A}_i := \sum_{ijk\in\mathsf{F}} \theta_i^{jk}\tilde{N}_i^{jk}(\tilde{N}_i^{jk})^\mathsf{T},$$

mirroring Equation 4.

Gradient. Let N_i be the area weighted normal at vertex $i \in \mathsf{V}$, let $v_i^{jk} := N_i \hat{\times} N_{ijk}$, let $\mu_i := v_i^{jk}\hat{\times}N_i$, and let $\mu_f := v_i^{jk}\hat{\times}N_{ijk}$, where $u\hat{\times}v := u\times v/|u\times v|$ denotes the normalized cross product. Then the gradient of $\tilde{N}_i^{jk}$ with respect to the position f_p of a vertex $p \in \mathsf{V}$ can be expressed as

$$\begin{aligned}\nabla_{f_p}\tilde{N}_i^{jk} = &\left(\mu_i\mu_f^T + \frac{\varphi_{ijk}}{\sin\varphi_{ijk}}v_i^{jk}(v_i^{jk})^T\right)\nabla_{f_p}N_{ijk}\\ &-\left(\mu_i\mu_f^T + \varphi_{ijk}N_i\mu_i^T + \frac{\varphi_{ijk}}{\tan\varphi_{ijk}}v_i^{jk}(v_i^{jk})^T\right)\nabla_{f_p}N_i\end{aligned} \tag{11}$$

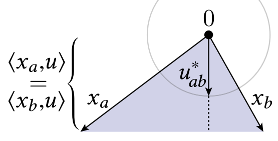
The gradients for N_i and N_{ijk} can be expressed via the expressions from Appendix B.1; the gradient of the overall energy can then be expressed by substituting $\tilde{N}_i^{jk}$ for N_{ijk} in Equation 10.

B.5.1 Branching.

Energy. In the intrinsic case, one can avoid the branching artifacts described in Section 4.1.4 by penalizing the minimum width of the convex hull of the n points $\widetilde{N} \in \mathbb{R}^2$. This width can be computed via the method of rotating calipers in $O(n\log n)$ time, including construction of the convex hull. However, since n is always quite small (about six on average) a simpler implementation is to just minimize the energy

$$\min_{|u|=1}\underbrace{\max_{ijk,\,ipq\in\mathrm{St}(i)}\langle\widetilde{N}_i^{jk} - \widetilde{N}_i^{pq}, u\rangle^2}_{=:\psi}$$

by enumerating all distinct pairs of vectors $x_a = \pm\widetilde{N}_i^{jk}$, $x_b := \pm\widetilde{N}_i^{pq}$. The minimizing vector u_{ab}^* for any such pair will be the vector pointing along the altitude of the triangle $(0, x_a, x_b)$ (see inset), and one can easily show that the minimum width of the convex hull is then the value of ψ among all such vectors u_{ab}^*. The subgradient is found by simply taking the gradient of the term maximizing ψ—here the only new expression is the gradient of the unit altitude u_{ab}^*, given by



$$\nabla_{x_a}u_{ab}^* = -\tfrac{1}{|w|^3}(N_i\times w)w^\mathsf{T},$$

where $w := x_b - x_a$ (and likewise for x_b).