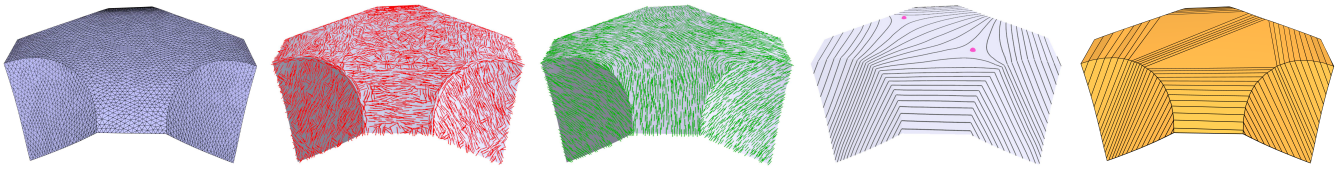
Dev2PQ: Planar Quadrilateral Strip Remeshing of Developable Surfaces

FLOOR VERHOEVEN, ETH Zurich, Switzerland
AMIR VAXMAN, Utrecht University, The Netherlands
TIM HOFFMANN, TU Munich, Germany
OLGA SORKINE-HORNUNG, ETH Zurich, Switzerland



(a) input mesh (b) locally estimated rulings (c) gradient direction field (d) optimized scalar field (e) remeshing result

Fig. 1. Developable shapes can be digitally acquired by 3D scanning or freeform modeling (a). In such scenarios, the meshing is typically not aligned to principal curvature directions, which hampers practical applications, such as fabrication with flat polygonal panels (Fig. 2). Our method remeshes an input mesh of a (piecewise) developable surface into a curvature aligned, planar polygonal mesh (e) by computing a vector field (c), from which we integrate a function whose isolines (d) align as well as possible to the locally estimated noisy rulings (b). Our vector field contains automatically-placed singularities in the planar region (d), which result in naturally placed triangular patches.

We introduce an algorithm to remesh triangle meshes representing developable surfaces to planar quad dominant meshes. The output of our algorithm consists of planar quadrilateral (PQ) strips that are aligned to principal curvature directions and closely approximate the curved parts of the input developable, and planar polygons representing the flat parts of the input that connect the PQ strips. Developable PQ-strip meshes are useful in many areas of shape modeling, thanks to the simplicity of fabrication from flat sheet material. Unfortunately, they are difficult to model due to their restrictive combinatorics. Other representations of developable surfaces, such as arbitrary triangle or quad meshes, are more suitable for interactive freeform modeling, but generally have non-planar faces or are not aligned to principal curvatures. Our method leverages the modeling flexibility of non-ruling based representations of developable surfaces, while still obtaining developable, curvature aligned PQ-strip meshes. Our algorithm optimizes for a scalar function on the input mesh, such that its isolines are extrinsically straight and align well to the locally estimated ruling directions. The condition that guarantees straight isolines is nonlinear of high order and numerically difficult to enforce in a straightforward manner. We devise an alternating optimization method that makes our problem tractable and practical to compute. Our method works automatically on any developable input, including multiple patches and curved folds, without explicit domain decomposition. We demonstrate the effectiveness of our approach on a variety of developable surfaces and show how our remeshing can be used alongside handle based interactive freeform modeling of developable shapes.

Authors' addresses: Floor Verhoeven, ETH Zurich, Switzerland, floor.verhoeven@inf.ethz.ch; Amir Vaxman, Utrecht University, The Netherlands, a.vaxman@uu.nl; Tim Hoffmann, TU Munich, Germany, tim.hoffmann@ma.tum.de; Olga Sorkine-Hornung, ETH Zurich, Switzerland, sorkine@inf.ethz.ch.

0730-0301/2022/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS Concepts: • **Computing methodologies** → **Mesh models**; **Mesh geometry models**.

Additional Key Words and Phrases: developable surfaces, planar quadrilateral meshes, curvature line nets, remeshing

ACM Reference Format:
Floor Verhoeven, Amir Vaxman, Tim Hoffmann, and Olga Sorkine-Hornung. 2022. Dev2PQ: Planar Quadrilateral Strip Remeshing of Developable Surfaces. *ACM Trans. Graph.* 1, 1 (January 2022), 18 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Developable surfaces are commonly used in architecture and product design due to the simplicity of their fabrication. Such surfaces are locally isometric to a planar domain, which means they can be manufactured by mere bending of sheet material, such as metal. Freeform developable surfaces form a rich and interesting shape space, but they are notoriously difficult to design due to their highly constrained nature. Therefore in most cases, only simple forms are used, such as cylinders and cones.

The majority of methods for developable surface modeling use rulings based representations (see, e.g., [Solomon et al. 2012; Tang et al. 2016]), or isometry optimization (see, e.g., [Burgoon et al. 2006; Fröhlich and Botsch 2011]). The recently proposed discrete orthogonal geodesic nets [Rabinovich et al. 2018] and the checkerboard pattern isometries [Jiang et al. 2020] represent developable surfaces without explicitly accounting for principal curvature directions or patch decomposition. A more recent approach to creating developable surfaces is through the approximation of existing shapes with developables. Proposed methods include cutting surfaces into developable pieces [Sharp and Crane 2018], fitting developable patches to a surface [Ion et al. 2020] and flowing existing surfaces to reduce a discrete developability measure [Binninger et al. 2021; Kohlbrenner et al. 2021; Liu and Jacobson 2021; Sellán et al. 2020; Stein et al. 2018].

While these discrete representations of developable surfaces are excellently suited for creative exploration and design of freeform developable *shapes*, they fall short of providing a suitable final representation for manufacturing. For that purpose, it is especially important to have planar mesh faces aligned to principal curvature directions [Alliez et al. 2003; Liu et al. 2006; Tang et al. 2016], that only the ruling-based representations provide. A curvature-line representation of a developable surface possesses the desired properties for fabrication once the shape is fixed, since the minimal curvature lines on a developable surface are rulings on which the normal is constant; as such, they can be easily tessellated into planar polygons that approximate the surface shape well. In fact, meshes comprised of planar quadrilateral strips (with no interior vertices) constitute a well-known model for discrete developable surfaces, whose refinement and convergence properties have been studied [Liu et al. 2006]. Additionally, the planar quadrilateral strip representation allows for the shape to be fabricated by bending developable material, such as sheet metal, along the ruling edges of the mesh (as shown in Fig. 23). General planar quadrilateral meshes that do not consist of strips, i.e., not aligned to minimal curvature, are not suitable for this fabrication process.

In this paper, we develop a method to convert a triangle mesh representation of a developable surface into a discrete curvature-line representation in order to reap the benefits of both worlds: the support for developable shape creation provided by a representation of choice, and the desirable properties for fabrication offered by the curvature-line representation. Our method produces strips of planar quadrilaterals (PQ) aligned to principal curvature directions that closely approximate the curved parts of a given input mesh, along with planar polygons representing the flat parts of the input (see Fig. 1). In particular, our method produces precisely straight rulings, modeled as individual edges in the output mesh.

The past decade has seen a highly active stream of fruitful research on field aligned quad meshing, where principal curvature fields have naturally received special attention [Bommes et al. 2012; de Goes et al. 2015; Vaxman et al. 2016]. However, to the best of our knowledge, no existing general remeshing method is guaranteed to produce completely straight edge sequences that are consistent with curvature lines, which are often difficult to obtain fully and faithfully

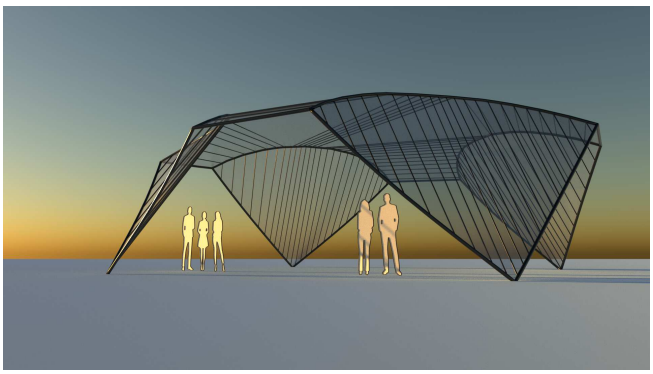


Fig. 2. An architectural illustration of our result from Fig. 1, fitted with flat glass-like panels.

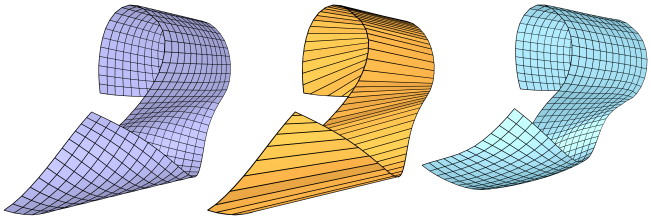


Fig. 3. Attempting to convert a quad mesh of a developable shape to a PQ mesh using a general-purpose planarization technique (ShapeUp [Bouaziz et al. 2012]) significantly alters the shape and makes it non-developable. This happens because the edges of the input mesh are generally not aligned to principal curvature directions. Our method is applied to a trivial triangulation of the input mesh. We report Hausdorff distance h with respect to bounding box diagonal and the maximal planarity error p.

for discrete meshes. In this work we exploit this specific constrained setting and the geometric structure of developable shapes to reproduce straight minimum-curvature lines, as well as automatically segment the input into curved and planar parts in a robust manner that is consistent with the structure dictated by developability.

Our method is based on fitting a scalar function on the input mesh, such that its isolines are straight and align as best as possible to the *locally* estimated rulings on non-planar regions (see Fig. 1). The condition that guarantees straight isolines on developable surfaces is simple to formulate: the normalized gradient of the scalar field needs to be divergence free. This nonlinear and high-order condition is numerically difficult to enforce in a straightforward manner. We therefore devise a dedicated optimization scheme that factors the problem into a divergence-free and integrable directional field optimization that is subsequently integrated into a scalar function. This makes our problem tractable and practical to optimize. We extract the isolines of the obtained scalar field at the desired resolution, and remesh the input into strips of planar quads whose chordal edges are the isolines, i.e., the rulings. We supplement the mesh by planar polygonal faces that represent the planar patches of the input surface. The flexibility of the field-to-function design allows for the automatic placement of singularities, flat regions and curved folds without explicitly segmenting different curvature regions on the mesh.

We demonstrate the effectiveness of our approach on a variety of input developable shapes represented by general triangle meshes (and triangulated quad meshes) and show how our remeshing can be used side-by-side with freeform modeling of developables.

2 RELATED WORK

Remeshing general meshes into (planar) quad meshes is an active area of research. A comprehensive review is beyond the scope of this paper, but we highlight the main features of existing approaches most closely related to our work.

As stated in the introduction, the quadrilaterals in freeform models of developable surfaces are usually non-planar, and typically

neither triangle nor quad meshes are curvature aligned. Our goal is to obtain a curvature-aligned remeshing with planar faces. Planarization of general polygonal meshes has been explored in several works [Alexa and Wardetzky 2011; Bouaziz et al. 2012; Diamanti et al. 2014; Poranne et al. 2013; Tang et al. 2014]. These methods take arbitrary shapes as input and are not specifically targeted at developable surfaces. Typically, applying a general planarization method to developable surfaces leads to poor results in terms of curvature alignment and shape approximation (see Fig. 3).

A different approach to obtaining PQ meshes from general developable input meshes is to utilize the fact that PQ meshes are a discrete model for conjugate nets and seek a remeshing that is aligned to ruling directions. Many curvature-aligned or just conjugate quad remeshing techniques for general shapes exist, see e.g. [Bommes et al. 2012; Diamanti et al. 2014; Jakob et al. 2015; Liu et al. 2011; Zadravec et al. 2010]. Similar to our method, these techniques rely on numerical estimation of the principal curvature directions, but they do not guarantee exact alignment or straight edge sequences and may introduce unnecessary singularities on developable shapes. Their optimization process might fail to create precise, straight rulings on developable surfaces, unlike the algorithm we propose in this work (see Fig. 5). From a fabrication viewpoint, a general PQ remeshing method is lacking since it merely produces a PQ mesh, rather than a PQ strip mesh, which allows for simpler construction.

A more promising approach to PQ meshing of developable shapes is a dedicated technique that utilizes their specific properties. Peternell [2004] converts a scan of a single torsal developable patch into a PQ mesh by thinning its tangent-space representation into a one-dimensional object (a simple curve). This approach is not immediately applicable to composite and possibly piecewise developable surfaces that consist of multiple torsal patches and planar regions. Kilian and colleagues [2008] compute a torsal patch decomposition for 3D scans of physical developable surfaces by estimating flat regions and ruling directions. This approach may struggle with developable meshes that are coarse in comparison to scans due to insufficient data density for reliable ruling fitting. Their method relies on the ruling estimates to initialize a planar mesh development, which is used in the subsequent optimization. The connectivity of this initial mesh cannot be changed during the optimization, and thus determines the approximation quality that can be obtained. Locally estimated rulings on developable meshes can be quite noisy and inaccurate, as we discuss in Sec. 4. We avoid a direct domain decomposition based on rulings employed in [Kilian et al. 2008] and instead devise a global constrained optimization approach. As a result, our method is successful on coarse and noisy inputs. Additionally, in contrast to [Kilian et al. 2008], our method can handle *piecewise* developable surfaces that are not curved folds constructed from a single sheet of material. Their method optimizes and deforms a surface to become developable as a whole, whereas our work takes a piecewise approach to remeshing the existing surface without deforming it. We compare our results to their work in Sec. 6.

Wang and colleagues [2019] use discrete parallel geodesic nets as a discrete model for developable surfaces. They require the geodesic strips to be of constant width for a surface to be developable, but do not impose any requirements on the directions of these strips and as such do not have a ruling-aligned representation for developables. They also use parallel geodesic nets to approximate surfaces by piecewise developable strips. The individual geodesic strips of a parallel geodesic net are approximated by piecewise planar faces in a postprocessing step, but this does not guarantee that compatibility between neighboring strips is preserved. In contrast, our method produces a complete, connected remeshing of the input developable surface with strips aligned to the rulings, rather than in the orthogonal direction.

While targeting geodesic fields, rather than planar quad remeshing, the works by Vekhter et al. [2019] and Pottmann et al. [2010] show parallels to our proposed method. They compute a unit curl-free field, while we compute a divergence-free field—these two kinds of fields are in fact duals. Nevertheless, our field has further constraints in terms of ruling alignment, which we take into account. In addition, our optimization strategy is different, interlacing integrability optimization with divergence reduction. We discuss this in further detail in Sec. 3.

3 BACKGROUND

We next summarize relevant facts about both continuous and discrete developable surfaces that inform our algorithm and offer a directional-field based definition of developable surfaces. We provide a discrete setup for these fields in Sec. 4, and an optimization scheme in Sec. 5.

3.1 Developable surface parameterization

A C^2-continuous surface $\mathcal{S}$ that has vanishing Gauss curvature everywhere is a smooth developable surface. A general developable comprises multiple developable patches $\{\mathcal{S}_i\}$, $\bigcup \mathcal{S}_i = \mathcal{S}$, where each such patch is either a *torsal* patch (a *curved* ruled surface with constant normal along each ruling) or a *planar* patch. The rulings are completely contained in each $\mathcal{S}_i$, i.e., they extend up to the boundary $\partial\mathcal{S}_i$ [Massey 1962]. The *planar patches* are regions with vanishing mean curvature $H = \kappa_2/2$, where κ_2 is the max (absolute) curvature. They are bounded by rulings of torsal patches and the boundary of the surface, as shown in Fig. 4.

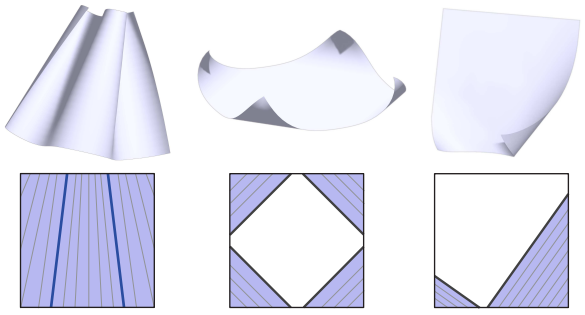


Fig. 4. Developable surfaces (top row) and their decompositions into planar and curved (torsal) patches, shown on the 2D development (bottom row). We display the planar patches in white and the curved patches in purple. The rulings are illustrated as thin grey lines, with the borders between curved and flat patches in thick black and inflection lines in blue.

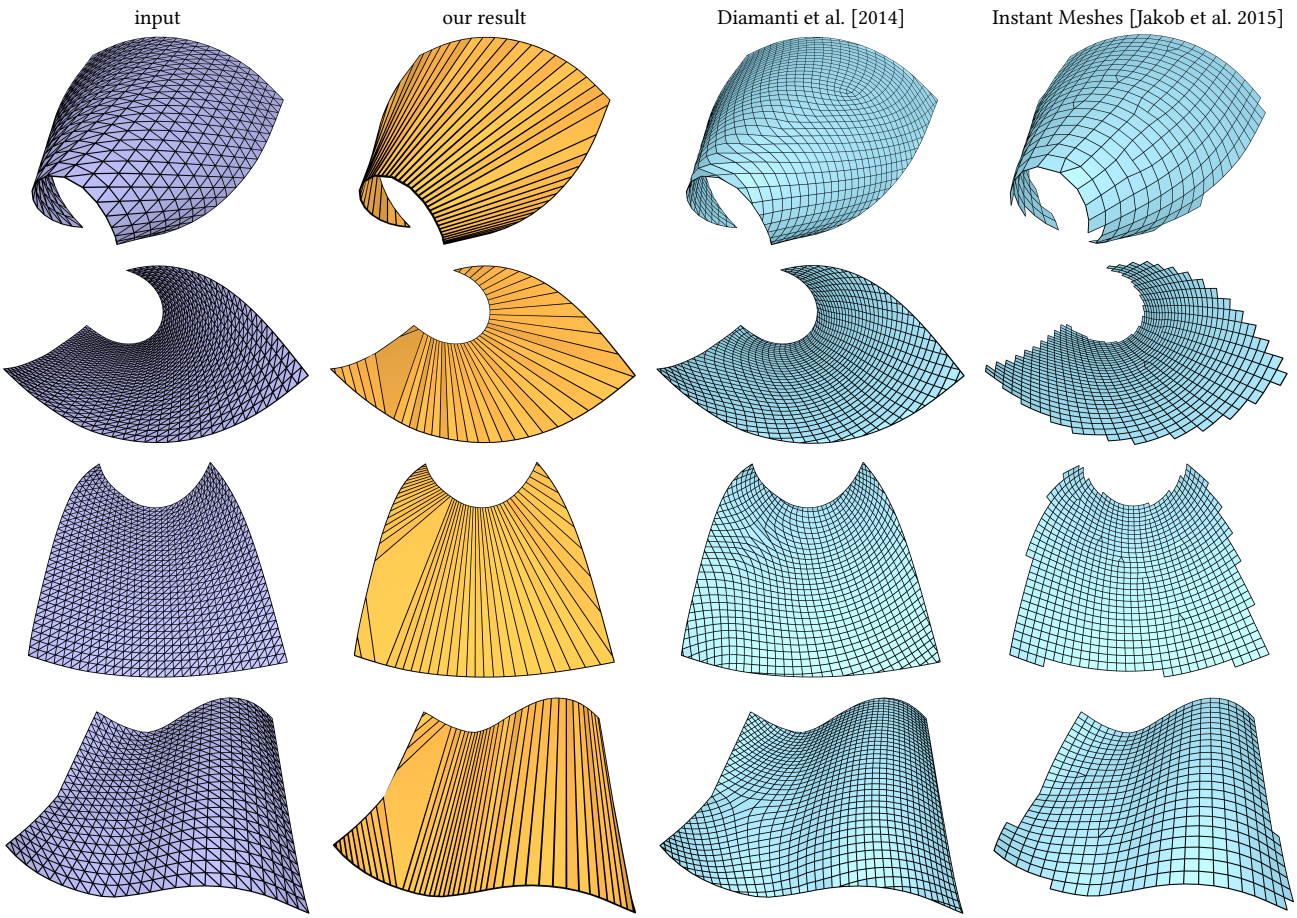


Fig. 5. Remeshing an input developable surface using the directional field design of [Diamanti et al. 2014] does not result in globally straight edge sequences. Instant Meshes, the curvature-aligned quad dominant remeshing technique of [Jakob et al. 2015], introduces superfluous singularities and does not always succeed in finding the exact rulings. For Instant Meshes we use the following settings: 4-RoSy extraction, quad-dominant mesh extraction, no boundary alignment (to ensure better curvature alignment; trimming can be done in a post-processing step).

Non-smooth developables. We also consider more general, piecewise developable surfaces. One type is *creased* shapes, where several smooth developable surfaces are joined along curves with only C^0-continuity [Huffman 1976]. These curves are termed *curved folds* when the surface is globally isometric to a planar domain (as in Fig. 22), and *creases* when this is not the case (e.g., Fig. 18). We treat curved folds and creases identically in the rest of this paper and refer to them as creases from now on. Another type is surfaces that contain *point singularities*, such as cone apexes (see Fig. 19). These surfaces are locally non-developable at the singularities; they can be constructed by gluing parts of the boundary of a developable surface together while allowing isometric deformation. The cone apexes are easily identified, and to run our method we remove the apex vertices that don't coincide with a crease together with their incident faces. If desired, they can be added back in post-processing. We note that our method requires that the individual pieces are sufficiently smooth, and allow a definition of rulings whose endpoints are always on boundaries or creases (as is the case for the surface types described above), therefore explicitly excluding surfaces that look like crumpled paper.

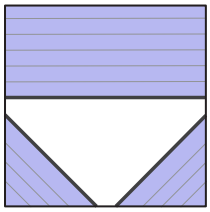
Conjugate nets. Consider a single torsal patch $\mathcal{S}_i$, where we parameterize the patch with coordinates $\mathcal{S}_i(u, w)$ as follows:

$$\mathcal{S}_i(u, w) = p(u) + w\, r(u), \tag{1}$$

where $p(u) : \mathbb{R} \to \mathbb{R}^3$ is a *generating curve,* and every u-isoline is a straight line with direction $r(u) : \mathbb{R} \to \mathbb{S}^2$, i.e., a ruling. The Gauss map $n(u, w)$ must be constant on the u-isolines in order for $\mathcal{S}_i$ to be developable: $n(u, w) = n(u)$. This means that the u-isolines on $\mathcal{S}_i$ are *extrinsically straight*; they constitute lines in $\mathbb{R}^3$.

The rulings are the minimum curvature lines of $\mathcal{S}_i$. The uw-parameterization constitutes a *conjugate net* [Liu et al. 2006]. In particular, choosing $p(u)$ to be a max curvature line (i.e., having the $p(u)$ curve intersect all rulings at right angles) makes $\mathcal{S}_i(u, w)$ a principal curvature line parameterization.

Seamless parameterization. When $\mathcal{S}$ is C^2-continuous, every torsal patch either borders a planar patch, or the outer boundary of $\mathcal{S}$. Planar patches allow to connect several different ruling strips, with multiple orientations, by introducing *singularities* within the planar patch (see inset). These planar patches cannot be ruled in a manner that is consistent with all adjacent torsal patches, and so u on a planar patch will be a smooth interpolation of the u function on the adjacent torsal patches (whilst locally also being subject to the constraints detailed in Sec. 3.2). Since the rulings are by definition 2-symmetric (i.e., they are invariant to the sign of $r(u)$), the singularities can be of indices $\pm\frac{1}{2}$, and the function u is only *seamless* rather than continuous on $\mathcal{S}$, similarly to the stripe patterns of [Knöppel et al. 2015]. Note however that u is locally fully continuous (it can be "combed") on each torsal patch $\mathcal{S}_i$. Across creases, rulings are only C^0-continuous, and thus we constrain u to be C^0 there as well. We further note that works such as [Diamanti et al. 2014; Jakob et al. 2015; Liu et al. 2011] intermix between the u and the w coordinates, and enable full quad-mesh $\pm\frac{1}{4}$ singularities (see Fig. 5). While this provides more meshing flexibility for general curved surfaces, it in fact hinders the ability to correctly capture the pure foliation topology of the ruling stripes comprising the developable surfaces.



3.2 Ruling fields

Our work focuses on designing directional fields that generate the rulings of a developable surface from other representations, and integrate them to compute u. Consider the gradient vector field ∇u, which is by definition orthogonal to the isolines of u. The geodesic curvature of isolines is defined as: $\kappa_g(u) = \nabla \cdot \frac{\nabla u}{\|\nabla u\|}$ [Sethian 1999]. Since the u-isolines of the uw-parameterization of a developable surface following Eq. (1) are extrinsically flat, we have $\forall u,\ \kappa_g(u) = 0$. As such, ruling fields are both geodesic and principal.

Denote by $r^\perp$ a unit-length vector field orthogonal to the ruling directions r in the tangent bundle of $\mathcal{S}$, such that $\frac{\nabla u}{\|\nabla u\|} = r^\perp$. Next, consider a unit length 2-vector field Y on a developable surface $\mathcal{S}$, which is the assignment of a tangent vector Y to every point $p \in \mathcal{S}$, and which is defined up to sign. If we align Y with $r^\perp$, we have by definition

$$\nabla u \parallel Y. \tag{2}$$

For simplicity we first consider the case where Y does not have singularities, and the surface $\mathcal{S}$ does not contain creases. This allows Y to be combed (i.e., the sign branching to be resolved) in every local surface patch by consistently choosing one of the two directions to obtain a smooth 1-vector field. Using this combed single-vector field version of Y we then get:

$$\nabla \cdot Y = \nabla \cdot \frac{\nabla u}{\|\nabla u\|} = 0. \tag{3}$$

This means that Y is a divergence-free unit vector field. Our objective is to design Y and integrate u from it, which leads to the question for which divergence-free unit fields Y such a u exists. Y must be *integrable* up to a multiplicative scalar. That is, there must exist a positive scalar function s, $s(p) > 0,\ \forall p \in \mathcal{S}$, for which (using the combed version of Y):

$$\nabla \times (sY) = 0. \tag{4}$$

The geometric meaning of the scalar function s is the *density* of the isolines of u at point p. A non-constant s comes up naturally when the isolines have a fan-like structure (for instance, the rulings of a cone).

Singularities and combing. We design Y as a 2-vector field, where it is only defined up to sign. Therefore, the divergence and curl operators do not automatically apply. Rather, in every local surface patch that does not contain singularities, Y can be *combed* by consistently choosing one of the two directions to obtain a smooth single-vector field on which we adhere to conditions (3) and (4).

Since our field Y is 2-symmetric, singularities have indices that are integer multiples of $\pm\frac{1}{2}$. As Y is a unit field, it is not defined there, and neither is u. As a consequence, it is not divergence free in any neighborhood that contains the singularity, and the isolines of u are not straight there (see Fig. 1). Note that singularities either arise on planar patches, or on cone apexes, and therefore do not compromise the properties of the field on torsal patches. Following the common paradigm of seamless parameterization (e.g., [Bommes et al. 2009; Diamanti et al. 2015]), this is the reason why we design the field sY as curl-free, rather than as the *conservative* ∇u, which is only locally defined in simply-connected non-singular patches.

Relation to geodesic fields. Vekhter et al. [2019] and Pottmann et al. [2010] both apply the unit-length divergence property to design geodesic fields; more precisely, Vekhter et al. [2019] work with the dual curl-free vector field $Y^\perp$ and define a similar integrability measure. Nevertheless, our work handles further challenges, as it is not enough to target geodesic fields to guarantee that they follow rulings, even though rulings are geodesics. It is in fact theoretically impossible to characterize rulings of a developable merely as geodesics, since they depend on the shape operator and are thus extrinsic. Therefore, Y has to be designed such that $Y^\perp$ aligns to prescribed rulings. As we see in Sec. 4, estimating and aligning to reliable rulings is a challenging task that must include completion in unreliable regions.

Ruling field at creases. Rulings on two developable patches adjacent to a crease typically do not form a single, intrinsically straight line, but rather meet at an angle (see e.g. Fig. 21, 22). We therefore do not require Y to be divergence-free near creases, effectively allowing the vector field to break across them. Furthermore, the ruling field does not have to be strictly divergence-free on planar regions to create planar quads, and it will not be divergence-free around singularities; nevertheless, we optimize for this property everywhere smooth, while letting singularities emerge naturally, to avoid identifying and segmenting these regions explicitly.

3.3 Discrete ruling-aligned developable meshes

A discrete sampling of the u-isolines of a principally-aligned parameterization creates a quadrilateral mesh whose faces are planar up to second order [Liu et al. 2006]. Anisotropic quadrilateral meshing aligned to principal directions is known to have optimality properties in terms of approximation quality (see e.g. [Alliez et al. 2003]).

These facts motivate curvature-aligned polygonal remeshing, in particular for fabrication purposes.

Since we only design and discretize u, leaving the coordinate w as a degree of freedom, our discretization for a torsal patch is that of a mesh comprising long planar polygons. These polygons are for the most part quadrilaterals whose edges are two boundary curves and two straight rulings; thus, a torsal patch is represented as a PQ-strip model. Planar patches are represented as big flat polygons, where the non-straight isolines are fully contained in the plane, and we are therefore allowed to straighten them out. The planar polygons are in general non-quad, since they may contain singularities determined by the number of torsal patches it connects; nevertheless, their planarity makes them easy to refine if required.

4 DISCRETIZATION

The input to our algorithm is a triangle mesh $\mathcal{M} = \{\mathcal{V}, \mathcal{E}, \mathcal{F}\}$ representing the (piecewise) developable surface, where $\mathcal{V}$ denotes the set of vertices, $\mathcal{E}$ the set of edges and $\mathcal{F}$ the faces. To regularize the scale of surface curvature between different surfaces, and our optimization parameters, we scale the input $\mathcal{M}$ to have unit-length bounding box diagonal. We define u as a piecewise-linear vertex-based function $u(v)$, $v \in \mathcal{V}$, and consequently represent r, $r^\perp$, and ∇u as face-based piecewise-constant tangent fields; we denote this space as $\mathcal{X}$. We use the conforming discrete gradient $G : (\mathcal{V} \to \mathbb{R}) \to \mathcal{X}$ and divergence $D : \mathcal{X} \to (\mathcal{V} \to \mathbb{R})$ operators, and the non-conforming discrete curl operator $C : \mathcal{X} \to (\mathcal{E} \to \mathbb{R})$. Their explicit expressions can be found in, e.g., [Brandt et al. 2017].

Estimating rulings. We compute a ruling direction $r(f)$, $\forall f \in \mathcal{F}$, as the eigenvector corresponding to the minimal eigenvalue of the face-based shape operator $S(f)$, as defined in [de Goes et al. 2020]. Since we know the ruling only up to sign, we represent it unambiguously using a *power representation* [Azencot et al. 2017; Knöppel et al. 2013]: we first represent $r(f)$ as a complex number in a local coordinate system and then square this complex number to have a representation that is invariant to the sign of the direction, i.e., we store $R(f) = r^2(f)$. We also define $R^\perp(f) = (r^\perp(f))^2$, the power representation of the ruling locally rotated by 90 degrees.

Confidence weights. A clean domain decomposition into planar and torsal regions would significantly simplify the fitting of individual developable patches. Unfortunately, we cannot obtain such a clean segmentation directly, because the curvature measure (like the ruling estimates) is noisy and does not delineate planar and torsal patches nicely (Fig. 6). Therefore, we model on the assumption that the rulings are least reliable in planar or near-planar regions, and mostly consistent in strongly curved areas (see Figs. 1 and 7). We thus attach a *relative confidence* weight $w(f)$ to each face $f \in \mathcal{F}$, as a function of the discrete absolute max and min curvatures $\kappa_1(f)$ and $\kappa_2(f)$:

$$w(f) = \theta_1 \left(1 - e^{\theta_2 (\kappa_1(f) - \kappa_2(f))^2}\right). \tag{5}$$

For $\kappa_1(f)$ and $\kappa_2(f)$ we use the absolute largest and smallest eigenvalues of the shape operator $\mathcal{S}(f)$, and set $\theta_1 = 0.8$, $\theta_2 = -0.014$.

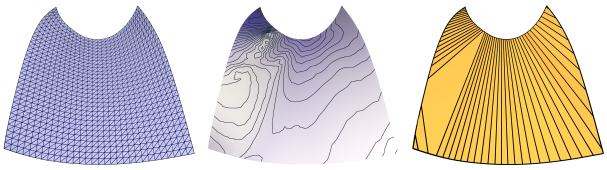
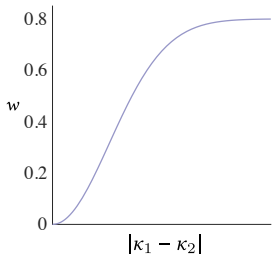


Fig. 6. The curvature isolines of $|\kappa_1 - \kappa_2|$ (middle) do not provide a clean delineation between torsal and planar parts. However, our method automatically places a planar polygon in the appropriate region, without being provided with an explicit decomposition (right).



The confidence function is a logistic curve (see inset), facilitating a stronger distinction between confidence in planar and near-planar regions (albeit still small compared to stronger curved regions). The value of $w(f)$ is capped at 0.8 by design, ensuring that we never fully rely on a ruling. We define $\mathcal{V}_b$ to be all vertices on the boundary of $\mathcal{M}$ and $\mathcal{F}_b$ all faces that contain a vertex in $\mathcal{V}_b$, and we set $w(f) = 0$ for these faces.

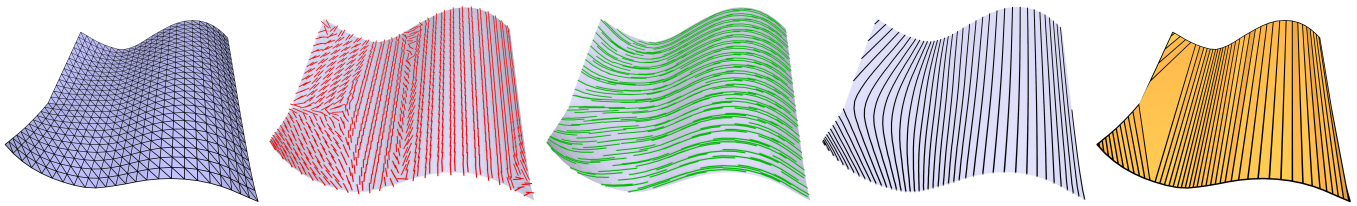
Creases. Our method requires as input the explicit identification of the set of crease edges $\mathcal{E}_c$ that define curved-fold creases and boundaries to developable pieces. We define $\mathcal{V}_c$ as all vertices that are incident on an edge in $\mathcal{E}_c$, and from this we define the set of faces adjacent to them: $\mathcal{F}_c$ is the set of all faces that have one or more vertices in $\mathcal{V}_c$. We update the confidence weights by setting $w(f) = 0$ for all faces in $\mathcal{F}_c$. Using $\mathcal{V}_c$ and $\mathcal{V}_b$ we initialize $\mathcal{V}^*$ as the set of mesh vertices but with the boundary and crease vertices excluded, i.e. : $\mathcal{V}^* = \mathcal{V} \setminus (\mathcal{V}_b \cup \mathcal{V}_c)$. The user can provide crease edges as input or we can make an initial guess for the crease edges based on the dihedral angle of adjacent faces and manually add edges belonging to softer creases. We require that the final set of crease edges divides the surface into smooth developable surfaces. The required seams for this segmentation are typically easily visually distinguishable (for example using reflection lines). Figures 18, 20, 21, 22 and 11 show examples of developable surfaces with creases and Fig. 20 explicitly shows them on a complex model. When creases end in the interior of a developable piece rather than on another crease or boundary, so called *open creases*, we duplicate their interior vertices and define them as mesh boundaries. Examples of such open creases can be seen in Figures 19, 20 and 11 (the chair model).

5 METHOD

We describe our approach to remeshing a (near-)developable input triangle mesh to a curvature-aligned, planar polygonal mesh consisting primarily of PQ strips.

5.1 Optimization problem

Setup. We compute the face-based shape operator and the ruling related quantities $r(f)$ and $r^\perp(f)$. Our computed field Y, and the ruling fields $r(f)$ and $r^\perp(f)$ are represented as either vectors in $\mathbb{C}^{|\mathcal{F}|}$, or the equivalent $\mathbb{R}^{2|\mathcal{F}|}$, defined in a local basis defined on



(a) input mesh $\mathcal{M}$ (b) input rulings r (c) streamlines of Y (d) isolines of u (e) our result $\mathcal{M}'$

Fig. 7. Our remeshing pipeline: (a) The original input $\mathcal{M}$; (b) the noisy input rulings r; (c) our computed Y field, visualized with streamlines; (d) the isolines of the optimized function u; (e) the final remeshing result $\mathcal{M}'$. Note how the isolines in (d) bend inside the planar region, which gets meshed as one large polygon, but are straight in the torsal regions, which result in PQ-strips.

each $f \in \mathcal{F}$; for simplicity we do not distinguish between them and it is made clear by the context. We further compute the quantities $R(f) = r^2(f)$, $R^\perp(f) = (r^\perp)^2(f), \forall f \in \mathcal{F}$, as well as the confidence weights $w(f)$, as detailed in Sec. 4.

We consider the diagonal face-based mass matrix either for vector-valued quantities $M_\mathcal{X} : 2|\mathcal{F}| \times 2|\mathcal{F}|$, or for scalar or complex quantities $\mathcal{M}_\mathcal{F} : |\mathcal{F}| \times |\mathcal{F}|$, holding the face areas $m(f)$. We further consider the edge-based mass matrix $M_\mathcal{E}$ holding edge masses

$$m(e) = \frac{\|e\|}{\|e_{\text{dual}}\|} \left(m(f) + m(g)\right)/2,$$

where $\|e_{\text{dual}}\|$ is defined as the summed length of the two dual edges from the midpoint of e to the barycenters of the adjacent faces f and g. Finally, for a vector field $Y \in \mathcal{X}$ we use the *integrated* discrete divergence $DY = G^\mathsf{T} M_\mathcal{X} Y$ $(\in \mathbb{R}^{|\mathcal{V}|})$, where G is the discrete gradient operator, which for a triangular face f consisting of vertices i, j, k and scalar function u is defined as $Gu_{ijk}(f) = \frac{1}{2m(f)}(e^\perp_{jk} u_i + e^\perp_{ki} u_j + e^\perp_{ij} u_k)$.

Objective. We optimize for a gradient ruling field $Y(f)$, according to the requirements of Sec. 3.2. That is, $Y(f)$ should have unit norm, it should align to the estimated rulings $r^\perp(f)$ up to sign and according to confidence, it should be divergence free away from creases, boundaries, and singularities, and it should be curl-free up to scaling. Our variables are then $Y(f)$ itself, its power representation $\Gamma(f) = Y^2(f)$, where $\Gamma(f)$ should align to the perpendicular power ruling field $R^\perp(f)$ according to the confidence $w(f)$, and where Y is divergence-free away from singularities. Furthermore, we optimize for a scalar field $s(f)$, such that $s(f) \cdot Y(f)$ is curl-free. Our objective breaks down to the following terms:

Alignment objective. Our alignment term is

$$E_a(\Gamma) = \sum_{f \in \mathcal{F}} m(f) w(f) \left\|\Gamma(f) - R^\perp(f)\right\|^2, \tag{6}$$

where $m(f)$ is the face area of f and $w(f)$ is the confidence weight as defined in Eq. (5). This can be formulated in matrix form as

$$E_a(\Gamma) = \left(\Gamma - R^\perp\right)^\mathsf{H} M_\mathcal{F} W_\mathcal{F} \left(\Gamma - R^\perp\right), \tag{7}$$

where $W_\mathcal{F}$ is the diagonal matrix of per-face confidences for complex numbers or scalars, and Γ and $R^\perp$ are arranged as $|\mathcal{F}| \times 1$ complex vectors. Note the *conjugate* transpose $\left(\Gamma - R^\perp\right)^\mathsf{H}$.

Unit-norm divergence-free objective. We ideally want the field to be perfectly divergence-free and have unit norm everywhere. However, this is impossible at singularities (Sec. 3.2) and in general would only be important on torsal patches. We follow [Viertel and Osting 2019] and [Sageman-Furnas et al. 2019] by employing a *Ginzburg-Landau* functional, introducing the following objective term (defined as $\epsilon \to 0$):

$$E_d(Y) = \sum_{v \in \mathcal{V}} \frac{1}{m(v)} |DY(v)|^2 + \frac{1}{\epsilon^2} \sum_{f \in \mathcal{F}} m(f) \Big(\|Y(f)\|^2 - 1 \Big), \tag{8}$$

where $m(v)$ is the barycentric Voronoi area of vertex v; note that its reciprocal is used since Dy is an integrated quantity. When $\epsilon \to 0$, this is analogous to minimizing the divergence of a unit-norm field after removing a ball of radius ϵ around singularities. Since the unit-norm divergence-free condition is satisfiable on torsal patches in a direction that matches with the alignment terms, singularities (if any) will naturally be located inside planar regions.

Smoothness regularizer. To encourage the field to smoothly transition from curved to planar parts, and in general to regularize low-confidence regions, we add a small smoothness term that encodes the smoothness of the power vector field across edges. For each interior edge e adjacent to faces f and g, the power smoothness [Knöppel et al. 2013] is measured as:

$$\|\Gamma(f)\, \bar{e}_f^2 - \Gamma(g)\, \bar{e}_g^2\|^2. \tag{9}$$

Here, $\bar{e}_f$ is the conjugate of e_f, which is the complex representation of the normalized edge vector e in the basis of f, and similarly for g. Our smoothness regularizer then becomes:

$$E_s(\Gamma) = \sum_{e \in \mathcal{E}} m(e)\, (1 - w(e))\, \|\Gamma(f)\, \bar{e}_f^2 - \Gamma(g)\, \bar{e}_g^2\|^2, \tag{10}$$

where $w(e) = (w(f) + w(g))/2$. In matrix form, we write this energy as $E_s(\Gamma) = \Gamma^\mathsf{H} L_2 \Gamma$, where

$$L_2 = G_\mathcal{E}^\mathsf{H} M_\mathcal{E} \left(I - W_\mathcal{E}\right) G_\mathcal{E}, \tag{11}$$

where $G_\mathcal{E}$ implements the differences $\Gamma(f)\, \bar{e}_f^2 - \Gamma(g)\, \bar{e}_g^2$ from Eq. (9).

Integrability. We use the discrete curl operator C to measure integrability of the (per-face) scaled field sY:

$$CsY(e) = \left\langle s(f)Y(f) - s(g)Y(g),\ e \right\rangle. \tag{12}$$

We constrain

$$CsY = 0. \tag{13}$$

To constrain s to be positive and prevent large density variations, we further bound

$$\forall f \in \mathcal{F},\ s_{\text{low}} < s(f) < s_{\text{high}}\,. \tag{14}$$

We provide the values used for s_{low} and s_{high} in Sec. 5.2.

Branching and singularities. Generating Γ from Y is well-defined. However, the inverse has a sign degree of freedom. We follow common practice by arbitrarily choosing a sign in each face, and relating Y values across faces by using *principal matching* [Diamanti et al. 2014]; in our context, this means we match vectors according to the smallest rotation angle. The curl and divergence operators are always understood to be defined with relation to the matching at every edge and vertex, with the exception of singularities, creases, and boundary vertices (where we do not optimize for divergence).

Full optimization problem. Our optimization problem can then be finally formulated as follows:

$$(\Gamma, Y, s) = \arg\min\ \omega_a E_a(\Gamma) + \omega_d E_d(Y) + \omega_s E_s(\Gamma),\quad s.t. \tag{15}$$

$$\forall f \in \mathcal{F},\ \Gamma(f) = Y^2(f), \tag{16}$$

$$CsY = 0, \tag{17}$$

$$\forall f \in \mathcal{F},\ s_{\text{low}} < s(f) < s_{\text{high}}\,. \tag{18}$$

Here, $\omega_a, \omega_d, \omega_s$ are scalar weights. Similar to [Sageman-Furnas et al. 2019], we seek solutions where $\frac{\omega_s}{\omega_d} \to 0$ and $\frac{\omega_s}{\omega_a} \to 0$ to allow the solution to converge to a divergence-free unit-norm field aligned to rulings away from planar regions and singularities.

5.2 Optimization algorithm

Our energy is nonlinear and its constraints use discrete variable quantities such as the matching. As the optimization problem is separable in the Γ, Y and s variables, we optimize for them in an alternating fashion, following the spirit of [Sageman-Furnas et al. 2019] and [Viertel and Osting 2019]. Our method proceeds as described in Algorithm 1.

ALGORITHM 1: Optimize for ruling field

Initialize $\Gamma^0 = R^\perp,\ k = 0, \mathcal{V}^* = \mathcal{V} \setminus (\mathcal{V}_b \cup \mathcal{V}_c)$
repeat
  $k \leftarrow k + 1$
  $\Gamma_a^k \leftarrow \texttt{ImplicitAlign}(\Gamma^{k-1})$
  $\Gamma_s^k \leftarrow \texttt{ImplicitSmooth}(\Gamma_a^k)$
  $\forall f \in \mathcal{F},\ \Gamma_u^k(f) \leftarrow \frac{\Gamma_s^k(f)}{\left\|\Gamma_s^k(f)\right\|}$
  $\left(Y_u^k, C^k, D^k, \mathcal{V}^*\right) \leftarrow \texttt{LocalRawRepresentation}(\Gamma_u^k)$
  $Y_d^k \leftarrow \texttt{ProjectDivFree}(Y_u^k, D^k, \mathcal{V}^*)$
  $Y_c^k \leftarrow \texttt{ProjectCurlFree}(Y_d^k, C^k)$
  $\Gamma^k \leftarrow \texttt{PowerRepresentation}(Y_c^k)$
until $\max_f \|\Gamma^k(f) - \Gamma^{k-1}(f)\| < 10^{-3}$;

The function $\texttt{ImplicitAlign}(\Gamma^{k-1})$ reduces the alignment energy E_a by a single implicit Euler step, by solving the following linear system:

$$\left(I + \frac{\omega_a}{\mu_a} W_{\mathcal{X}}\right) \Gamma_a^k = \Gamma^{k-1} + \frac{\omega_a}{\mu_a} W_{\mathcal{F}} R^\perp. \tag{19}$$

The implicit step size ω_a is scaled by μ_a, which is the lowest nonzero eigenvalue of $W_{\mathcal{F}}$. Note that the mass matrix $M_{\mathcal{F}}$ is cancelled out in the gradient and eigenvalue. Similarly, $\texttt{ImplicitSmooth}(\Gamma_a^k)$ solves the following linear system:

$$(M_{\mathcal{F}} + \frac{\omega_s}{\mu_s} L_2)\Gamma_s^{k'} = M_{\mathcal{F}} \Gamma_a^k\,. \tag{20}$$

with the lowest nonzero eigenvalue μ_s so that $\exists x \neq 0,\ L_2 x = \mu_s M_{\mathcal{F}} x$. The step size ω_a is fixed to 0.1, and the step size ω_s starts as 0.005 and is halved every 30 iterations, to ensure that the alternation with the renormalization of Γ converges.

After normalizing the current vector field, it is transformed into the "raw" representation Y, where the principal matching (and consequently, the singularities) are computed, and from them the curl matrix C and divergence matrix D are updated. Furthermore, this function updates $\mathcal{V}^*$ according to the current singularities. Note that the sets of boundary vertices $\mathcal{V}_b$ and crease vertices $\mathcal{V}_c$ (Sec. 4) are always mutually exclusive with $\mathcal{V}^*$.

Next, $\texttt{ProjectDivFree}(Y_u^k)$ finds the closest divergence-free solution to Y_u^k by solving the following linear system:

$$\arg\min_{Y_d^k} \|Y_d^k - Y_u^k\|^2 \ \text{ s.t. } \ DY_d^k\left(\mathcal{V}^*\right) = 0. \tag{21}$$

Specifically we do this by solving

$$\arg\min_{x^*} \|x^*\| \ \text{ s.t. } \ Dx^*\left(\mathcal{V}^*\right) = -DY_u^k\left(\mathcal{V}^*\right),$$

where $x^* = Y_d^k - Y_u^k$. For x^* to be a minimum-norm solution adhering to the constraints, it should be expressible as $x^* = D^\mathsf{T} w$ for some w. So we can solve $DD^\mathsf{T} w\left(\mathcal{V}^*\right) = -DY_u^k\left(\mathcal{V}^*\right)$, set $x^* = D^\mathsf{T} w$ and finally obtain the divergence-free solution as $x^* + Y_u^k$.

Finally, the function $\texttt{ProjectCurlFree}(Y_d^k)$ finds the closest scaled curl-free solution by solving the following convex system:

$$\arg\min_{Y_c^k,\, s} \|Y_c^k - sY_d^k\|^2, \tag{22}$$

$$s.t.\ CY_c^k = 0, \tag{23}$$

$$0.4 \le s \le 1.6. \tag{24}$$

Although we have no guarantee of convergence (as discussed in Sec. 6), in our experiments it typically takes 50 iterations of our optimization algorithm or less for Γ to converge.

5.3 Vector field integration and meshing

Having an integrable Y, we use a mixed-integer integration scheme [Bommes et al. 2009] to obtain a seamless globally smooth parameterization which produces u. The input triangle mesh is cut into a topological disc, where the singularities are on the boundary, and then a corner-based u function is extracted, which is seamless across the cuts, using integer translations. We configure the integrator to produce $u \in \mathbb{Z} + \frac{1}{2}$ values at singular vertices, since then the integer isolines avoid meeting at these singularities, and we obtain a single polygon around the singularity.

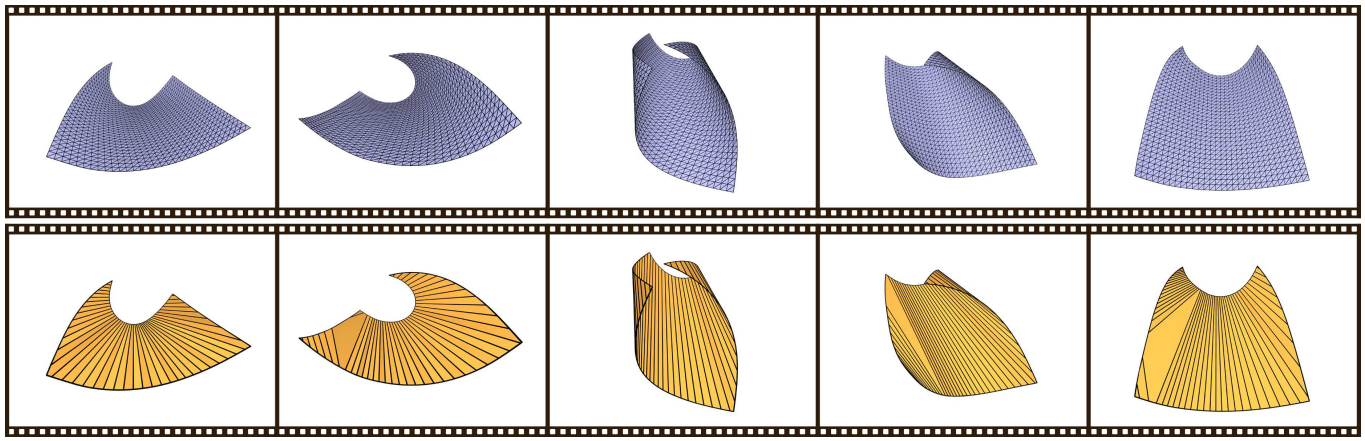


Fig. 8. Several snapshots from an interactive editing session (top row). The user deforms the DOG model by interacting with point handles at some selected vertices. At any time during the interactive session, the user may pause and invoke our remeshing algorithm and view the curvature-aligned remesh (bottom row). Note how the combinatorial structure of the ruled remeshing automatically changes to accommodate the changes in the surface geometry, without forcing the user to specify the patch decomposition manually.

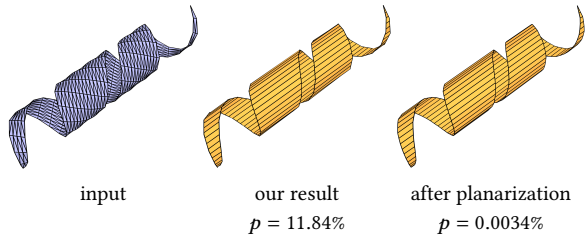


Fig. 9. Our result from Fig. 24 is planarized using ShapeUp [Bouaziz et al. 2012], achieving maximal face planarity error of $p = 0.0034\%$, compared with $p = 11.84\%$ in our initial result. The visual difference between the results is negligible. The Hausdorff distances are reported in Table 3.

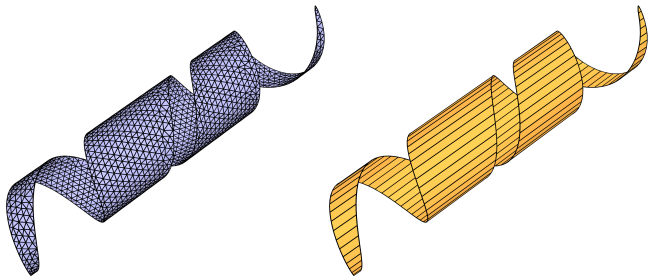


Fig. 10. Our result on a better tessellation of the input from Fig. 9 has maximal planarity error of $p = 2.58\%$, compared with $p = 11.84\%$ initially.

To create the final PQ-strip mesh, we trace the integer isolines of u (at a user-specified global resolution) and then collapse all valence-2 vertices that are not on the boundary, thereby removing all interior vertices. This effectively straightens the polylines of the isolines, which has little effect in torsal regions, since the isolines are already almost straight by the optimization (see Fig. 1 and Fig. 7). However, the isolines in planar regions, which might be more curved if a singularity causes the divergence-free constraint to be excluded, become chords between boundary vertices. We note that since we have a full parameterization of the surface, control over individual panel width is also possible by specifying a custom set of u values for tracing the isolines. We illustrate our remeshing pipeline in Fig. 7.

6 RESULTS AND DISCUSSION

We implemented our algorithm using libigl [Jacobson et al. 2018] and Directional [Vaxman et al. 2017a] on a machine with i7-8569U CPU and 16 GB RAM. Our typical input mesh resolution is 1800 faces, and for this approximate input size the vector field design part of our method takes 4–5 seconds, of which the majority of the time is spent in the `ProjectCurlFree` step, i.e., solving the convex optimization Eq. (22)-(24). We currently use CVX [Grant and Boyd 2014], but this part can be optimized for better speed. Although we do not have a formal convergence guarantee for our alternating algorithm, we observe that it typically converges to our specified tolerance level within 10–20 iterations for vector fields without singularities and 40–50 iterations for shapes with planar parts that introduce singularities in the vector field. We also test our method on inputs of up to 160k faces, which does not cause problems for convergence. The parameterization part of our method takes approximately 10–15 seconds.

A variety of our results can be seen in Fig. 24. Note that our method preserves the input boundary vertices, and therefore our output faces are quadrilateral-like higher degree polygons, rather than actual quadrilaterals. Examples of our results with various boundary shapes and non-disk topologies are included in Fig. 24. Our method is applicable to developables with curved folds, as seen in Fig. 1, 2 and 22 (input models from [Rabinovich et al. 2019]) as well as in Fig. 11. It can handle piecewise developable shapes, such as D-forms (shapes obtained by gluing together two planar

domains with the same perimeter) from [Jiang et al. 2020] and sphericons from [Tang et al. 2016], see Fig. 18, as well as other shapes with creases from [Tang et al. 2016], see Fig. 21. Works by Stein et al. [2018], Sellán et al. [2020], Ion et al. [2020] and others produce piecewise developable approximations, but they are not PQ meshes. Our work can be used to remesh those surfaces, see Fig. 20 for an example. We successfully apply our method to glued constructions from [Jiang et al. 2020], including point singularities, see Fig. 19. We have physically fabricated some of our results, shown in Fig. 23. Table 3 lists the most important statistics about our results.

Developable surface editing with dynamic connectivity. To demonstrate the utility of our approach, we use the point handle-based editing system of [Rabinovich et al. 2018] to interactively deform an input discrete orthogonal geodesic net (DOG) and create a sequence of a few developable surfaces, on which we run our algorithm (offline) after trivial triangulation. See Fig. 8 and the accompanying video for some examples of such editing sessions. Note the natural change in the combinatorics that our algorithm induces to model exact developability, which can change considerably even for small deformations in the input.

Planarity evaluation. Since our output meshes have no interior vertices inside the developable patches, the ultimate accuracy measure for the developability of our results is the planarity of the mesh faces. We measure planarity of each quadrilateral face by the ratio of the distance between the diagonals to their average length, in percent [Liu et al. 2006]. For higher-degree polygons, we compute the root-mean-square (RMS) error of all quads constructed from every 4 consecutive vertices in the polygon. An acceptable stringent tolerance for the planarity error is $\leq 1\%$ [Vaxman et al. 2017b]. It is generally not expected for parameterization based methods to achieve planarity to more than first order, so that usually further planarization post-processing is needed. We show the raw maximum and mean planarity error values of our results without any post-processing in Table 3. Even though our output meshes are quite coarse, our planarity errors are typically very low without such planarity optimization, and very close to the tolerance. Our worst maximum planarity error is obtained on a mesh with very thin features (Fig. 24, 5^{th} row, middle column), and the output quad with this maximal planarity error is towards the end of the spiral. As can be seen in Fig. 24 and Fig. 10, the input triangulation is very coarse there. We planarize this example, which has the worst maximum planarity error ($p = 11.84\%$) to zero planarity ($p = 0.0034\%$) using ShapeUp [Bouaziz et al. 2012], and reach a visually highly similar result, see Fig. 9. This demonstrates the capability of our algorithm to utilize the information in the original mesh effectively. Interestingly, the triangulation of the input mesh is similar to a Schwartz lantern, and remeshing this input to a more uniform triangulation already drastically brings down the maximal planarity error of our algorithm to $p = 2.58\%$, see Fig. 10.

Constraints and objective values. As described in Sec. 5.2, we have an iterative multi-step optimization algorithm. Because of its alternating nature, we can only guarantee that the converged solution will meet the curl-free constraint that is optimized for in the final step. We see that the divergence of the final (normalized) vector field is not fully zero everywhere, but it is nevertheless very low (see Table 1). A plot of the divergence values over the mesh shows that it mostly concentrates on planar regions (see inset). As the divergence is close to zero in torsal regions, the resulting function isolines are very straight and similar to their simplified version there. The simplified isolines in planar regions possibly deviate more, but since any straight line in a planar region is a ruling this does not pose a problem. Our final converged vector field may be far from unit-norm but this does not raise any issues as long as the divergence of its normalized version is zero.

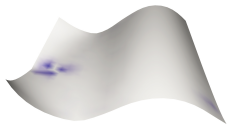


Table 1. Maximal and mean absolute divergence for converged vector fields. The reported absolute divergence values are computed on the final normalized fields and exclude values at singularities and boundaries.

mesh	max divergence	mean divergence
Fig. 7	0.0105	0.00046
Fig. 8 left	0.0032	0.00004
Fig. 16 bottom	0.0102	0.00050

Comparison with state of the art. In Fig. 11 we compare our results with the work by Kilian et al. [2008] through a visual comparison between (triangulated versions of) their output meshes and our outputs obtained from an isotropic remeshing of their output as our input. Because we require the input surface to already be developable, our method does not work directly on the surface scans that they use as input. We generated our input by explicitly preserving the crease edges and isotropically remeshing the patches in between. For the triangulated output meshes of Kilian et al. we determined a lower bound on the maximal and mean planarity errors by taking the global maximum and mean of the minimal planarity values attainable by pairing each face with each of its neighbors. We obtain visually similar results and also obtain comparable planarity error measures, whilst requiring less user input and manual tweaking. Specifically we do not require the user to ensure that the initial mesh connectivity is a valid developable decomposition.

Effect of output resolution. We extract u isolines of varying isovalues to create output meshes of different resolutions; we then measure their planarity and approximation quality w.r.t. the input mesh in terms of Hausdorff distance (Fig. 12). We note that the approximation quality and the planarity improve with higher resolution, although even for the coarsest resolution these metrics are already below tolerance.

Comparison with analytical principal-curvature directions. We test our method on an input mesh sampled from an analytical clothoid surface with varying resolution and compare the obtained vector field Y with the analytical max curvature directions. See Fig. 13 and Table 2. Note that the input to our method are *numerically estimated* ruling directions r, not their analytical values. As the data shows, upon refinement of the input mesh, our output field converges towards the analytical solution.

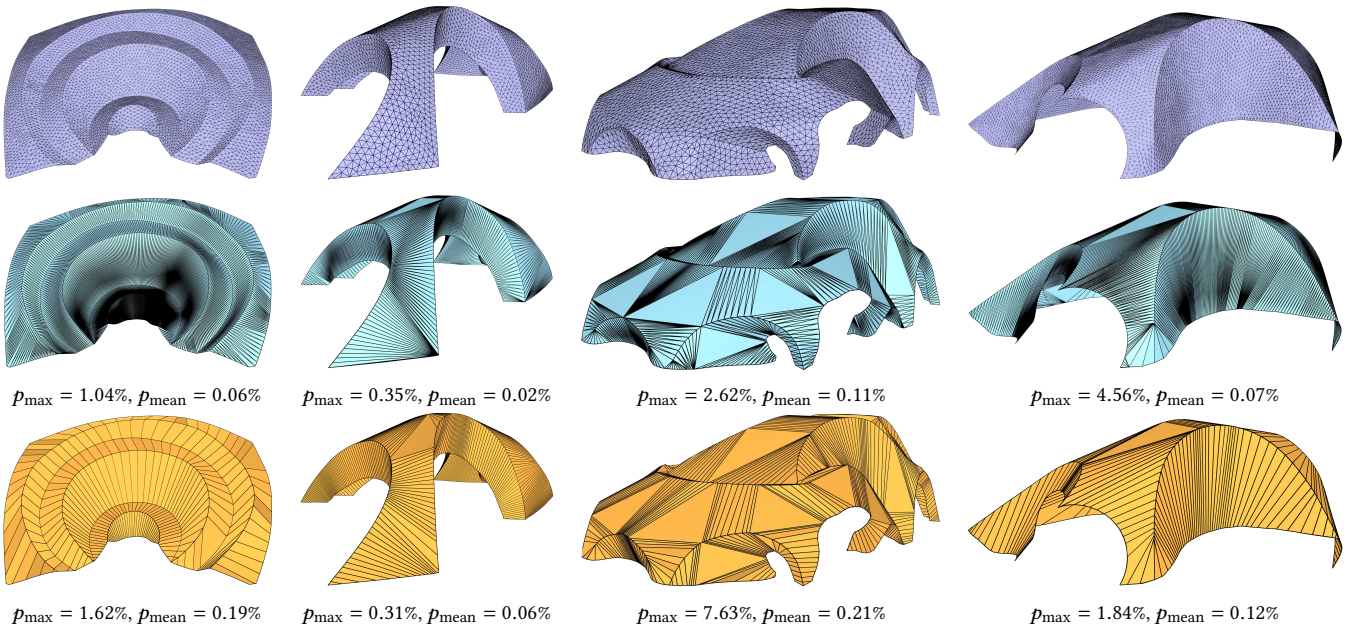


Fig. 11. We compare our results (bottom row) with those of [Kilian et al. 2008] (middle row, polygons are triangulated). As our method requires an input that is already developable, we used an isotropic remeshing of the output produced by Kilian et al. [2008] as input (top row). The car model (third from left) is designed by Gregory Epps. Note that this output is a higher resolution version of the one in Fig. 17.

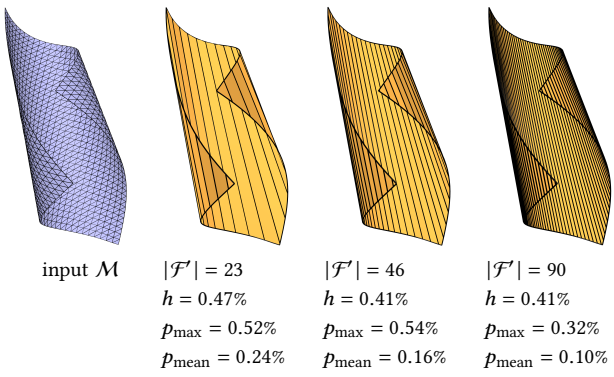


Fig. 12. Sampling the isolines of our optimized function u with increasing density leads to finer remeshing of the input mesh, where the Hausdorff distance to the input h, as well as the maximal and mean polygon planarity error, p_{max} and p_{mean}, decrease. The output resolution is denoted by the number of faces $|\mathcal{F}'|$. The Hausdorff distance is reported relative to the bounding box diagonal.

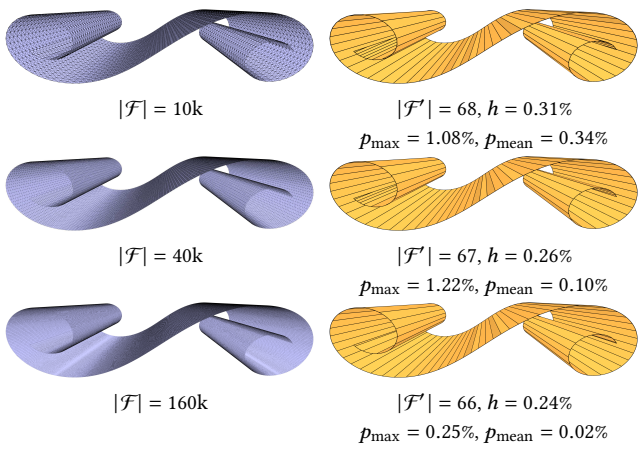


Fig. 13. As the input resolution $|\mathcal{F}|$ of a sampled analytical developable surface increases, the approximation accuracy and the planarity of our remeshed result increase. The meshing direction also aligns better with the mesh boundaries that coincide with analytical ruling directions in this case as the resolution increases.

Robustness. Our method is robust with respect to the parameters ω_a and ω_s, for which there is a range of values for that leads to visually very similar results. As the relative weight of ω_s with respect to ω_a increases, the vector field turns into a more constant field, reducing alignment quality of the final output mesh. Because of the fact that the smoothing step is the last one before the vector field is projected onto the divergence free solution space, ω_s is the dominating parameter in determining the rough layout of the final vector field. The parameter ω_a (within reasonable range) has a smaller contribution but can affect the exact placement of singularities and the iterations required to reach convergence (see Fig. 14). For noisy inputs, as in Fig. 15, our method does not converge with our standard parameter settings, or it converges but generates a vector field with a large amount of singularities. For these cases, simply increasing ω_s ensures that the optimization converges, although some small and noisy details may be lost (in Fig. 15 (right)

Table 2. Angular Difference (in degrees) between our optimized vector field and the analytical principal curvature directions on the clothoid mesh shown in Fig. 13.

$\lvert\mathcal{F}\rvert$	max °	mean °
10k	9.49	2.24
40k	4.80	1.19
160k	2.31	0.52

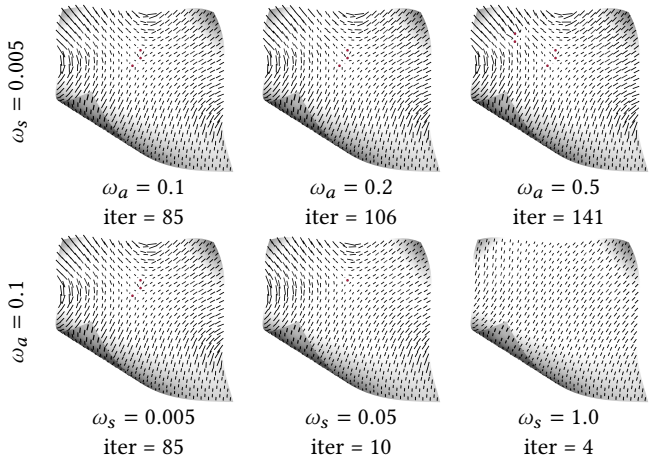


Fig. 14. A reasonable value for ω_s (top row) leads to visually similar results under varying ω_a. High values for ω_a cause the vector field to follow the noisy ruling evaluations in the central planar region, generating more singularities and increasing the iterations required to reach convergence. A too high value for ω_s (bottom row) causes over-smoothing of the field and overrules the alignment to the ruling estimates. The left column demonstrates our standard parameters.

we use $\omega_s = 0.15$). This shows that our method with the help of the smoothness term manages to recover a principally-aligned vector field even if the information from the input ruling directions is very weak. For optimal alignment the value of ω_s should be chosen as small as possible but as high as necessary so to not introduce excessive singularities; e.g., for the cone in the second to last row of Fig. 24 we use $\omega_s = 0.00005$ to emphasize better alignment near the boundary.

Limitations. Our method is not entirely triangulation independent, as shown in Fig. 16. If the input meshing does not allow for an accurate estimation of the principal curvature directions, this might lead to poor ruling estimation and diminished performance of our algorithm in terms of planarity error. This is most noticeable near the corners of the given input, where there is relatively little data for our algorithm to align to. In order to minimize bias introduced by the triangulation, it is advisable to triangulate polygonal input meshes with higher valences by inserting a new vertex at the face center and connecting it to the vertices of the original polygon in a triangle fan.

Furthermore, as we treat curved folds in identical manner to creases (namely we assume they delineate a developable surface

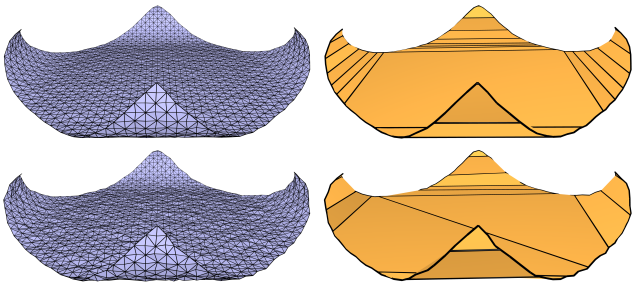


Fig. 15. Our method is robust to noise on developable inputs. These examples show our fourth example from Fig. 16, but with random vertex displacements applied. Left: a displacement of maximally 12.5% of the average edge length is applied, right: maximally 25% of the average edge length. Our method still recovers a meshing that is compatible with the original principal directions.

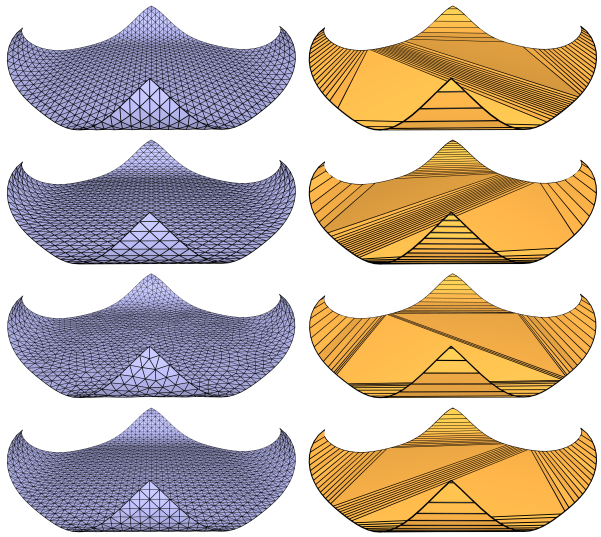


Fig. 16. Different triangulations of a quad mesh lead to different remeshing results, mainly in the near-planar regions. Nevertheless, all of the resulting meshing directions on the planar region are valid.

piece), there is no guarantee that the curved folded surface as a whole remains developable in one piece. An interesting direction for future work would be to incorporate known geometrical constraints at curved folds into our method.

If a cone apex is present in the mesh (see Sec. 3.1), the natural behaviour encouraged by our algorithm is to place a singularity on the crease to compensate for the curvature of the seam (see the cylinder example in Fig. 21 and Fig. 17, bottom). Nevertheless, our algorithm may fail to put the singularity exactly on the seam, depending on discretization. As a result, our optimization might get stuck, oscillating between nearby solutions with different singularity configurations (Fig. 17). Even though our method nominally fails to converge for the car example in Fig. 17, a reasonable result close to the expected one is still obtained.

Our method may struggle with thin features, e.g., as part of a piecewise developable, as these can often provide no alignment information at all. In the future it would be interesting to see how the vector field on surrounding developable pieces can be used to add constraints to these thin features, since in the final meshing we wish to guarantee continuity throughout the pieces. As shown in Fig. 13, our output quality with respect to Hausdorff distance, as well as mean and maximal planarity error increases as the input resolution increases. Our method is therefore dependent on the input resolution but still performs well on low resolution inputs.

Finally, we have no theoretical guarantees that the ruling-aligned edges in our output mesh do not intersect, although we never see this happen in our experiments. In torsal regions, the guiding field discourages overlapping behavior, as rulings on a developable are ordered. For planar regions, the guiding field contains more noise, but here the smoothness requirement for the vector field (and thus the corresponding parameterization) strongly discourages crossovers.

7 CONCLUSION

We presented an algorithm that converts developable surfaces represented by triangle meshes to a discrete curvature line parameterization, i.e., polygonal meshes with planar faces, where all interior edges correspond to rulings. This conversion from an unstructured triangulation of a developable surface to a curvature-aligned PQ mesh is an important step in the developable modeling and fabrication pipeline, for which thus far a robust practical solution was missing. Our method can only be expected to work well for nearly developable input shapes. We show a non-developable example in the inset, where the edge simplification step is omitted as it would degenerate the mesh. It would be interesting to see

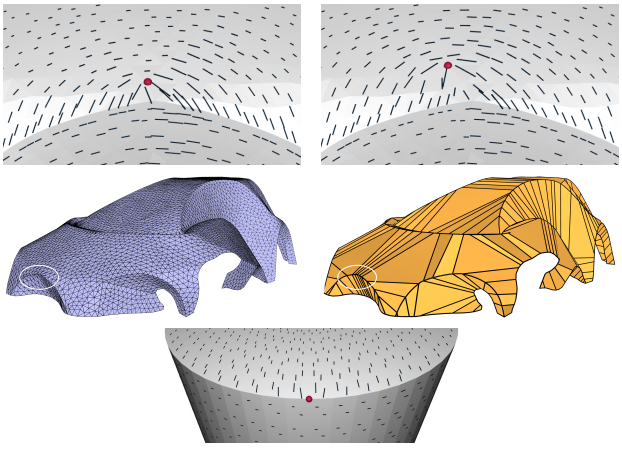


Fig. 17. When a cone apex of non-trivial curvature is present in the input mesh, our method might get stuck oscillating between solutions with different singularity configurations. The top row shows our method struggling to put the singularity exactly on the crease and oscillating between two solutions for the circled part of the car model (courtesy of [Kilian et al. 2008], designed by Gregory Epps). The middle row shows the input mesh and our obtained result. In comparison, the bottom row shows a successfully placed singularity on the crease of the cylinder model from Fig. 21.

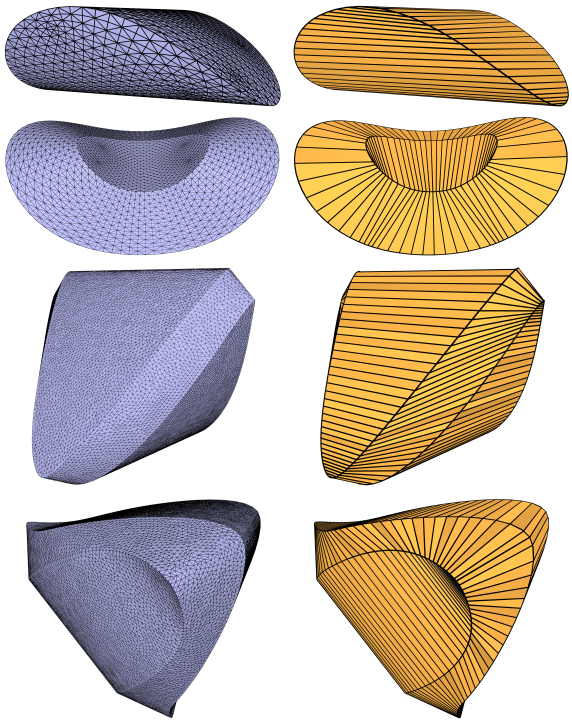


Fig. 18. Sphericons and D-forms are piecewise developable surfaces with creases connecting the individual pieces, and therefore can be remeshed with our method. The top two models are courtesy of [Jiang et al. 2020], the bottom two are courtesy of [Tang et al. 2016].

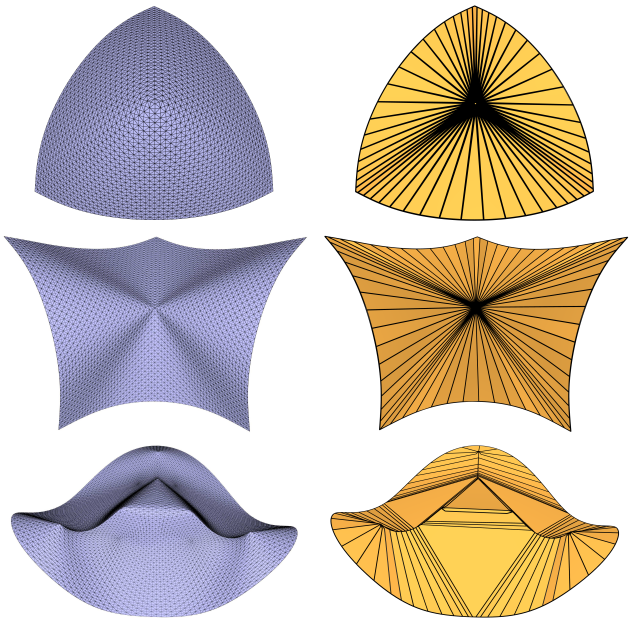


Fig. 19. Our method applied to glued developable surfaces that include cone apexes. The bottom model contains open creases that end in cone apexes. These models are courtesy of [Jiang et al. 2020].

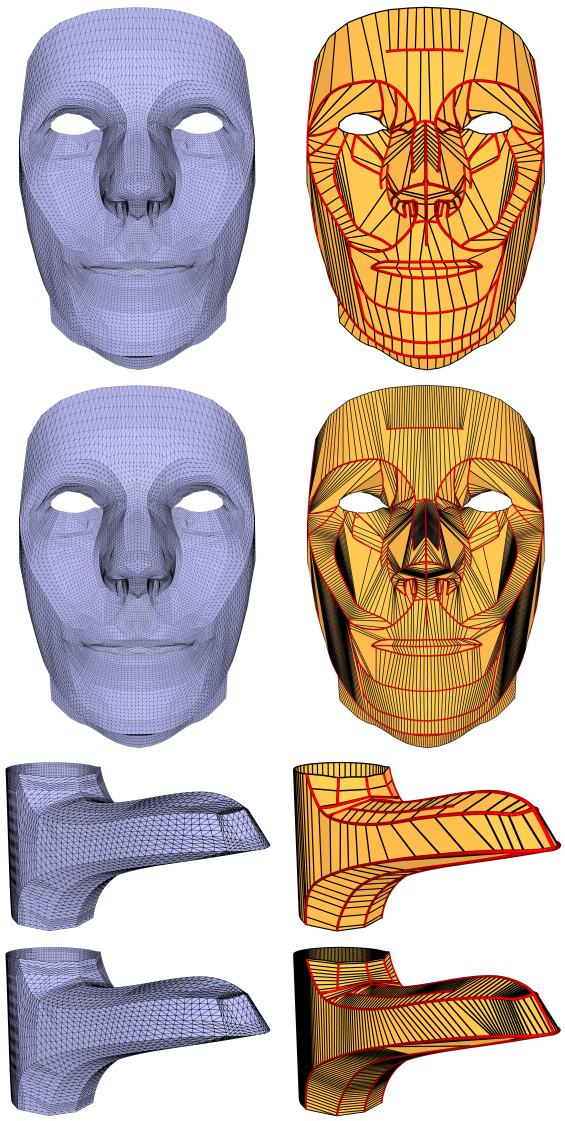
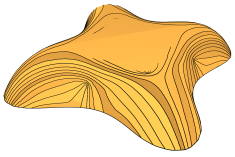


Fig. 20. Our method can handle open creases when they are defined as mesh boundaries. Examples of open creases can be seen on the forehead of the mask or on the top of the faucet. Since these creases are defined as mesh boundaries, seamless parameterization and meshing across them is no longer guaranteed. The red lines highlight the set of crease edges $\mathcal{E}_c$. The models are courtesy of [Stein et al. 2018].

how we can adapt our remeshing algorithm to be usable for the approximation of non-developable surfaces by developable patches. Another venue for further study would be the automatic tuning of the values ω_s and ω_a based on the noise levels of the estimated input rulings. Finally, it is conceivable to adapt the output mesh resolution based on the local curvature, allowing a denser representation in more curved areas.



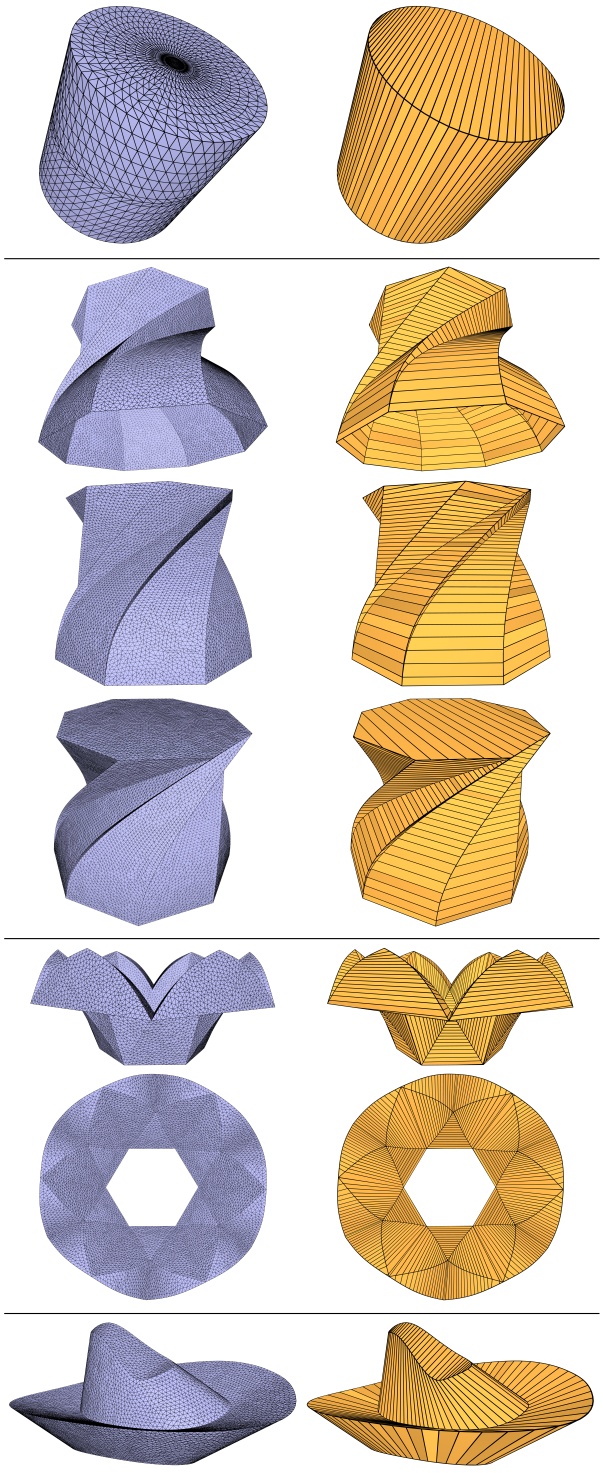


Fig. 21. Our method can handle piecewise developable surfaces with or without boundary and of different genera. These models are courtesy of [Tang et al. 2016].

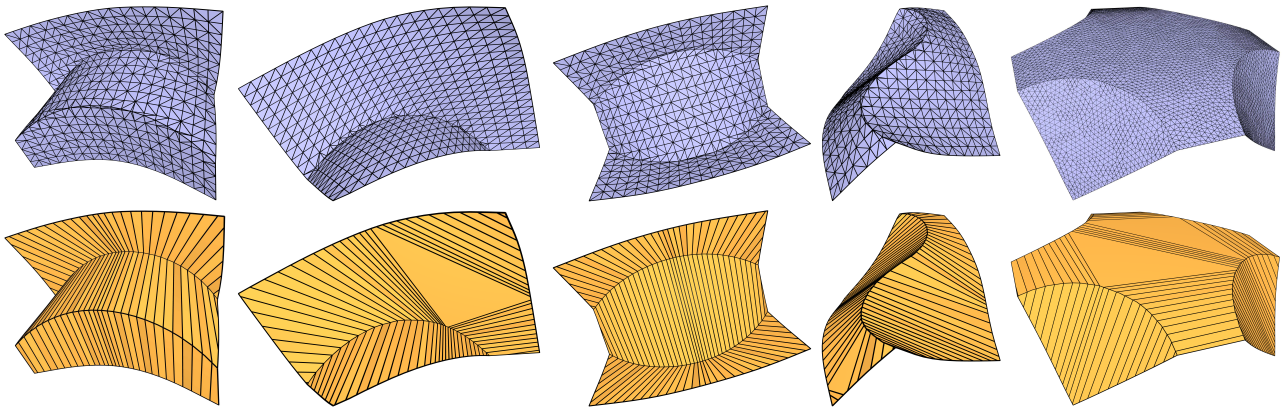


Fig. 22. For surfaces with curved folds our method produces meshes with faces that align well along the folds. Models courtesy of [Rabinovich et al. 2019].

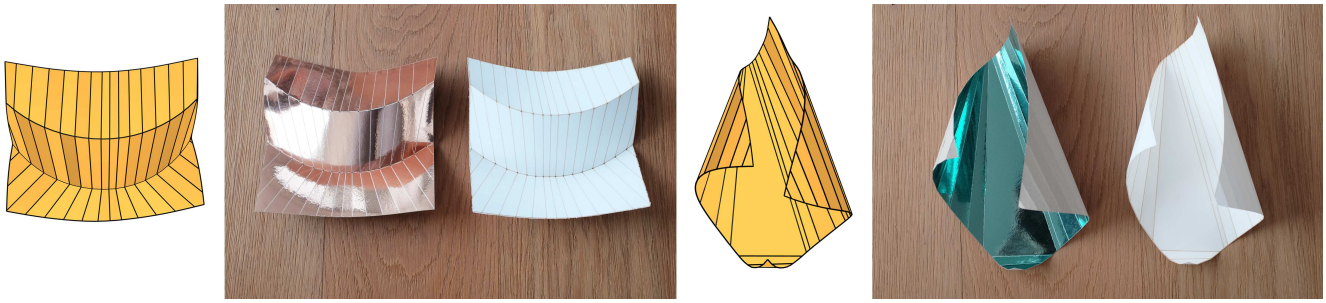


Fig. 23. Our output meshes can be physically fabricated from planar sheets of stiff material. For this experiment, we parameterize our output mesh to the plane and etch the flattened mesh edges into cardboard using a laser cutter. Appropriately bending the sheet of cardboard along the edges then gives a shape that matches our output.

ACKNOWLEDGMENTS

The authors are grateful to Helmut Pottmann, Michael Rabinovich and Alexander Sorkine-Hornung for illuminating discussions and guidance, and to Kaan Baki for help with video production. We also thank the authors of [Kilian et al. 2008], [Tang et al. 2016], [Stein et al. 2018] and [Jiang et al. 2020] for providing us with models from their works. This work is supported in part by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101003104) and by the Deutsche Forschungsgemeinschaft-Collaborative Research Center, TRR 109, "Discretization in Geometry and Dynamics".

REFERENCES

Quaglino Alessio. 2012. *Membrane locking in discrete shell theories.* Ph.D. Dissertation. Niedersächsische Staats-und Universitätsbibliothek Göttingen.

Marc Alexa and Max Wardetzky. 2011. Discrete Laplacians on General Polygonal Meshes. *ACM Trans. Graph.* 30, 4, Article 102 (July 2011), 10 pages. https://doi.org/10.1145/2010324.1964997

Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. 2003. Anisotropic Polygonal Remeshing. *ACM Trans. Graph.* 22, 3 (July 2003), 485–493. https://doi.org/10.1145/882262.882296

Omri Azencot, Etienne Corman, Mirela Ben-Chen, and Maks Ovsjanikov. 2017. Consistent functional cross field design for mesh quadrangulation. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

Alexandre Binninger, Floor Verhoeven, Philipp Herholz, and Olga Sorkine-Hornung. 2021. Developable Approximation via Gauss Image Thinning. *Computer Graphics Forum (proceedings of SGP 2021)* 40, 5 (2021), 289–300. https://doi.org/10.1111/cgf.14374

D. Bommes, B. Lévy, N. Pietroni, E. Puppo, C. Silva, M. Tarini, and D. Zorin. 2012. State of the Art in Quad Meshing. In *Proc. EUROGRAPHICS, State-of-the-Art Reports (STARs).*

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-Integer Quadrangulation. *ACM Trans. Graph.* 28, 3, Article 77 (2009), 10 pages. https://doi.org/10.1145/1531326.1531383

Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. 2012. Shape-up: Shaping discrete geometry with projections. *Comput. Graph. Forum* 31, 5 (2012), 1657–1667.

Christopher Brandt, Leonardo Scandolo, Elmar Eisemann, and Klaus Hildebrandt. 2017. Spectral Processing of Tangential Vector Fields. *Computer Graphics Forum* 36, 6 (2017), 338–353. https://doi.org/10.1111/cgf.12942 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12942

Rob Burgoon, Zoë J. Wood, and Eitan Grinspun. 2006. Discrete Shells Origami. In *Computers and Their Applications.*

Dominique Chapelle and Klaus-Jürgen Bathe. 1998. Fundamental considerations for the finite element analysis of shell structures. *Computers & Structures* 66, 1 (1998), 19–36.

Fernando de Goes, Andrew Butts, and Mathieu Desbrun. 2020. Discrete Differential Operators on Polygonal Meshes. *ACM Trans. Graph.* 39, 4, Article 110 (July 2020), 14 pages. https://doi.org/10.1145/3386569.3392389

Fernando de Goes, Mathieu Desbrun, and Yiying Tong. 2015. Vector Field Processing on Triangle Meshes. In *SIGGRAPH Asia 2015 Courses* (Kobe, Japan) *(SA '15).* Association for Computing Machinery, New York, NY, USA, Article 17, 48 pages. https://doi.org/10.1145/2818143.2818167

Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. 2014. Designing N-PolyVector Fields with Complex Polynomials. *Computer Graphics Forum* 33, 5 (2014), 1–11. https://doi.org/10.1111/cgf.12426 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12426

Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Integrable PolyVector Fields. *ACM Trans. Graph.* 34, 4, Article 38 (July 2015), 12 pages.

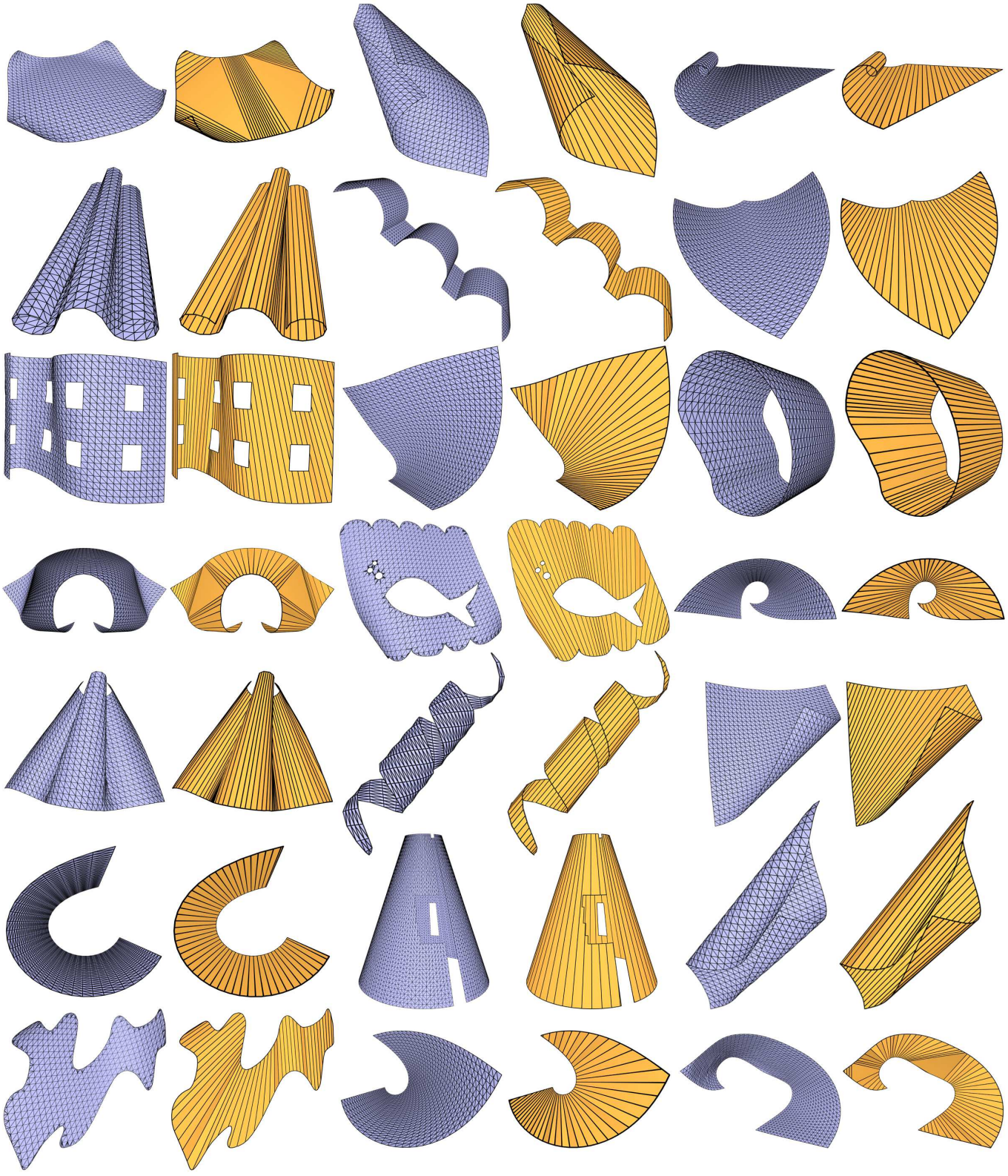


Fig. 24. Various remeshing results obtained with our method. Note that our method can handle a wide variety of developable geometry and topology, including cylindrical topology, multiple holes and sophisticated boundary shapes.

https://doi.org/10.1145/2766906
Stefan Fröhlich and Mario Botsch. 2011. Example-Driven Deformations Based on Discrete Shells. *Comput. Graph. Forum* 30, 8 (2011), 2246–2257.
Michael Grant and Stephen Boyd. 2014. CVX: Matlab Software for Disciplined Convex Programming, version 2.1. http://cvxr.com/cvx.
David A. Huffman. 1976. Curvature and creases: a primer on paper. *IEEE Trans. Computers* 25, 10 (1976), 1010–1019.
Alexandra Ion, Michael Rabinovich, Philipp Herholz, and Olga Sorkine-Hornung. 2020. Shape Approximation by Developable Wrapping. *ACM Transactions on Graphics (proceedings of SIGGRAPH ASIA)* 39, 6 (2020). https://doi.org/10.1145/3414685.3417835
Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. https://libigl.github.io/.
Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Instant Field-Aligned Meshes. *ACM Trans. Graph.* 34, 6 (Nov. 2015). https://doi.org/10.1145/2816795.2818078
Caigui Jiang, Cheng Wang, Florian Rist, Johannes Wallner, and Helmut Pottmann. 2020. Quad-mesh based isometric mappings and developable surfaces. *ACM Trans. Graph.* 39, 4 (2020), 128:1–128:13.
Martin Kilian, Simon Flöry, Zhonggui Chen, Niloy J. Mitra, Alla Sheffer, and Helmut Pottmann. 2008. Curved folding. *ACM Trans. Graph.* 27, 3 (2008), 75:1–75:9.
Martin Kilian, Aron Monszpart, and Niloy J. Mitra. 2017. String Actuated Curved Folded Surfaces. *ACM Trans. Graph.* 36, 4, Article 64a (May 2017), 13 pages. https://doi.org/10.1145/3072959.3015460
Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Globally optimal direction fields. *ACM Trans. Graph.* 32, 4 (2013).
Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2015. Stripe Patterns on Surfaces. *ACM Trans. Graph.* 34 (2015). Issue 4.
M. Kohlbrenner, U. Finnendahl, T. Djuren, and M. Alexa. 2021. Gauss Stylization: Interactive Artistic Mesh Modeling based on Preferred Surface Normals. *Computer Graphics Forum* 40, 5 (2021), 33–43. https://doi.org/10.1111/cgf.14355 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14355
Hsueh-Ti Derek Liu and Alec Jacobson. 2021. Normal-Driven Spherical Shape Analogies. *Computer Graphics Forum* 40, 5 (2021), 45–55. https://doi.org/10.1111/cgf.14356 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14356
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. https://doi.org/10.1145/1141911.1141941
Yang Liu, Weiwei Xu, Jun Wang, Lifeng Zhu, Baining Guo, Falai Chen, and Guoping Wang. 2011. General Planar Quadrilateral Mesh Design Using Conjugate Direction Field. *ACM Trans. Graph.* 30, 6 (Dec. 2011), 1–10. https://doi.org/10.1145/2070781.2024174
William S Massey. 1962. Surfaces of Gaussian curvature zero in Euclidean 3-space. *Tohoku Mathematical Journal, Second Series* 14, 1 (1962), 73–79.
Martin Peternell. 2004. Developable surface fitting to point clouds. *Computer Aided Geometric Design* 21, 8 (2004), 785–803.
Roi Poranne, Elena Ovreiu, and Craig Gotsman. 2013. Interactive Planarization and Optimization of 3D Meshes. *Comput. Graph. Forum* 32, 1 (2013), 152–163. https://doi.org/10.1111/cgf.12005
Helmut Pottmann, Qixing Huang, Bailin Deng, Alexander Schiftner, Martin Kilian, Leonidas Guibas, and Johannes Wallner. 2010. Geodesic Patterns. *ACM Trans. Graphics* 29, 3 (2010). http://www.geometrie.tugraz.at/wallner/geopattern.pdf to appear.
Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2018. Discrete Geodesic Nets for Modeling Developable Surfaces. *ACM Trans. Graph.* 37, 2 (2018), 16.
Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2019. Modeling Curved Folding with Freeform Deformations. *ACM Trans. Graph.* 38, 6 (2019).
Andrew O. Sageman-Furnas, Albert Chern, Mirela Ben-Chen, and Amir Vaxman. 2019. Chebyshev Nets from Commuting PolyVector Fields. *ACM Trans. Graph.* 38, 6, Article 172 (Nov. 2019), 16 pages. https://doi.org/10.1145/3355089.3356564
Silvia Sellán, Noam Aigerman, and Alec Jacobson. 2020. Developability of heightfields via rank minimization. *ACM Transactions on Graphics* 39 (07 2020). https://doi.org/10.1145/3386569.3392419
James Albert Sethian. 1999. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. Vol. 3. Cambridge university press.
Nicholas Sharp and Keenan Crane. 2018. Variational Surface Cutting. *ACM Trans. Graph.* 37, 4 (2018).
Justin Solomon, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2012. Flexible developable surfaces. *Comput. Graph. Forum* 31, 5 (2012), 1567–1576.
Oded Stein, Eitan Grinspun, and Keenan Crane. 2018. Developability of triangle meshes. *ACM Trans. Graph.* 37, 4 (2018).
Chengcheng Tang, Pengbo Bo, Johannes Wallner, and Helmut Pottmann. 2016. Interactive design of developable surfaces. *ACM Trans. Graph.* 35, 2, Article 12 (2016), 12 pages.
Chengcheng Tang, Xiang Sun, Alexandra Gomes, Johannes Wallner, and Helmut Pottmann. 2014. Form-finding with Polyhedral Meshes Made Simple. *ACM Trans. Graph.* 33, 4 (2014). https://doi.org/10.1145/2601097.2601213
Amir Vaxman et al. 2017a. Directional: A library for Directional Field Synthesis, Design, and Processing. https://doi.org/10.5281/zenodo.3338174
Amir Vaxman et al. 2017b. libhedra: geometric processing and optimization of polygonal meshes. https://github.com/avaxman/libhedra.
Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. 2016. Directional field synthesis, design, and processing. *Comput. Graph. Forum* 35, 2 (2016), 545–572.
Josh Vekhter, Jiacheng Zhuo, Luisa F Gil Fandino, Qixing Huang, and Etienne Vouga. 2019. Weaving Geodesic Foliations. *ACM Trans. Graph.* 38, 4, Article 34 (July 2019), 22 pages. https://doi.org/10.1145/3306346.3323043
Ryan Viertel and Braxton Osting. 2019. An Approach to Quad Meshing Based on Harmonic Cross-Valued Maps and the Ginzburg–Landau Theory. *SIAM Journal on Scientific Computing* 41, 1 (2019), A452–A479. https://doi.org/10.1137/17M1142703 arXiv:https://doi.org/10.1137/17M1142703
Hui Wang, Davide Pellis, Florian Rist, Helmut Pottmann, and Christian Müller. 2019. Discrete Geodesic Parallel Coordinates. *ACM Trans. Graph.* 38, 6, Article 173 (Nov. 2019), 13 pages. https://doi.org/10.1145/3355089.3356541
Mirko Zadravec, Alexander Schiftner, and Johannes Wallner. 2010. Designing Quad-dominant Meshes with Planar Faces. *Computer Graphics Forum* 29, 5 (2010), 1671–1679. https://doi.org/10.1111/j.1467-8659.2010.01776.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2010.01776.x

Table 3. Statistics of our results, reported by figure number (in scanline order within figures that show multiple results). We record the number of output mesh vertices $|\mathcal{V}'|$ and faces $|\mathcal{F}'|$, the number of optimization iterations needed to reach convergence, as well as the maximum and mean face planarity error p (in percentage of the average diagonal length of the face). We also report the Hausdorff distance between the input and output mesh. Many of our meshes meet the common planarity tolerance of $\leq 1\%$ without any planarization optimization.

Fig.	$\lvert\mathcal{V}'\rvert$	$\lvert\mathcal{F}'\rvert$	Iter.	$p_{\max}$ [%]	p_{mean} [%]	h[%]
1	693	137	47	0.38	0.06	0.46
3	270	74	5	1.80	0.41	0.67
5	180	51	8	1.53	0.28	0.45
5	206	44	5	0.59	0.15	0.85
5	218	50	9	0.88	0.18	1.07
5	190	48	5	0.95	0.19	2.55
7	190	48	5	0.95	0.19	2.55
8	208	45	4	0.41	0.11	0.66
8	206	44	5	0.59	0.15	0.85
8	210	46	3	0.54	0.16	0.41
8	208	45	4	0.45	0.17	1.39
8	218	50	9	0.88	0.18	1.07
9	288	107	52	11.84	1.98	0.48
9	288	107	-	0.00	0.00	0.95
10	520	110	6	2.58	0.71	0.09
11	1115	275	16	1.62	0.19	0.28
11	1603	747	14	0.31	0.06	1.90
11	3117	1202	299***	7.63	0.21	4.03
11	1473	294	10	1.84	0.12	0.18
12	164	23	3	0.52	0.24	0.47
12	210	46	3	0.54	0.16	0.41
12	298	90	3	0.32	0.10	0.41
13	384	68	7	1.08	0.34	0.31
13	632	67	7	1.22	0.10	0.26
13	1130	66	36	0.25	0.02	0.24
15	176	29	18	3.64	1.04	0.56
15	156	19	22	5.42	2.51	1.34
16	254	68	19	3.83	0.42	0.63
16	254	68	23	3.82	0.42	0.66
16	242	62	51	4.12	0.50	0.23
16	258	70	23	3.34	0.48	0.21
17	2194	447	299***	4.78	0.34	2.90
18	202	100	12	0.14	0.05	0.20
18	324	100	10	0.46	0.14	0.22
18	858	274	9	1.66	0.09	1.15
18	966	272	6	1.14	0.08	0.13
19	306	54	2	0.15	0.04	0.15
19	438	54	6	0.14	0.04	1.91
19	578	93	41	2.55	0.30	1.09
20	2455	440	46	2.65	0.25	3.29
20	3833	1599	46	3.19	0.17	1.37
20	972	195	17	1.71	0.26	0.53
20	1677	752	17	2.16	0.13	0.75
21	288	170	18	0.19	0.05	0.09
21*	1970	567	49	0.82	0.05	0.04
21**	1947	636	18	0.26	0.05	0.01
21	738	211	15	0.84	0.10	0.14
22	258	98	4	1.16	0.24	0.23
22	243	80	44	0.67	0.20	0.62
22	280	115	6	1.98	0.16	0.35
22	232	82	21	2.24	0.32	0.42
22	693	137	47	0.38	0.06	0.46
23	196	52	4	2.39	0.37	0.26
23	210	28	9	3.49	0.43	1.47
24	254	68	15	3.63	0.41	0.65
24	210	46	4	1.25	0.19	0.23
24	204	43	20	7.18	0.50	0.34
24	212	47	3	0.56	0.33	0.60
24	390	70	7	3.93	0.31	0.67
24	204	43	3	0.69	0.14	0.10
24	422	88	5	3.35	0.56	0.85
24	208	45	7	0.43	0.10	0.46
24	226	63	8	0.40	0.12	0.18
24	328	75	40	0.78	0.18	0.69
24	1033	71	7	1.72	0.34	0.38
24	218	50	4	2.35	0.20	0.32
24	242	58	4	0.80	0.36	1.29
24	288	107	52	11.84	1.98	0.48
24	208	45	4	0.38	0.11	0.20
24	214	48	9	0.15	0.05	0.08
24	502	53	16	0.32	0.06	0.09
24	194	45	3	1.19	0.24	0.40
24	609	47	7	2.89	0.60	0.21
24	204	43	7	0.49	0.13	0.39
24	288	67	16	1.31	0.20	0.82

* Same model displayed from 3 viewing angles.
** Same model displayed from 2 viewing angles.
*** Maximal number of iterations, not converged.