Dev2PQ: Planar Quadrilateral Strip Remeshing of Developable Surfaces

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

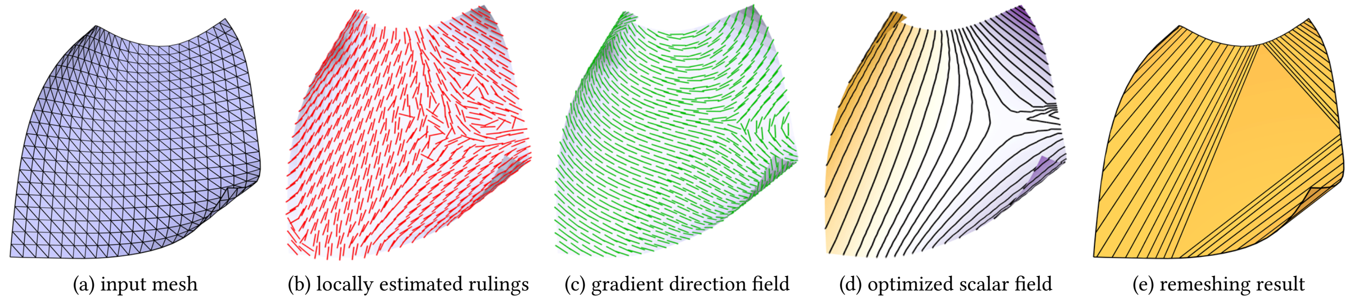
(a) input mesh
(b) locally estimated rulings
(c) gradient direction field
(d) optimized scalar field
(e) remeshing result

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 (d) whose level sets align as well as possible to the locally estimated rulings (b). Our vector field contains a naturally-placed singularity in the planar region, which automatically generates a triangular polygon.

ABSTRACT

We introduce an algorithm to remesh meshes representing developable surfaces to planar quad dominant meshes. The output of our algorithm consists of planar quadrilateral (PQ) strips aligned to principal curvature directions and closely approximating the curved parts of the input developable, along with planar polygons representing the flat parts of the input. Such developable PQ-strip meshes are useful in many areas of shape modeling thanks to the simplicity of fabrication from flat sheet material. However, they are difficult to use in the modeling phase due to their restrictive combinatorics and locking issues. 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 curvature. Our method enables one to leverage the modeling flexibility of non-ruling based representations of developable surfaces but still obtain developable, curvature-aligned PQ-strip meshes. Our algorithm optimizes a scalar function on the input mesh, such that its level sets are straight and align well to the locally estimated ruling directions. The condition that guarantees straight level sets is nonlinear of high order and numerically difficult to enforce in a straightforward manner. We devise a dedicated 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.

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 hard to design due to their highly constrained nature, which is why in practice, most of the time 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

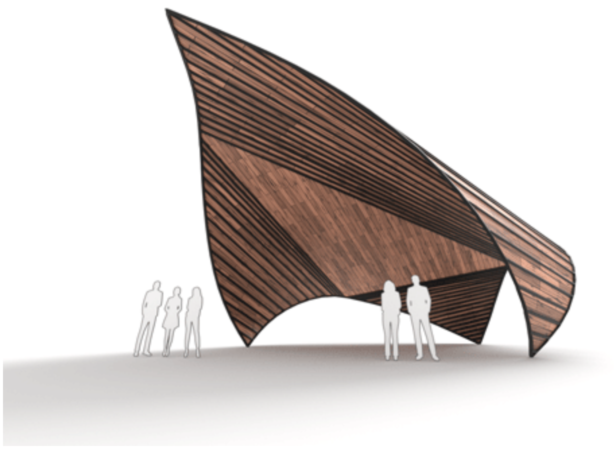

Fig. 2. Developable meshes with planar faces have many applications in fabrication and architecture. Rendering of our result from Fig. 1.

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

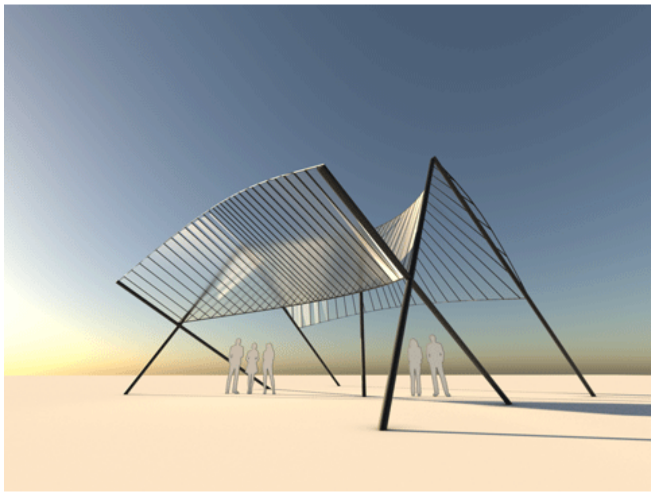

Fig. 3. Rendering of a result from Fig. 18 with two curved folds.

Botsch 2011]). Using such representations in the common handle-based editing paradigm is hindered by locking issues that make the exploration of the developable shape space difficult, as described in [Alessio 2012; Chapelle and Bathe 1998; Rabinovich et al. 2018; Tang et al. 2016]. The recently proposed discrete orthogonal geodesic nets [Rabinovich et al. 2018] and the checkerboard pattern isometries [Jiang et al. 2020], lift this limitation by representing developable surfaces without explicitly accounting for principal curvature directions. These methods enable intuitive surface modeling via freeform deformation, as if bending and manipulating a piece of paper.

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]. 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 with a constant normal along them, such that 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].

In this paper, we develop a method to convert a triangle mesh representation of a developable surface into discrete curvature line representation in order to reap the benefits of both worlds: the support for unhindered 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 and closely approximating 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. If desired, the long polygons can be further tessellated into circular PQ elements by inserting the other family of parametric lines to form trapezoids.

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; Vaxman et al. 2016]. However, to the best of our knowledge, no existing general remeshing method is guaranteed to perfectly align to principal directions and produce edge flows that are entirely consistent with curvature lines, which are often difficult to obtain fully and faithfully for discrete meshes. In this work we exploit the 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 level sets 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 level sets 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 directional-field optimization and the fitting of a scalar function to it. This makes our problem tractable and practical to optimize. We extract the level sets of the obtained scalar field at the desired resolution and remesh the input into strips of planar quads whose chordal edges are the level sets, i.e., the rulings, supplemented 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 inclusion 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. 4).

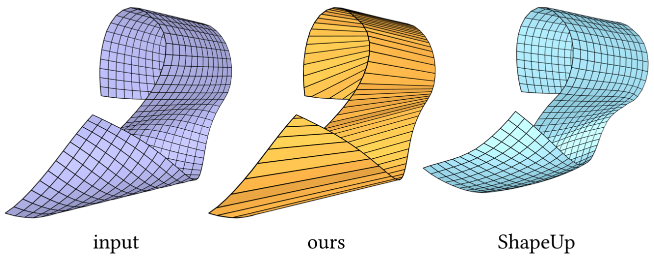

Fig. 4. Attempting to convert a quad mesh to a PQ mesh using a general-purpose planarization technique (ShapeUp [Bouaziz et al. 2012]) significantly alters the shape and makes it non-developable. This happens because the edges of the input mesh are generally not aligned to principal curvature directions. Our method is applied to a trivial triangulation of the input mesh. The Hausdorff distance also indicates that our method approximates the input better: for our result it is 0.67% of bounding box diagonal, and for the ShapeUp result it is 9.74%.

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. 6).

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 fitting and the strict segmentation structure imposed by developable geometry. 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.

Wang and colleagues [2019] represent developable surfaces by discrete parallel geodesic nets. This representation admits a local estimate of ruling directions, which they use to approximate their surfaces by thin, separate strips of nearly planar quadrilaterals. In contrast, our method produces a complete, connected remeshing of the input developable surface with globally consistent and straight rulings.

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 CONTINUOUS AND DISCRETE DEVELOPABLE SURFACES

In this section, we summarize relevant facts about 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 curvature. They are bounded by rulings of torsal patches and the boundary of the surface, as shown in Fig. 5.

Non-smooth developables. We also consider more general, piecewise developable surfaces. One type are *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. 18), and *creases* when this is not the case (e.g., Fig. 15). 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. 16). These

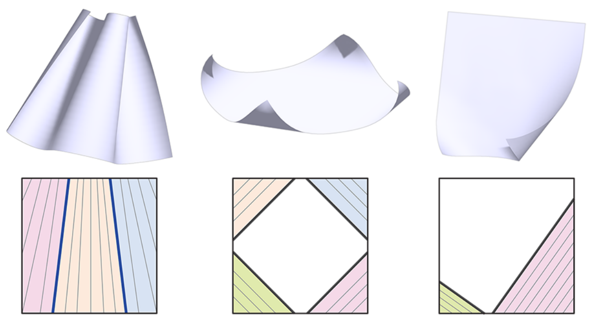

Fig. 5. Developable surfaces (top row) and their decompositions into planar and curved (torsal) patches, shown on the 2D development (bottom row). We display the planar patches in white and the curved patches in different colors. The rulings are illustrated as thin grey lines, with the borders between curved and flat patches in thick black and inflection lines in blue.

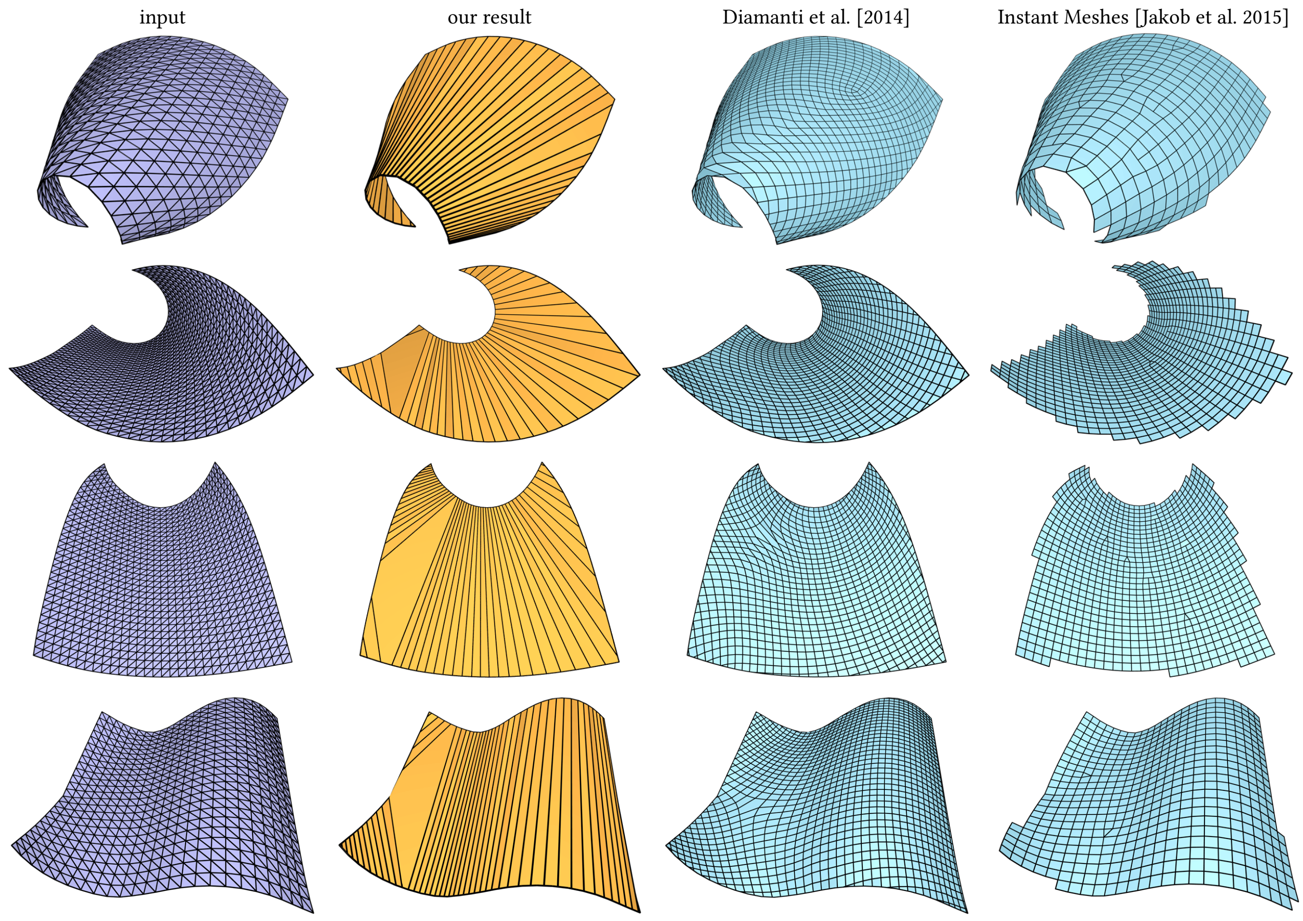

Fig. 6. Remeshing an input developable surface using the vector field design of [Diamanti et al. 2014] does not result in globally straight edge sequences. Instant Meshes, the curvature-aligned quad dominant remeshing technique of [Jakob et al. 2015], introduces singularities and does not always succeed 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).

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 together with their incident faces. If desired, they can be added back in post-processing.

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

$$S_i(u, w) = p(u) + w\, r(u), \tag{1}$$

where $p(u) : \mathbb{R} \to \mathbb{R}^3$ is a *generating curve*, and every u-level-set 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-level-sets in order for S_i to be developable: $n(u, w) = n(u)$. This means that the u-level-sets are *extrinsically flat*; 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 $S_i(u, w)$ a principal curvature line parameterization.

Conjugate nets are more versatile than is expressible by Eq. (1) for torsal patches; by enforcing u to be continuous between the patches $\mathcal{S}_i$, and allowing for singularities to emerge, we obtain a single conjugate parameterization for the entire surface $\mathcal{S}$. If the surface is C^2-continuous, singularities must naturally be located in planar regions (see Fig. 1). Furthermore, u, like rulings, is C^0-continuous across crease curves.

Order of symmetry. Note that the u and w coordinates are distinct in a single torsal patch, as the u-level-sets are rulings, and the w-level-sets are generating curves. Since we maintain these roles on the entire surface $\mathcal{S}$, we get that u and w are always separable, and each is individually symmetric up to sign. While this is in accordance with principal parameterizations, general conjugate nets typically allow u and w to intermix when the goal is to obtain PQ meshes [Diamanti et al. 2014; Liu et al. 2011]. According to the taxonomy in [Vaxman et al. 2016], separable sign-symmetric u and w are 2^2-symmetric, and when intermixed they have 4-symmetry. In our work, we design the 2^2 variant, as we are looking to generate PQ-strip meshes from the u-level-sets, rather than a generic PQ discretization; in fact, we only design u, defining a strip foliation on the mesh, and let w be a dependent orthogonal coordinate.

3.2 Ruling fields

Our work focuses on designing directional fields that generate the rulings of a developable surface from other representations, and meshing accordingly. Consider the vector field ∇u, which is by definition orthogonal to the level sets of u. The geodesic curvature of level sets is defined as: $\kappa_g(u) = \nabla \cdot \frac{\nabla u}{\|\nabla u\|}$ [do Carmo 1976]. Since the u-level-sets are extrinsically flat, and a developable is locally isometric to the plane, we have $\forall u,\ \kappa_g(u) = 0$.

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-directional field γ on a developable surface $\mathcal{S}$, which is the assignment of a tangent vector γ to every point $p \in \mathcal{S}$, and which is defined up to sign. If we align γ with $r^\perp$, we have by definition

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

For simplicity we first consider the case where γ does not have singularities, and the surface $\mathcal{S}$ does not contain creases. We then get:

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

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

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

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

Singularities and combing. We design γ as a 2-directional 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, γ can be *combed* by consistently choosing one of the directions to obtain a smooth single-vector field on which we adhere to conditions (3) and (4).

Since our field γ is 2-symmetric, singularities have indices that are integer multiples of $\pm\frac{1}{2}$. Therefore, the field γ 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 level sets of u are not straight there. However, singularities cannot arise inside torsal patches by definition, since the ruling field is well defined. Therefore there are only two possible types of singularities: a) cone apexes, of index 1; the parameter u is undefined at such vertices, and we avoid them in our formulation, as explained in Sec. 3.1; b) singularities inside planar regions, where they compensate for the different orientations of the neighboring torsal patches (see Fig. 1c). As such, the fact that the level sets are not straight in these planar regions does not pose a subsequent problem in our discretization (see Sec. 3.3).

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 $\gamma^\perp$ 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, γ has to be designed such that $\gamma^\perp$ 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. Given the second fundamental form II, we then also must have:

$$\forall p \in \mathcal{S},\ II\left(\gamma^\perp(p), \gamma^\perp(p)\right) = 0. \tag{5}$$

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. 17, 18). We therefore do not require γ to be divergence-free near creases, effectively allowing the vector field to break across them.

3.3 Discrete ruling-aligned developable meshes

A discrete sampling of the u, w level sets of a conjugate net creates a quadrilateral mesh whose faces are planar up to second order [Liu et al. 2006]. Sampling curvature-line parameterizations leads to circular quad meshes up to second order [Bobenko and Pinkall 1999] (a quadrilateral is circular if it can be inscribed into a circle). In discrete differential geometry, circular meshes constitute a model of discrete curvature line nets [Bobenko and Suris 2008; Bobenko and Tsarev 2018] and admit vertex offsets. Moreover, 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 design u independently and leave w as a dependent coordinate, 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 level sets are fully contained in the plane, and we are therefore allowed to straighten them out. If the sum of singularity indices in a planar region $\mathcal{S}_i$ is I_i , then the polygon will have $4 + 2I_i$ sides; for instance, in Fig. 8, the singularity is of index $\frac{1}{2}$, and the planar polygon is a hexagon (with small boundary edges).

4 DISCRETIZATION

The input to our algorithm is a triangle mesh $\mathcal{M} = \{\mathcal{V}, \mathcal{E}, \mathcal{F}\}$ of the (piecewise) developable surface. 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 $\mathcal{X}$. For convenience, we uniformly scale the input $\mathcal{M}$ such that its average edge length becomes 1. We use the conforming discrete gradient $G : \mathcal{V} \to \mathcal{X}$ and divergence $D : \mathcal{X} \to \mathcal{V}$ operators, and the non-conforming discrete curl operator $C : \mathcal{X} \to \mathcal{E}$. 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 $R^{\perp}(i) = (r^{\perp}(i))^2$, 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, but in reality we cannot obtain such a clean segmentation directly, because the measure of curvedness is noisy, like the ruling estimates, and does not delineate planar and torsal patches nicely, as can be seen in Fig. 7. Therefore we instead model the fact that the rulings are least reliable in planar or near-planar regions, and mostly consistent in strongly curved areas (see Figs. 1 and 8), by attaching 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 $\kappa_2(f)$:

$$w(f) = \theta_1 \left(1 - e^{\theta_2(\theta_3(\kappa_1(f) - \kappa_2(f)))^2}\right). \tag{6}$$

For $\kappa_1(f)$ and $\kappa_2(f)$ we use the absolute largest and smallest eigenvalues of the shape operator $\mathcal{S}(f)$, we set $\theta_1 = 0.8$, $\theta_2 = -0.9$ and $\theta_3 = 5$. The confidence function is a logistic curve, facilitating a stronger distinction between confidence in planar and near-planar regions (albeit still small compared to stronger curved regions). By design $w(f)$ is capped at 0.8 (assigned when κ_1 and κ_2 differ by 0.5 or more), 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.

Crease detection. As an optional step we identify a set of crease edges $\mathcal{E}_c$ by collecting all edges whose adjacent face normals differ by more than a user-defined threshold. 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 $\mathcal{F}_c$. This preprocessing step has been included for all results in Figures 17 and 18. Alternatively, the creases can be prescribed by a user to the same effect, as we did for the examples in Fig. 15.

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.

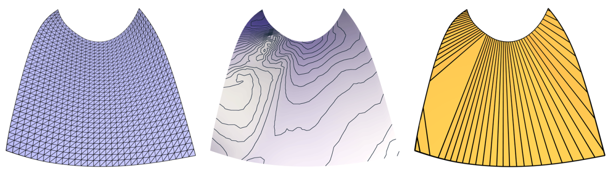

Fig. 7. Level sets of the curvedness measure $|\kappa_2 - \kappa_1|$ (middle) do not provide a clean delineation between torsal and planar parts. However, our method automatically places a planar polygon in this region, without being provided with an explicit decomposition (right).

5.1 Optimization problem

We start by computing the face-based shape operator and the ruling related quantities: $S(f)$, $r(f)$, $r^{\perp}(f)$, $R(f) = r^2(f)$, $R^{\perp}(f) = (r^{\perp})^2(f)$, as well as the confidence weights $w(f)$, as detailed in Sec. 4. We further consider the face-based mass matrix $M_{\mathcal{X}}$ holding the face areas $m(f)$ and an edge-based mass matrix $M_{\mathcal{E}}$ holding edge masses $m(e) = \frac{\|e\|}{\|e_{\text{dual}}\|}(m(f) + m(g))$ 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 $\gamma \in \mathcal{X}$ we use the discrete divergence $D\gamma = G^{\mathsf{T}} M_{\mathcal{X}} \gamma\ (\in \mathbb{R}^{|\mathcal{V}|})$, 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 $Gu_{ijk}(f) = \frac{1}{2m(f)}(e^{\perp}_{jk}u_i + e^{\perp}_{ki}u_j + e^{\perp}_{ij}u_k)$.

Our method optimizes for a unit field $\gamma(f)$ and its power representation $\Gamma(f) = \gamma^2(f)$, where $\Gamma(f)$ should align to the perpendicular ruling power field $R^{\perp}(f)$ according to the confidence $w(f)$, and where γ is divergence-free away from singularities. Furthermore, we optimize for a scalar field $s(f)$, such that $s\,\gamma$ is curl-free. For this, we introduce the following terms.

Alignment objective. Our alignment term is

$$E_a(\Gamma) = \sum_{f \in \mathcal{F}} m(f) w(f)\, \|\Gamma(f) - R^{\perp}(f)\|^2, \tag{7}$$

where $m(f)$ is the face area of f. This can be formulated in matrix form as

$$E_a(\Gamma) = \left(\Gamma - R^{\perp}\right)^H M_{\mathcal{X}} W_{\mathcal{X}} \left(\Gamma - R^{\perp}\right), \tag{8}$$

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

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 meaningful on torsal patches. We follow [Viertel and Osting 2019] and [Sageman-Furnas et al. 2019] by using a *Ginzburg-Landau* approach, introducing the following objective term:

$$E_d(\gamma) = \sum_{v \in \mathcal{V}} |D\gamma(v)|^2 + \frac{1}{\epsilon^2} \sum_{f \in \mathcal{F}} \left(\|\gamma(f)\|^2 - 1\right). \tag{9}$$

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

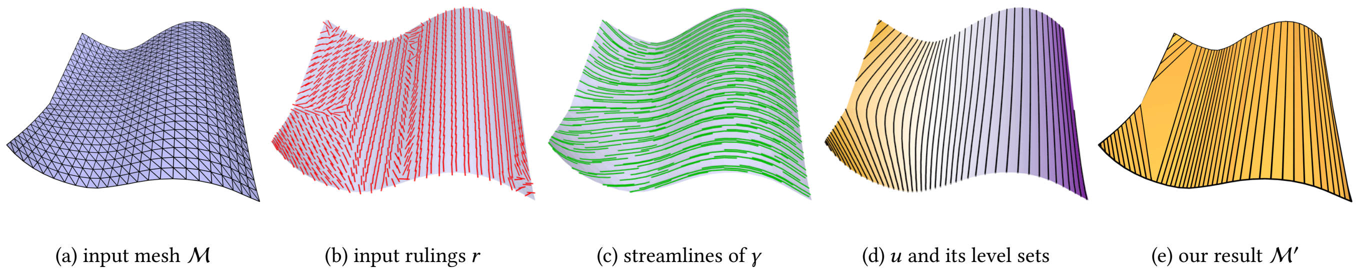
(a) input mesh $\mathcal{M}$
(b) input rulings r
(c) streamlines of γ
(d) u and its level sets
(e) our result $\mathcal{M}'$

Fig. 8. Our remeshing pipeline: (a) The original input $\mathcal{M}$; (b) the noisy input rulings r; (c) our computed γ field, visualized with streamlines; (d) the optimized function u with its level sets; (e) the final remeshing result $\mathcal{M}'$. Note how the level sets 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.

patches, this will naturally locate singularities (if any) inside planar regions, where they should be.

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 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:

$$\|\Gamma(f)\,\bar{e}_f^2 - \Gamma(g)\,\bar{e}_g^2\|^2. \tag{10}$$

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

$$E_s(\Gamma) = \sum_{e\in\mathcal{E}} m(e)\,(1-w(e))\,\|\Gamma(f)\,\bar{e}_f^2 - \Gamma(g)\,\bar{e}_g^2\|^2, \tag{11}$$

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

$$L_2 = G_{\mathcal{E}}^H M_{\mathcal{E}}\,(I - W_{\mathcal{E}})\,G_{\mathcal{E}}, \tag{12}$$

where $G_{\mathcal{E}}$ stacks the differences $\Gamma(f)\,\bar{e}_f^2 - \Gamma(g)\,\bar{e}_g^2$ from Eq. (10).

Integrability. We use the discrete curl operator C to measure integrability of the scaled field $s\gamma$:

$$Cs\gamma(e) = \langle s(f)\gamma(f) - s(g)\gamma(g),\ e\rangle\,. \tag{13}$$

We constrain

$$Cs\gamma = 0. \tag{14}$$

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

$$s_{\text{low}} < s < s_{\text{high}}. \tag{15}$$

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

Branching and singularities. Generating Γ from γ 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 γ 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:

$$(\Gamma,\gamma,s) = \arg\min\ \omega_a E_a(\Gamma) + \omega_d E_d(\gamma) + \omega_s E_s(\Gamma),\quad s.t. \tag{16}$$
$$\forall f\in\mathcal{F},\ \Gamma(f) = \gamma^2(f), \tag{17}$$
$$Cs\gamma = 0, \tag{18}$$
$$s_{\text{low}} < s < s_{\text{high}}\,. \tag{19}$$

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

5.2 Optimization algorithm

We now detail our numerical solution strategy for the optimization problem described in Eqs. (16)-(19). As the optimization problem is separable in the Γ, γ and s variables, we optimize for them in an alternating fashion, following the spirit of [Sageman-Furnas et al. 2019]. We alternate between implicit Euler steps that decrease the alignment and smoothness terms of the quadratic objective (16), pointwise renormalizations as needed for the E_d term and projections onto the spaces of divergence-free and integrable vector fields for the E_d term and Eq. (18). Our method proceeds as described in Algorithm 1.

ALGORITHM 1: Optimize vector field Γ

```
Initialize $\Gamma^0 = R^\perp,\ k = 0,\ \mathcal{V}^* = \mathcal{V} \setminus (\mathcal{V}_b \cup \mathcal{V}_c)$
repeat
    $k \leftarrow k+1$
    $\Gamma_a^k \leftarrow$ ImplicitAlign($\Gamma^{k-1}$)
    $\Gamma_s^k \leftarrow$ ImplicitSmooth($\Gamma_a^k$)
    $\forall f\in\mathcal{F},\ \Gamma_u^k(f) \leftarrow \frac{\Gamma_s^k(f)}{\|\Gamma_s^k(f)\|}$
    $\gamma_u^k \leftarrow$ LocalRawRepresentation($\Gamma_u^k$)
    $\mathcal{V}^* \leftarrow$ UpdateSingularities($\gamma_u^k$)
    $\gamma_d^k \leftarrow$ ProjectDivFree($\gamma_u^k$)
    $\gamma_c^k \leftarrow$ ProjectCurlFree($\gamma_d^k$)
    $\Gamma^k \leftarrow$ PowerRepresentation($\gamma_c^k$)
until $\max_f \|\Gamma^k(f) - \Gamma^{k-1}(f)\| < 10^{-3}$;
```

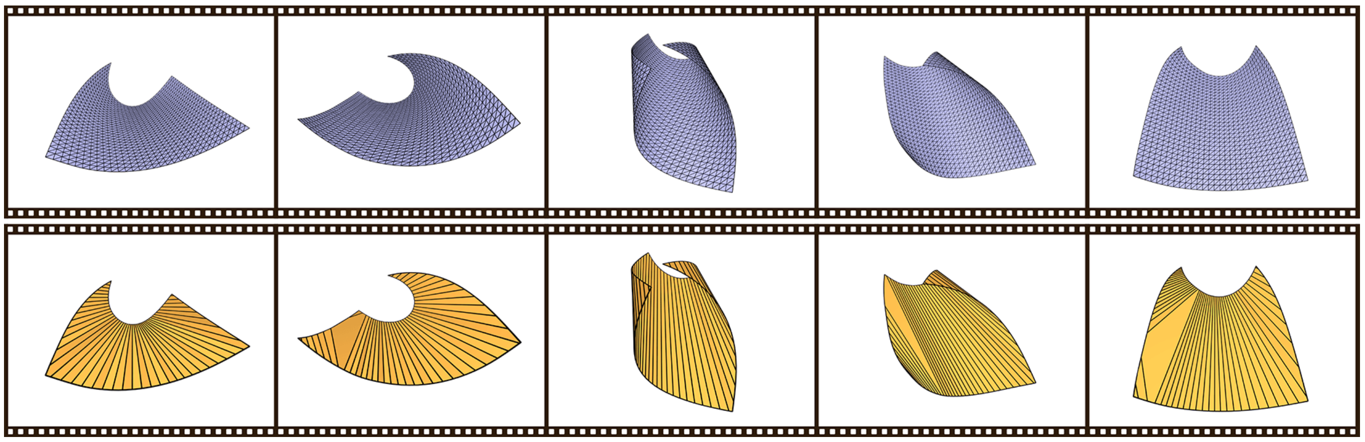

Fig. 9. Several snapshots from an interactive editing session. The user deforms the DOG model by interacting with point handles at some selected vertices. At any time during the interactive session, the user may invoke our remeshing algorithm and view the curvature-aligned remesh nearly instantaneously. 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.

The function `ImplicitAlign(`Γ^{k-1}`)` solves the following linear system:

$$\left(M_{\mathcal{X}} + \frac{\omega_a}{\mu_a}\nabla E_a\right)\Gamma_a^k = M_{\mathcal{X}}\,\Gamma^{k-1} + \frac{\omega_a}{\mu_a} M_{\mathcal{X}} W_{\mathcal{X}} \Gamma^0, \tag{20}$$

where $\nabla E_a = M_{\mathcal{X}} W_{\mathcal{X}}$.

Similarly, the function `ImplicitSmooth(`Γ_a^k`)` solves the following linear system:

$$(M_{\mathcal{X}} + \frac{\omega_s}{\mu_s}\nabla E_s)\Gamma_s^{k'} = M_{\mathcal{X}}\,\Gamma_a^k\,, \tag{21}$$

where ∇E_s is equal to L_2 from Eq. (12).

The Euler step sizes ω_a, ω_s are resized by μ_a, μ_s, respectively, where μ_a, μ_s are the lowest nonzero generalized eigenvalues of ∇E_a and ∇E_s, respectively, with $M_{\mathcal{X}}$ as the mass matrix (i.e., $\exists \beta \neq 0$ s.t. $\nabla E_s\,\beta = \mu\, M_{\mathcal{X}}\,\beta$). 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 and transforming it to a local raw representation, the function `UpdateSingularities(`γ_u^k`)` identifies current singularities in the vector field and updates $\mathcal{V}^*$ accordingly. Note that $\mathcal{V}_b$ and $\mathcal{V}_c$ always remain subsets of $\mathcal{V}^*$. Then the function `ProjectDivFree(`γ_u^k`)` solves the following linear system:

$$\arg\min_{\gamma_d^k} \|\gamma_d^k - \gamma_u^k\|^2 \;\text{ s.t. }\; D\gamma_d^k\left(\mathcal{V}^*\right) = 0, \tag{22}$$

where $D\gamma_d^k$ includes an encoding for the local principal matching around the vertices and is only applied to vertices in $\mathcal{V}^*$.

Finally, the function `ProjectCurlFree(`γ_d^k`)` solves the following convex system:

$$\arg\min_{\gamma_c^k,\, s} \|\gamma_c^k - s\gamma_d^k\|^2, \tag{23}$$

$$s.t.\; C\gamma_c^k = 0, \tag{24}$$

$$0.4 \leq s \leq 1.6, \tag{25}$$

where C also encodes the principal matching over edges. In our experiments it typically takes 50 iterations or less for Γ to converge.

5.3 Vector field integration and meshing

Having an integrable γ, we can use an off-the-shelf seamless function integrator implemented in Directional [Vaxman et al. 2017] to obtain 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 $u \in \frac{1}{2}\mathbb{Z}$ values around singularities, since then the level sets avoid meeting at these singularities, subdividing the planar polygons.

To create the final PQ-strip mesh, we trace the integer level sets of u and then collapse all valence-2 vertices that are not on the boundary. This effectively straightens the polylines of the level sets, which has little effect in torsal regions, since the level sets are almost straight by the optimization. However, the level sets in planar regions become chords between boundary vertices. We illustrate our remeshing pipeline in Fig. 8.

6 RESULTS AND DISCUSSION

We implemented our algorithm using libigl [Jacobson et al. 2018] and Directional [Vaxman et al. 2017] 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. (23)-(25). 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

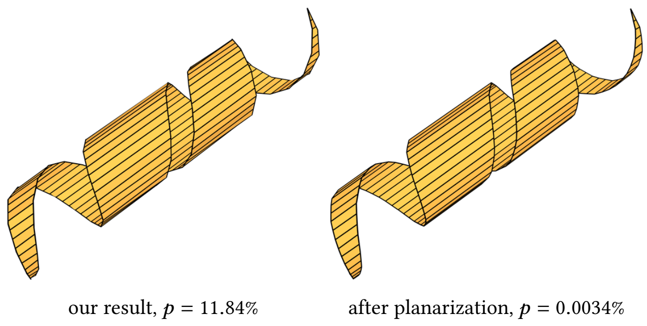

Fig. 10. Our result from Fig. 20 is planarized using ShapeUp [Bouaziz et al. 2012], achieving maximal face planarity error of $p = 0.0034\%$, compared with $p = 11.84\%$ in our initial result. The visual difference between the results is negligible. The Hausdorff distances are reported in Table 2.

of up to 160k faces – for which the method also converges, albeit slower (100 iterations). The parameterization part of our method takes ca. 10-15 seconds.

A variety of our results can be seen in Fig. 20. 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. 20. Our method is applicable to developables with curved folds, as seen in Fig. 3 and 18 (the input models are from [Rabinovich et al. 2019]). 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. 15, as well as other shapes with creases from [Tang et al. 2016], see Fig. 17. We successfully apply our method to glued constructions from [Jiang et al. 2020], including point singularities, see Fig. 16. We have physically fabricated some of our results, shown in Fig. 19. Table 2 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 after trivial triangulation. See Fig. 9 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 consecutive quads. An acceptable stringent tolerance for the planarity error is $\leq 1\%$. It is generally

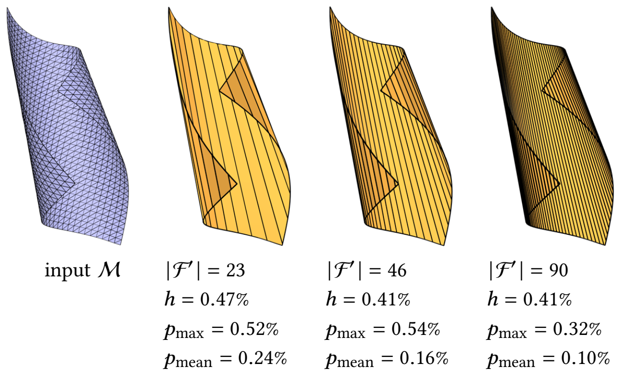

Fig. 11. Sampling the level sets of our optimized function u with increasing density leads to finer remeshing of the input mesh, where the Hausdorff distance to the input h, as well as the maximal and mean polygon planarity error, $p_{\max}$ and p_{mean}, decrease. The output resolution is denoted by the number of faces $|\mathcal{F}'|$. The Hausdorff distance is reported relative to the bounding box diagonal.

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 2. Even though our output meshes are quite coarse, our planarity errors are typically very low, very close to the tolerance. We planarize the example that has the worst maximum planarity error ($p = 11.84\%$) to perfection using ShapeUp [Bouaziz et al. 2012] and reach a visually highly similar result, see Fig. 10. This demonstrates the capability of our algorithm to utilize the information in the original mesh effectively.

Effect of output resolution. We vary the number of isovalues and extract varying amounts of level sets of u 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, see Fig. 11. We note that the approximation quality and the planarity improve with higher resolution, although even for the coarsest resolution these metrics are already very good.

Comparison with analytical principal curvature direction. We test our method on an input mesh sampled from an analytical clothoid surface with varying resolution and compare the obtained vector field γ with the analytical max curvature directions, see Fig. 12 and Table 1. Note that the input to our method are *numerically estimated* ruling directions r, not their analytical values. These estimates can have significant error w.r.t. to the true analytical value, as reported in the second and third column of Table 1, whereas our method achieves lower errors (fourth and fifth columns of Table 1). As the data shows, upon refinement of the input mesh, our output field converges towards the analytical solution.

Robustness. We observe robustness of our method with respect to the parameters ω_a and ω_s. There is a range of values for these parameters 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

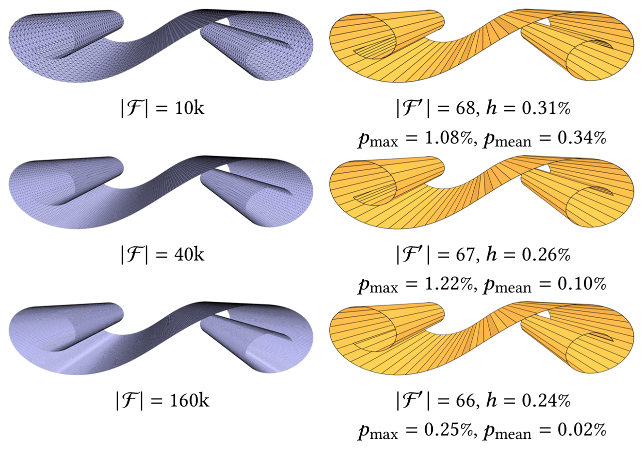
$|\mathcal{F}| = 10k$
$|\mathcal{F}'| = 68$, $h = 0.31\%$
$p_{\max} = 1.08\%$, $p_{\text{mean}} = 0.34\%$
$|\mathcal{F}| = 40k$
$|\mathcal{F}'| = 67$, $h = 0.26\%$
$p_{\max} = 1.22\%$, $p_{\text{mean}} = 0.10\%$
$|\mathcal{F}| = 160k$
$|\mathcal{F}'| = 66$, $h = 0.24\%$
$p_{\max} = 0.25\%$, $p_{\text{mean}} = 0.02\%$

Fig. 12. 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.

Table 1. Difference between our optimized vector field and the analytical principal curvature directions on the clothoid mesh shown in Fig. 12 (angular difference reported in degrees).

	Analytical vs. input		Analytical vs. output	
$\|\mathcal{F}\|$	max °	mean °	max °	mean °
10k	89.74	3.42	9.49	2.24
40k	89.82	1.86	4.80	1.19
160k	89.92	1.01	2.31	0.52

output mesh. For noisy inputs, as in Fig. 13 (right), 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. 13 (right) we use $\omega_s = 0.15$). For optimal alignment the value of ω_s should be chosen as small as possible; e.g., for the cone in the second to last row of Fig. 20 we use $\omega_s = 0.00005$ to emphasize better alignment near the boundary. Our method is robust to some noise on the input meshes, as can be seen in Fig. 13.

Limitations. Our method is not entirely triangulation independent, as shown in Fig. 14. If the input meshing is at odds with the principal curvature directions, this leads 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.

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. 12, 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.

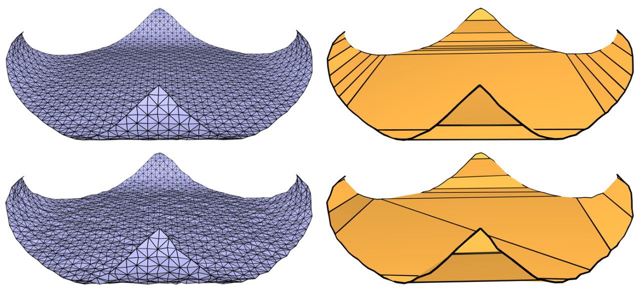

Fig. 13. Our method is robust to noise on developable inputs. These examples show our third example from Fig. 14, but with random vertex displacements applied. Left: a displacement of maximally 12.5% of the average edge length is applied, right: maximally 25% of the average edge length. Our method still recovers a meshing that is compatible with the original principal directions.

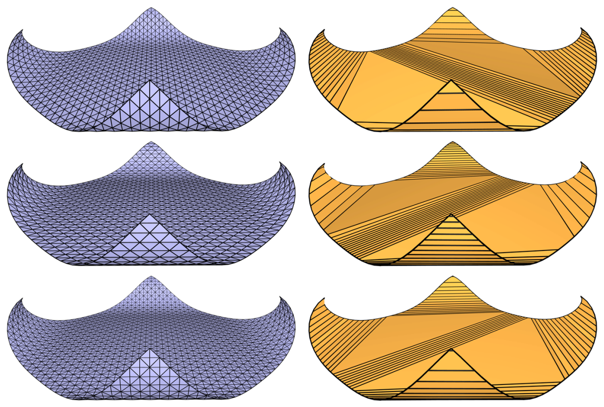

Fig. 14. Different triangulations of a quad mesh lead to different remeshing results, mainly in the near-planar regions. Nevertheless, all of the resulting meshing directions on the planar region are valid.

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. As shown in previous works, interactive deformation-based modeling with curvature-aligned meshes of developable surfaces is limited by the necessity to explicitly maintain the combinatorics of decomposition into torsal and planar patches. Using a non-curvature-aligned representation during the creative modeling stage and invoking our method whenever a curvature-aligned, easily refinable and fabricable representation is required,

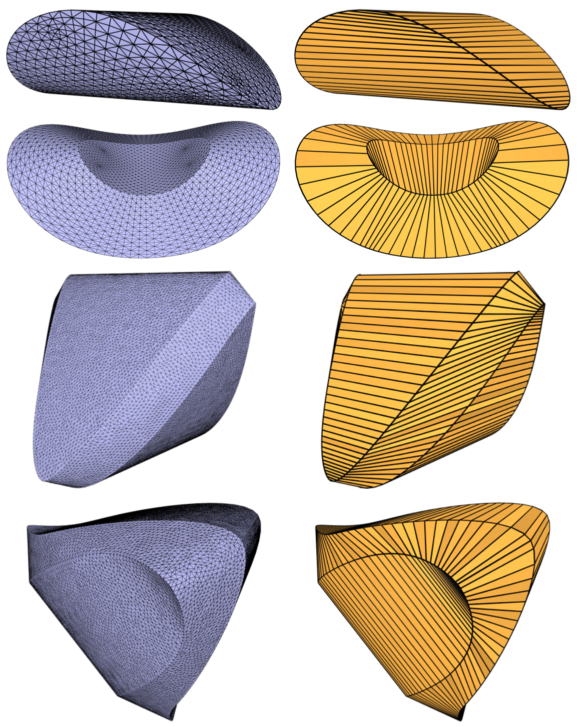
Fig. 15. Sphericons and D-forms are piecewise developable surfaces with creases connecting the individual pieces, and therefore can be remeshed with our method. The top two models are courtesy of [Jiang et al. 2020], the bottom two are courtesy of [Tang et al. 2016].

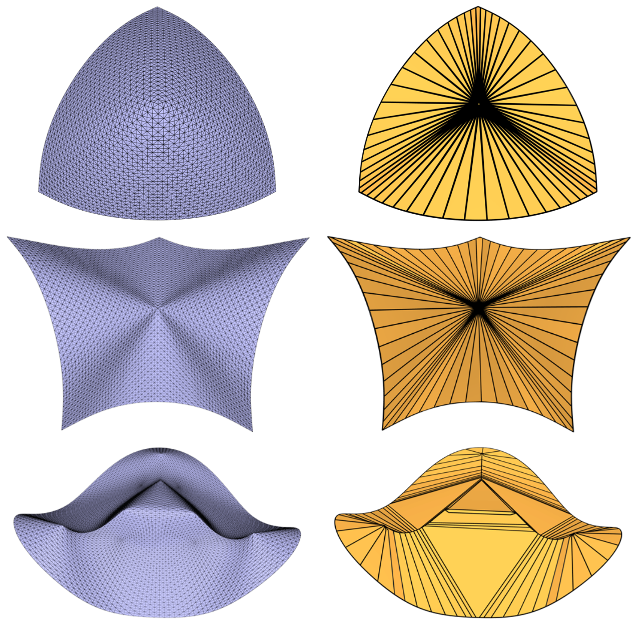
Fig. 16. Our method applied to glued developable surfaces that include cone apexes. These models are courtesy of [Jiang et al. 2020].

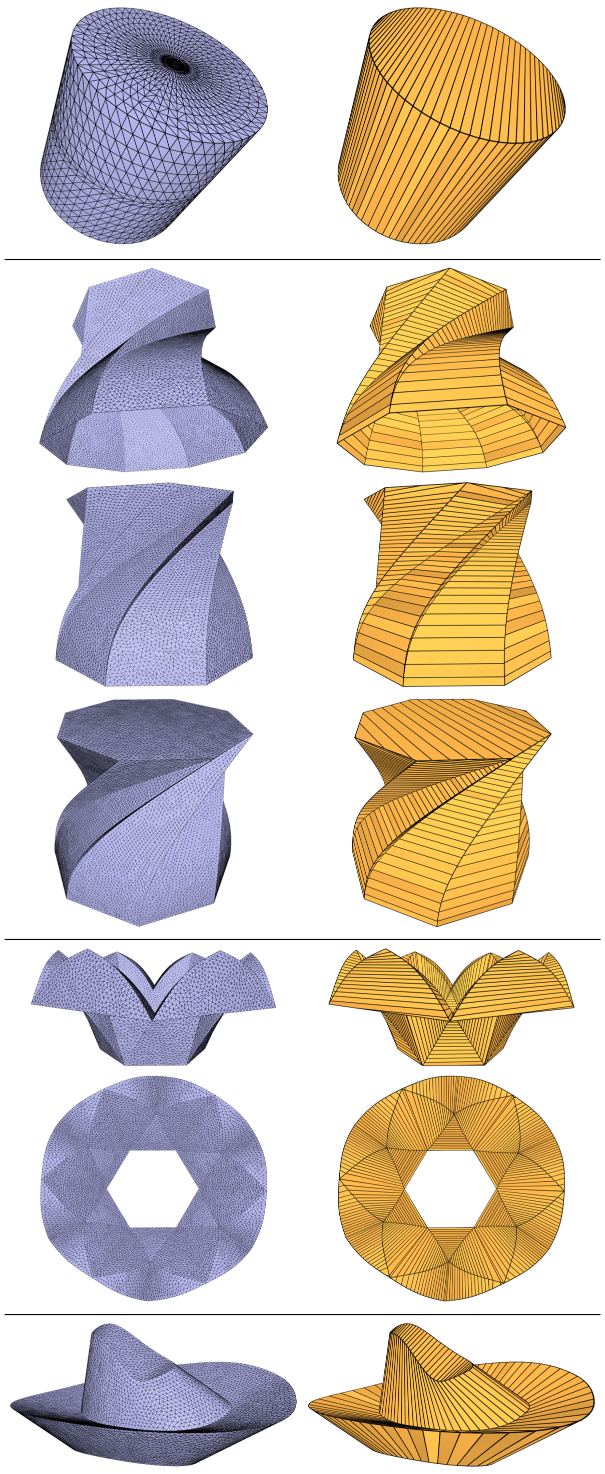
Fig. 17. Our method can handle piecewise developable surfaces with or without boundary and of different genera. These models are courtesy of [Tang et al. 2016].

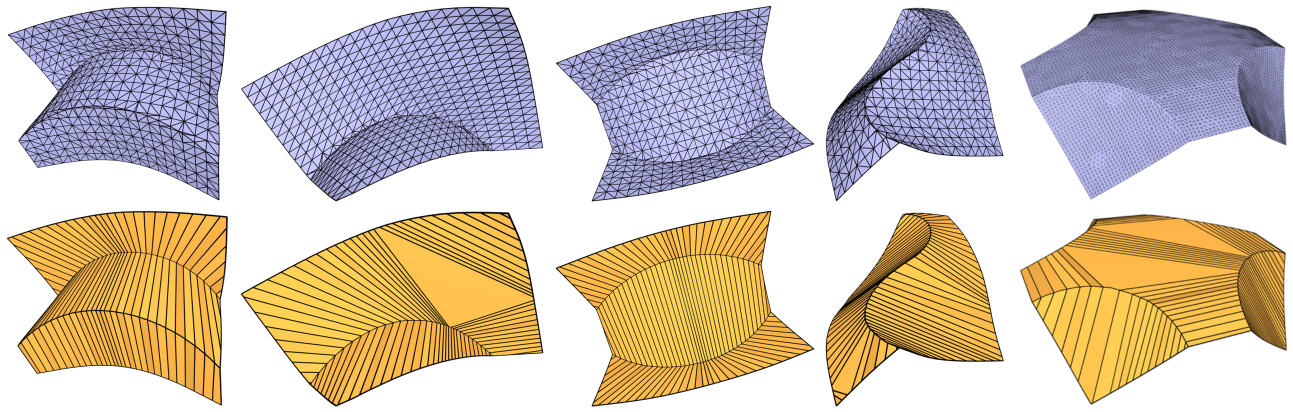

Fig. 18. For surfaces with curved folds our method produces meshes with faces that align well along the folds. Models courtesy of [Rabinovich et al. 2019].

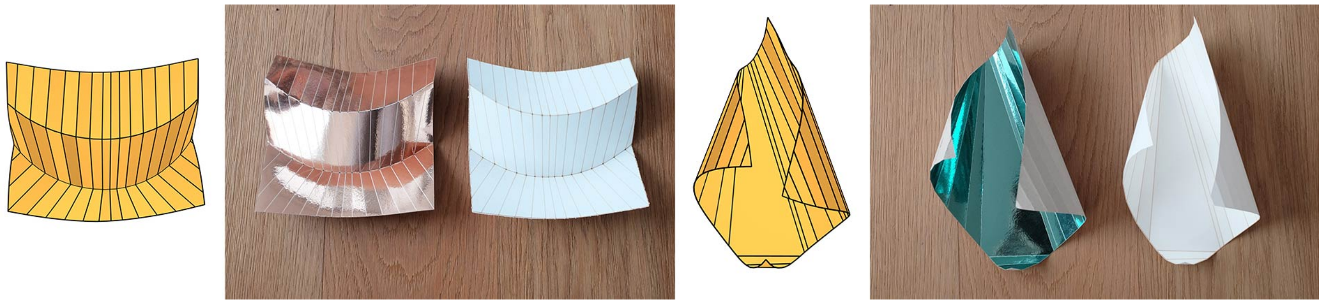

Fig. 19. Our output meshes can be physically fabricated from planar sheets of stiff material. For this experiment, we parameterize our output mesh to the plane and etch the flattened mesh edges into cardboard using a laser cutter. Appropriately bending the sheet of cardboard along the edges then gives a shape that matches our output.

lifts this limitation and provides a useful tool for designing developable surfaces for fabrication and architectural design applications.

In future work, it would be interesting to explore the use of our remeshing algorithm on non-developable input for the purpose of developable approximation. 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. Another interesting topic for future research would be the option to have adaptive output mesh resolution based on the local curvature, allowing a denser representation in more curved areas. Finally, it would be interesting to explore the utility of feeding back our optimized rulings in order to optimize or even subdivide the input discrete model.

ACKNOWLEDGMENTS

The authors would like to thank Michael Rabinovich and Helmut Pottmann for illuminating discussions, and Caigui Jiang and Chengcheng Tang for providing us with models from their previous work.

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.

Alexander I Bobenko and Ulrich Pinkall. 1999. Discretization of surfaces and integrable systems. *Oxford lecture series in mathematics and its applications* 16 (1999), 3–58.

Alexander I. Bobenko and Yuri B. Suris. 2008. *Discrete differential geometry: integrable structure.* Graduate studies in mathematics, Vol. 98. American Mathematical Society, Providence (R.I.).

Alexander I Bobenko and Sergey Tsarev. 2018. The curvature line parametrization from circular nets on a surface. *J. Math. Phys.* 59, 9 (2018), 091410.

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).*

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

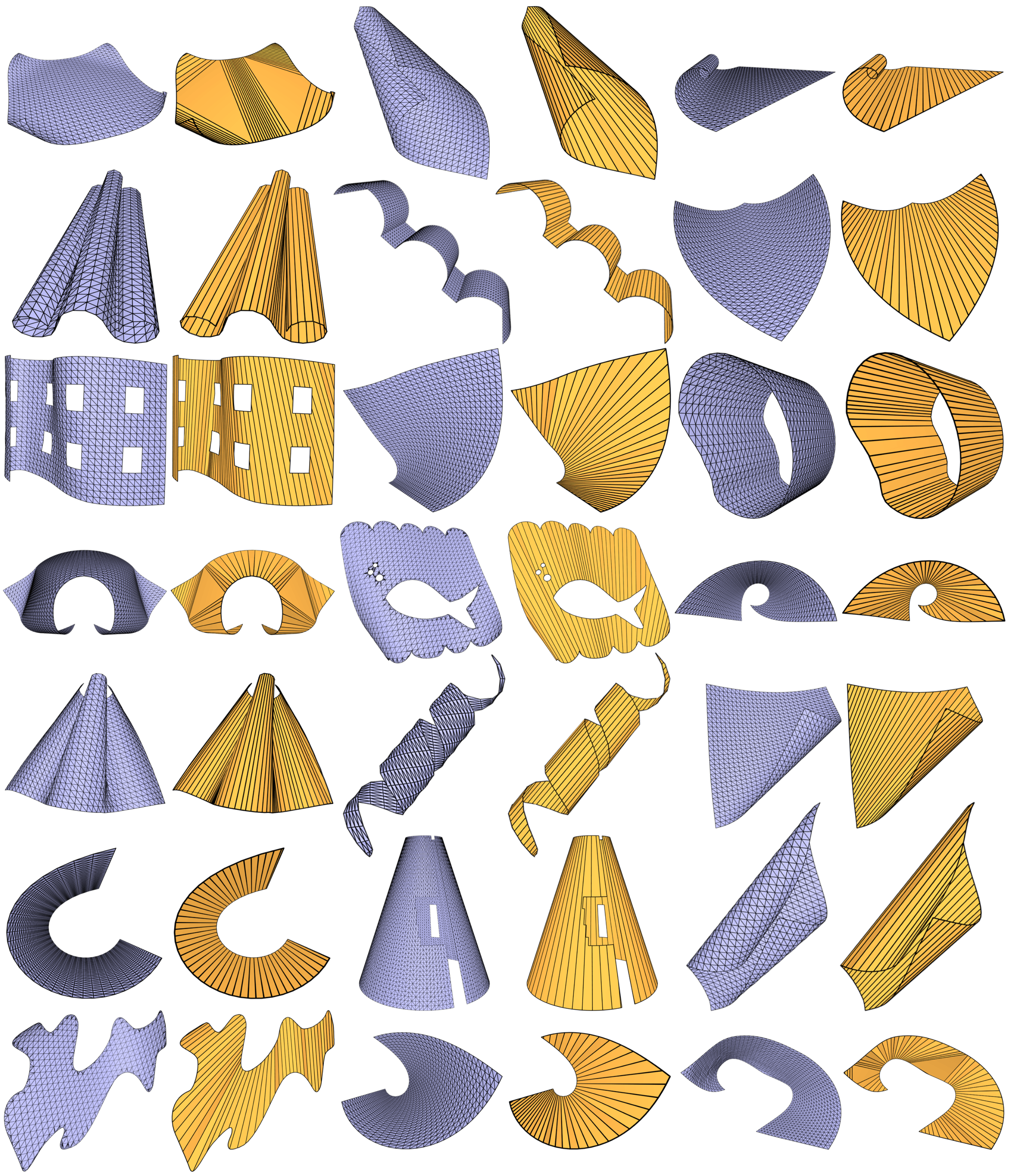
Fig. 20. Various remeshing results obtained with our method.

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

Manfredo P. do Carmo. 1976. *Differential Geometry of Curves and Surfaces.* Prentice-Hall.

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.

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.

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Globally optimal direction fields. *ACM Trans. Graph.* 32, 4 (2013).

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

Justin Solomon, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2012. Flexible developable surfaces. *Comput. Graph. Forum* 31, 5 (2012), 1567–1576.

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. 2017. Directional: A library for Directional Field Synthesis, Design, and Processing. https://doi.org/10.5281/zenodo.3338174

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 2. 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 $|\mathcal{F}'|$, 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.	$\lvert\mathcal{V}'\rvert$	$\lvert\mathcal{F}'\rvert$	$p_{\max}$ [%]	p_{mean} [%]	h[%]
1	204	63	1.26	0.28	0.78
4	270	74	1.80	0.41	0.67
6	180	51	1.53	0.28	0.45
6	206	44	0.59	0.15	0.85
6	218	50	0.88	0.18	1.07
6	190	48	0.95	0.19	2.55
8	190	48	0.95	0.19	2.55
9	208	45	0.41	0.11	0.66
9	206	44	0.59	0.15	0.85
9	210	46	0.54	0.16	0.41
9	208	45	0.45	0.17	1.39
9	218	50	0.88	0.18	1.07
10	288	107	11.84	1.98	0.48
10	288	107	0.00	0.00	0.95
11	164	23	0.52	0.24	0.47
11	210	46	0.54	0.16	0.41
11	298	90	0.32	0.10	0.41
12	384	68	1.08	0.34	0.31
12	632	67	1.22	0.10	0.26
12	1130	66	0.25	0.02	0.24
13	176	29	3.64	1.04	0.56
13	156	19	5.42	2.51	1.34
14	254	68	3.83	0.42	0.63
14	254	68	3.82	0.42	0.66
14	258	70	3.34	0.48	0.21
15	202	100	0.14	0.05	0.20
15	324	100	0.46	0.14	0.22
15	858	274	1.66	0.09	1.15
15	966	272	1.14	0.08	0.13
16	306	54	0.15	0.04	0.15
16	438	54	0.14	0.04	1.91
16	578	93	2.55	0.30	1.09
17	288	170	0.19	0.05	0.09
17*	1970	567	0.82	0.05	0.04
17**	1947	636	0.26	0.05	0.01
17	738	211	0.84	0.10	0.14
18	258	98	1.16	0.24	0.23
18	243	80	0.67	0.20	0.62
18	280	115	1.98	0.16	0.35
18	232	82	2.24	0.32	0.42
18	672	134	0.34	0.06	0.52
19	196	52	2.39	0.37	0.26
19	210	28	3.49	0.43	1.47
20	254	68	3.63	0.41	0.65
20	210	46	1.25	0.19	0.23
20	204	43	7.18	0.50	0.34
20	212	47	0.56	0.33	0.60
20	390	70	3.93	0.31	0.67
20	204	43	0.69	0.14	0.10
20	422	88	3.35	0.56	0.85
20	208	45	0.43	0.10	0.46
20	226	63	0.40	0.12	0.18
20	328	75	0.78	0.18	0.69
20	1033	71	1.72	0.34	0.38
20	218	50	2.35	0.20	0.32
20	242	58	0.80	0.36	1.29
20	288	107	11.84	1.98	0.48
20	208	45	0.38	0.11	0.20
20	214	48	0.15	0.05	0.08
20	502	53	0.32	0.06	0.09
20	194	45	1.19	0.24	0.40
20	609	47	2.89	0.60	0.21
20	204	43	0.49	0.13	0.39
20	288	67	1.31	0.20	0.82

* Same model displayed from 3 viewing angles.
** Same model displayed from 2 viewing angles.