Fabricable Discretized Ruled Surfaces

HASSAN BAHARAMI, FEIT, University of Technology Sydney, Sydney, Australia
MICHAL PIOVARCI, ETH Zurich, Zurich, Switzerland
MARCO TARINI, Computer Science, Universita degli Studi di Milano, Milano, Italy
BERND BICKEL, ETH Zurich, Zurich, Switzerland
NICO PIETRONI, University of Technology Sydney, Sydney, Australia

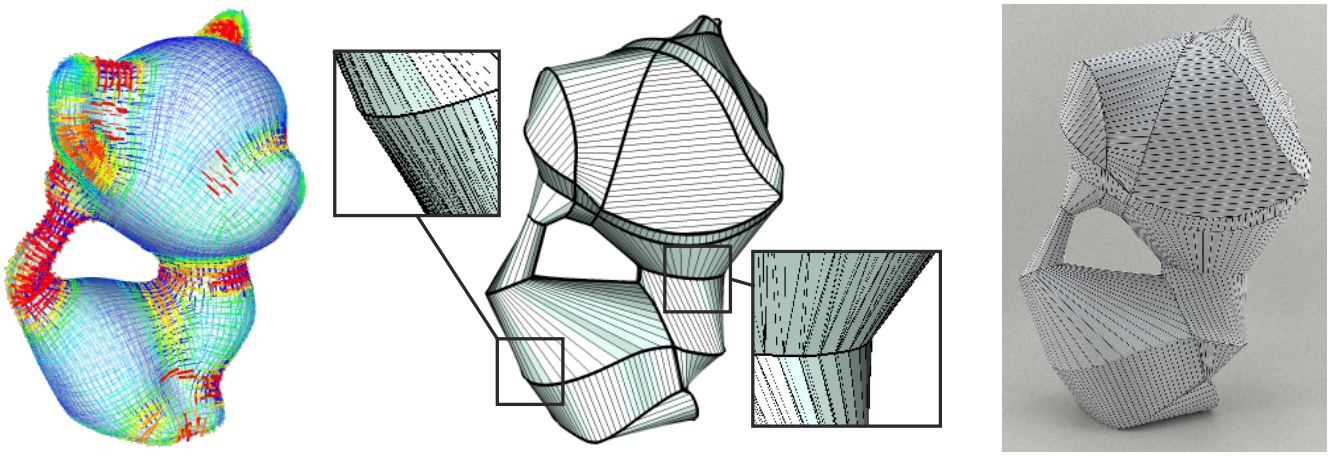


Fig. 1. Starting with a triangular mesh with a curvature-aligned cross-field (left), our system produces a decomposition into a seamless set of ruled surfaces (middle) that serves as a blueprint to easily fabricate the surface by means of inextensible materials, such as paper (right).

We present a method to automatically approximate a given surface with a small set of patches, each being a developable ruled surface featuring long-ruling lines. These construction primitives are attractive for their inherent ease of fabrication by cutting and folding inextensible materials and for their favo rable structural properties. Our algorithm strikes a good tradeoff between the simplicity of produced designs (in terms of the number and shapes of the patches) and approximation quality. To this end, it is guided by a smooth curvature-aligned cross-field.

Compared to traditional methods, we rely on final discretization steps to ensure the developability of the ruled surfaces and produce a fabricable layout, bypassing the need to enforce that the strips are strictly developable in continuous settings (which requires difficulty in enforcing geometric conditions). We demonstrate the effectiveness of the proposed algorithm by producing several viable designs and using them to physically fabricate various physical objects.

Authors' Contact Information: Hassan Baharami, FEIT, University of Technology Sydney, Sydney, New South Wales, Australia; e-mail: Hassan.Bahrami@student.uts.edu.au; Michal Piovarci, ETH Zurich, Zurich, Zürich, Switzerland; e-mail: piovarci@arch.ethz.ch; Marco Tarini, Computer Science, Universita degli Studi di Milano, Milano, Italy; e-mail: marco.tarini@unimi.it; Bernd Bickel, ETH Zurich, Zurich, Zürich, Switzerland; e-mail: bickelb@ethz.ch; Nico Pietroni, University of Technology Sydney, Sydney, New South Wales, Australia; e-mail: nico.pietroni@uts.edu.au.

ACM 0730-0301/2025/06-ART30
https://doi.org/10.1145/3734519

CCS Concepts: • **Computing methodologies** → **Computer graphics**; **Mesh geometry models**; **Parametric curve and surface models;**

Additional Key Words and Phrases: Fabrication, ruled surfaces, developable, surface approximation, patch decomposition

ACM Reference Format:
Hassan Baharami, Michal Piovarci, Marco Tarini, Bernd Bickel, and Nico Pietroni. 2025. Fabricable Discretized Ruled Surfaces. *ACM Trans. Graph.* 44, 3, Article 30 (June 2025), 15 pages. https://doi.org/10.1145/3734519

1 Introduction

Ruled surfaces are surfaces where each point belongs to a straight line on the surface, and can be understood as surfaces generated by the motion of a straight line (the ruling line, or rule) [Farin 1997]. They find many applications in shape manufacturing and architecture due to their favorable characteristics in terms of ease of construction, robustness, and visual appeal. For example, the rules can be implemented by straight beams connected by a textile (serving as a membrane connecting the beams) or by folding paper/cardboard along conveniently straight lines. Consequently, a useful problem is how to decompose a given shape into a small set of ruled surfaces, striving to maximize the geometric similarity and the length of the rules.

When the design is to be fabricated with inextensible materials (such as paper, cardboard, plywood, etc.), this already difficult task is made more complicated by the need to ensure that each ruling surface is also perfectly developable [O'neill 2006; Solomon et al. 2012; Rabinovich et al. 2019]. It is well known that, in order to be

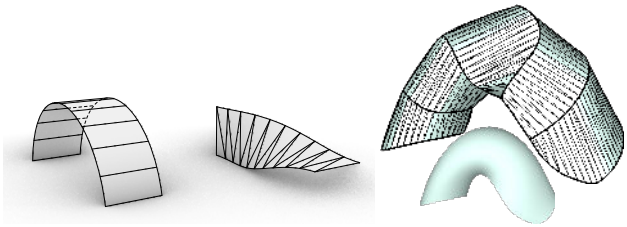


Fig. 2. A ruled surface (left) cannot feature torsion without compromising its developability. However, even a non-developable ruled surface featuring torsion becomes developable when discretized as a triangle strip (middle), making them better suited to approximate general shapes (right).

developable, a ruled surface must satisfy an additional constraint: all points along a given rule must share the same tangent plane [Stein et al. 2018]. In this sense, the task of decomposing a shape into ruled surfaces is one special case of the more general task of decomposing a shape into almost-developable patches, which is a difficult and deeply studied task in geometry processing [Stein et al. 2018; Verhoeven et al. 2022]. Unfortunately, guaranteeing developability while providing an automatic fabrication ready decomposition remains an open problem. Moreover, most method do not explicitly follow the natural curvatures of the input models, yielding to visually non-satisfactory approximations.

Developability via remeshing. In this work, we take a different route. We observe that the discretization of the developability problem inherently changes the constraints: any ruled surface, even a non-developable one in the continuous setting, can be trivially discretized into a developable surface by, first, electing a set of rules along the ruled surface, which splits the ruled surface into a set of (non-necessarily flat) rectangular shapes, then splitting each rectangle diagonally, adding new rules (see Figure 2). This produces a triangle strip that, as such, is developable by construction, because all triangles are flat and all vertices are on the boundary. Where the CAD community used this observation for the physical realization of individual disc-shaped patches [Wang and Elber 2014; Wang and Tang 2005], we used it as a basis for a new method for derivation of developable patch layouts. By relying on this final discretization step, our construction process can disregard the developability constraints, and this additional freedom potentially results in better decompositions (see Figure 2).

Specialized patch layouts. The discretization step guarantees full developability, but also makes the final shape a direct function on the patch boundaries. Consequently, we need a patch decomposition where each patch not only admits a general low-distortion parameterization, but also produces a low geometric error with the intended final discretization (which guarantees *no* distortion). By explicitly accounting for this additional objective we outperform generic patch-layout construction methods in terms of approximation error, as we experimentally verify (see Section 4).

Our method, in short, consists in searching, among plausible candidates, for a layout in which the final discretization will introduce low geometric deformation. The evaluation of a candidate layout includes an *explicit* estimate of the geometric error introduced by the final discretization, measured as the

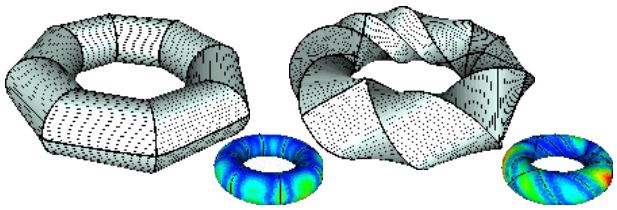


Fig. 3. Alignment of the ruling lines to principal curvature direction typically reduces the approximation error, as shown in this torus example. $\mathcal{E}$ is the Hausdorff distance, and ε is the average distance (both expressed as a percentage of the bounding-box diagonal).

deformation needed to straighten the potential ruling lines; standard distortion-minimization parametrization techniques are used solely to identify the potential set of ruled lines. Candidate layouts are generated by a novel heuristic that iteratively traces patch boundaries over the mesh, segmenting it into patches. This heuristic is guided by design principles, such as ensuring sufficient spacing between boundary lines.

Curvature alignment. One such design principle, specific to ruled surfaces, is that rule lines should align to the local direction of minimal curvature, thus exploiting the ability of each ruled surface to bend freely in the other direction to follow a target shape. This implies the need for patch boundaries to be aligned to the principal curvature directions, as illustrated in Figure 3. To this end, we trace our patch boundaries along a tangent direction field, which is precomputed to align to curvature directions. We empirically verify the practical importance of curvature alignment, which sets us apart from methods not designed to adhere to this principle (see Figure 31).

2 Related Work

We start with a review of developability in the context of manufacturing. Next, we summarize recent advances in the automated generation of developable meshes. Finally, we provide an overview of patch decomposition approaches.

2.1 Developability in Manufacturing

Developable surfaces have a long tradition in manufacturing. Their applications range from individual components [Lee and Bo 2016], through garments [Wang and Tang 2004; Rose et al. 2007], to architecture [Liu et al. 2006; Wang et al. 2019; Jiang et al. 2020b] and simulation [Schreck et al. 2016]. In this context, the key benefits of developable surfaces is that they allow for fast and cost-effective fabrication. Through the use of numerically controlled cutters or pre-fabricated forms, a developable surface can be quickly manufactured from a wide range of materials [Takezawa et al. 2016; Konaković et al. 2016; Schüller et al. 2018; Zhang et al. 2019; Jiang et al. 2020a]. When the material can undergo small but non negligible amounts of deformations (e.g., by producing wrinkles), it is sufficient to approximate the developability of the surface, like for example explored in Wang [2008]; this is not the case we focus on.

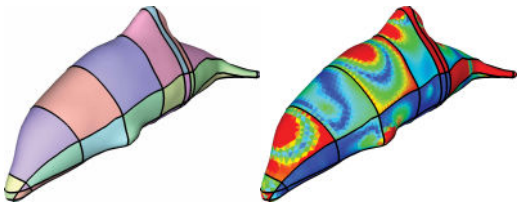


Fig. 4. Existing curvature-aligned patch decompositions, such as the one proposed by Pietroni and colleagues [Pietroni et al. 2021], combined with surface flattening techniques like as-rigid-as-possible parametrization [Liu et al. 2008], are generally not suitable for fabrication, as they do not guarantee full developability. The red regions indicate areas where stretching or compression exceeds 5% of the original length.

Once the individual flat pieces are manufactured, they need to be assembled to form the final three-dimensional shape. There are two common techniques: folding and joining. Folding relies on plastic deformation of bulk material [Kilian et al. 2008; Noma et al. 2020; Tahouni et al. 2020]. It is a technique often used in origami [Mitani and Suzuki 2004; Shatz et al. 2006; Massarwi et al. 2007; Dudte et al. 2016] and recently even to manufacture functional three-dimensional objects [Freire et al. 2023]. As an assembly process, folding has two key disadvantages. Since each element needs to be bent separately, there is only a limited number of segments that can be practically assembled. Moreover, as the segment size grows, folding becomes significantly more challenging. In contrast, joining two shapes, even on an architectural scale, is a relatively simpler process [Wang et al. 2019; Jiang et al. 2020b; Gavriil et al. 2020]. However, for practical purposes, the patch size plays a critical role in joined assemblies. A larger patch size is tempting as it reduces the overall number of joints. However, from a certain size and scale, large patches become unwieldy. In our method, we take both of these considerations into account. We optimize for a small number of patches to facilitate the assembly. However, we also enforce that each patch bends only along its ruling lines, which facilitates fabrication. For more information on developability and its applications in manufacturing, we suggest the survey of Yuan et al. [2023].

2.2 Developing Discrete Meshes

Developability is thoroughly studied in geometry processing, with many generative algorithms proposed for various purposes [Bo and Wang 2007; Tang et al. 2016; Rabinovich et al. 2018a, b, 2019]. In the context of fabrication with inextensible materials, this problem is often cast as the search for the closest perfectly developable approximation of an input discrete mesh. Methods in this category range from convex optimizations on constrained inputs [Sellán et al. 2020], through wrapping the object with developable sheets [Ion et al. 2020], segmenting the input into developable regions [Zhao et al. 2022], using genetic algorithms [Zhao et al. 2023] or or neural networks [Wu et al. 2023]. The main drawback of these methods is that they do not consider the fabricability of the final patch decomposition. More specifically, these methods enforce only the flattenability of the generated patches. However, consider as an example a crumpled piece of paper. Although flattenable by definition, it would be very difficult to reproduce exactly [Stein et al. 2018]. This observation led to the development of methods that improve

Table 1. A Table Providing an Overview of State-of-the-Art Methods and Their Respective Capabilities

Method	Curv Align	Full Develop	Automatic Layout
[Tang et al. 2016]	(✓)	✓	✗
[Schüller et al. 2018]	✗	✓	✓
[Stein et al. 2018]	(✓)	✗	✓
[Zhao et al. 2023]	✗	✗	✓
[Verhoeven et al. 2022]	✓	(✓)	✓
[Zhao et al. 2022]	✗	✗	(✓)
Proposed	✓	✓	✓

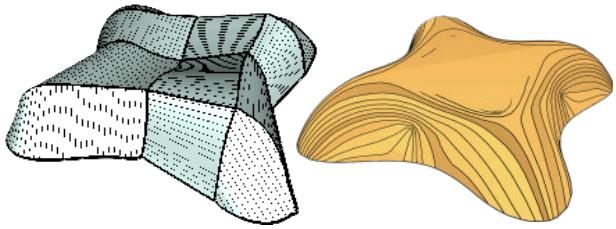


Fig. 5. Comparison of our method with Dev2PQ [Verhoeven et al. 2022]. Although Dev2PQ (right) produces, like we do, a perfectly developable discretization (under some assumptions), it is not designed to produce a viable patch decomposition for general surfaces that are not already developable in the continuous sense, unlike our method (left). Right image courtesy of Verhoeven et al. [2022].

the flattenability criterion with a constraint on the ruling lines of the surface [Stein et al. 2018; Binninger et al. 2021]. Our method also belongs to this category and aims to produce patches with non-intersecting, uniformly spaced ruling lines. The key difference with that line of previous work is that it does not provide patch decompositions. Their output is triangular meshes composed of developable regions. Generating a fabricable patch decomposition is highly nontrivial [Ion et al. 2020]. In contrast, our method is built on generating patches from the ground up. As a result, our outputs are ready for the fabrication by means of two-dimensional ruled blueprints.

2.3 Patch Decomposition

Flattening a three-dimensional surface into a planar domain is a non-trivial task, cast as the minimization of a stretch energy that penalizes flattening deformations. Flattening a general surface can produce large distortions [Yoshizawa et al. 2004] (and also self-intersections in the 2D domain [Rabinovich et al. 2017]). As a countermeasure, the input mesh can be cut into smaller patches [Poranne et al. 2017; Li et al. 2018; Sharp and Crane 2018]. For more details about patch decompositions, we refer the reader to the survey of Campen [2017].

State-of-the-art techniques, like ours, produce cuts by following a tangent vector-field defined on the surface, that can be in turn aligned to main curvature directions [Vaxman et al. 2016; Livesu et al. 2020; Nuvoli et al. 2019; Pietroni et al. 2016; Razafindrazaka et al. 2015; Pietroni et al. 2022, 2021](in particular, our proposal extends the robust cut construction procedure from Pietroni et al.

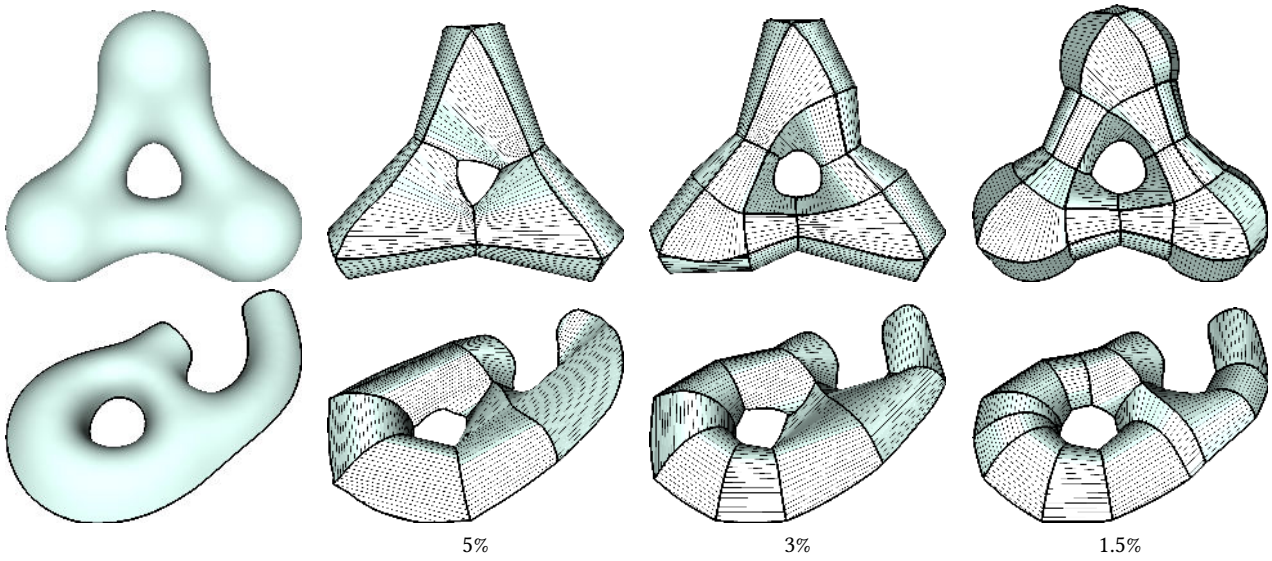


Fig. 6. Results obtained with different error thresholds $e_{\max}$ (expressed as fraction of the bounding-box diagonal of the original mesh).

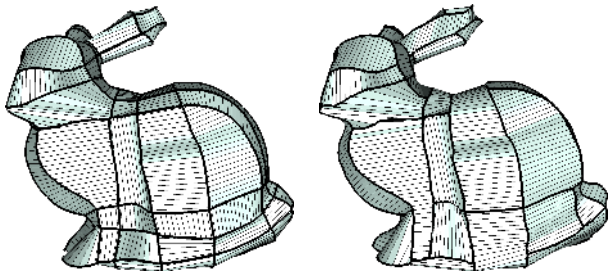


Fig. 7. An example patch decomposition with (left) and without (right) T-junctions.

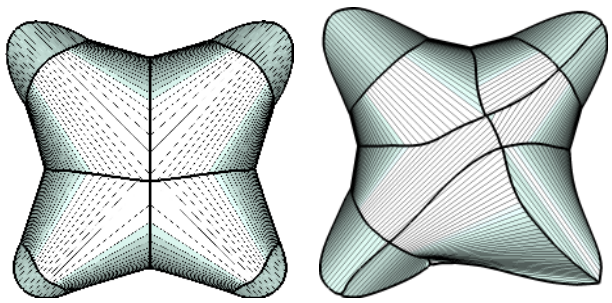


Fig. 8. An example of results with (left) and without (right) symmetry preservation.

[2021]). This policy is beneficial, among other things, to encourage cut straightness and shortness. Unfortunately, existing techniques in this class cannot be directly applied to our case. The typical residual stretch energies are way too large for the patch layout to serve as a fabrication plan for inextensible materials (see Figure 4). More importantly, the minimized stretch energy correlates only with the distortions that would be necessary to morph the patches into generic developable surfaces (such as curled pieces of paper). In contrast, we optimize the patch layout by predicting and minimizing the deformations that will be necessary to morph the patches into developable *ruled* surfaces.

2.4 Discussion

In Table 1, we provide an overview of the various methods and their main characteristics with respect to key aspects of the developable patch decomposition. These aspects include their ability to align patch borders and ruling lines to curvature directions, their ability to ensure full developability, and whether they include a method to automatically derive a fabricable patch layout. Among the existing methods that ensure strict developability, Dev2PQ [Verhoeven et al. 2022] is the only one that, like ours, uses a curvature field to guide the ruling lines. However, Dev2PQ is designed to work on surfaces that are already developable, up to discretization.

The comparison in Figure 5 shows that Dev2PQ is not suitable for developability-approximation of general surfaces.

3 Method

We seek to decompose the input target surface into a small set of *patches*, that can be well approximated by ruled surfaces; then, we produce a developable ruled surface for each patch; finally, each patch is flattened isometrically into a 2D region, with ruled lines mapped over straight segments traversing the region from side to side.

The set of the 2D regions serves as a blueprint that can be used to physically fabricate the input shape (possibly assisted by automatic cutting and guided by printed or pre-folded ruling lines).

The system is governed by a user-defined upper bound on the approximation error $e_{\max}$, which serves as a parameter to balance between the simplicity of the design (in terms of the number of the ruled surfaces) and the geometric adherence with the target shape. Figure 6 illustrates the effects of this parameter.

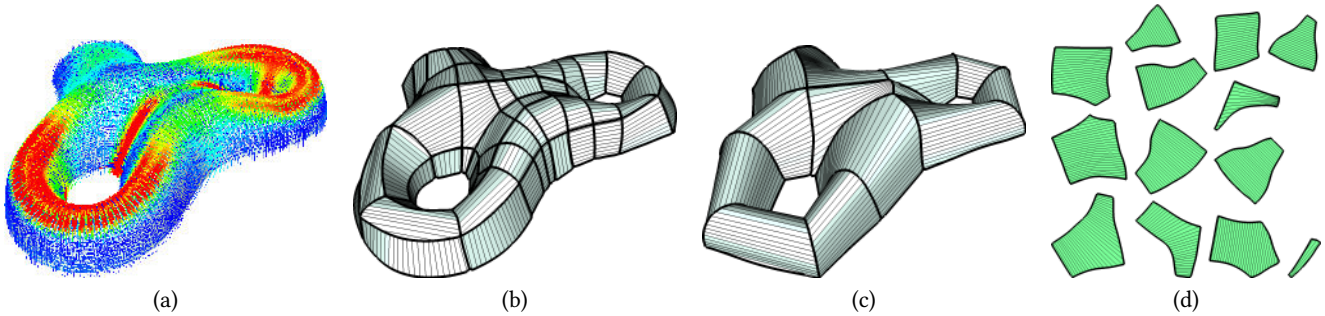


Fig. 9. A depiction of our method: (a) the input mesh, with a curvature-aligned smooth cross-field (red color indicates areas with strong curvature); (b) the result of the patch densification; (c) the outcome of the patch simplification step; and (d) culminating in the final patch layout.

Variant: T-junctions. Our patch layout can be opted to feature occasional T-junctions, which generally reduces the number of patches; otherwise, a strictly conforming layout can be requested. Figure 7 shows a comparison of this choice.

Variant: symmetry preservation. If the input surface exhibits a reflection symmetry, we can enforce its preservation by splitting the surface along the symmetry plane, executing our method on one side, and mirroring the results. Figure 8 shows a comparison of this choice.

3.1 Objectives

For a solution to serve as a fabricable design, it must meet several *strict* requirements:

Bijective developability In order to be realizable by cutting and bending a flat sheet of material, patches must admit a fully *isometric* and *bijective* mapping into a 2D region. Bijectivity implies that the 2D image of the patch must be free from overlaps.

Straight Ruling lines The ruling lines of each patch must be perfectly straight (both in 2D and 3D).

Water-tightness Neighboring patches must meet exactly at their boundary, without gaps.

In addition, a good solution must strike a good tradeoff between two conflicting objectives (controlled by the parameter $e_{\max}$):

Design simplicity To ease fabricability, we want the number of patches to be limited and the shapes to be simple.

Geometric fidelity The patch layout must approximate the target shape, within the predefined error tolerance.

About design simplicity. As a rule of thumb, the complexity of the realization grows with the number and length of the boundaries, which will require cutting (in 2D) and gluing (in 3D). Therefore, we want to avoid unnecessary patches or small patches, whenever not necessary to approximate the shape.

For the same reason, the boundaries of the patches should be preferably smooth and short. In our decomposition, the 2D patches are polygon-like shapes delimited by a small set of smooth curved lines. Additionally, ideal boundary lines are either approximately aligned to the rules of the patch or approximately orthogonal to them.

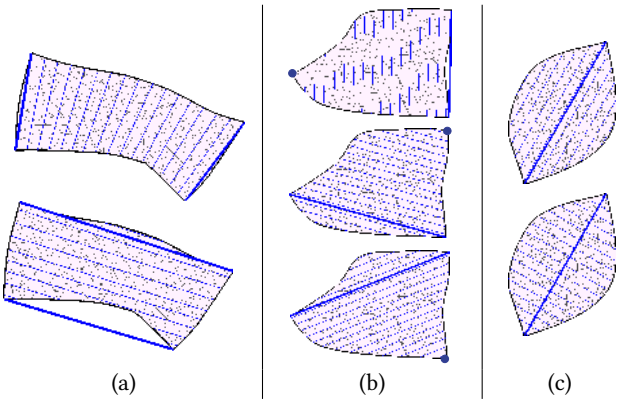


Fig. 10. Examples of candidates for ruling lines considered for (a) a four-sided patch, (b) a three-sided patch, and (c) a two-sided patch. The patches are shown in 2D parametric space. Candidates are ranked by considering how well they preserve their straightness when mapped back into 3D space.

Finally, we need patches to have a **disk-topology**, that is, to feature a single boundary; this disallows, for example, circular patches, or holed patches, which would hinder or considerably complicate most construction processes.

About geometric fidelity. The objective of geometric fidelity implies some degree of **curvature alignment**. By their nature, ruled surfaces can freely curve only in the direction orthogonal to the ruled lines. Therefore, it is beneficial for rules to be aligned to minimal curvature directions, so as to be free to bend along the maximal curvature direction. To this end, we employ a curvature-oriented cross-field to guide our construction process.

3.2 Overview

Our input is a two-manifold triangle mesh with a curvature-oriented tangent cross-field (a smooth 4-rotational-symmetric tangent vector field [Vaxman et al. 2016]).

Patch layout construction (Section 3.4). We populate the mesh with a network of intersecting *paths*, which split the input surface into patches. Paths are field-aligned lines either closed in a loop or terminating at the mesh boundaries. In a first cycle, we add paths over the mesh, one by one, until the resulting patch layout is determined to be *viable*. At each step, we trace a new path, striving

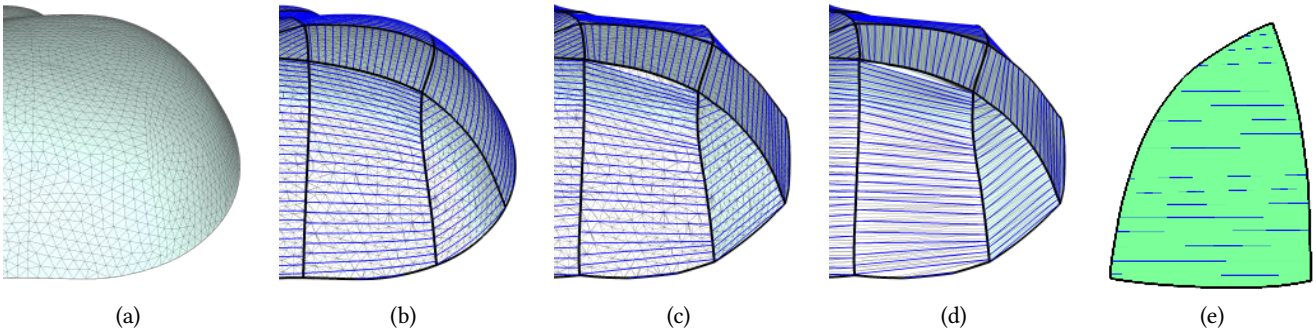


Fig. 11. A summary of meshing used during the entire process. Edges representing ruling lines are in blue, and thick black lines are edges forming the patch boundary: (a) Original meshing, after path smoothing (Section 3.4); temporarily re-meshed surface before (b), and after (c), the Morphing into approximate ruled surfaces (Section 3.3.2); (d) final remeshing of the patches (Section 3.5); and (e) the final 2D layout of one patch.

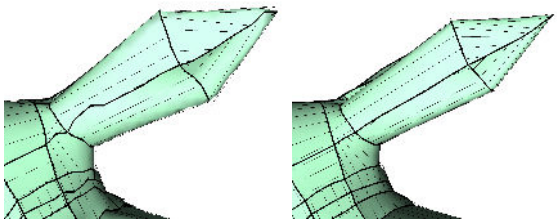


Fig. 12. The effect of the energy term E_{smooth} on the boundary shapes (left: without; right: with).

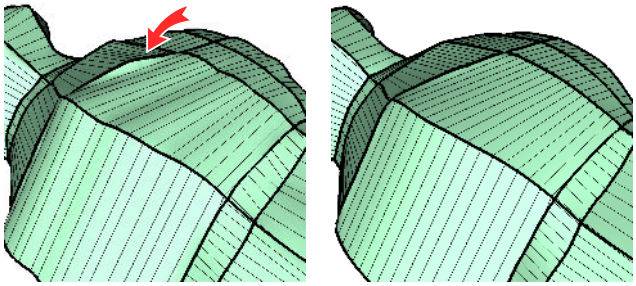


Fig. 13. Patches can bend abruptly over ruling lines are orthogonal, even when the target surface is flat (left). This undesired occurrence is prevented by introducing the energy term E_{flat} (right).

to address one remaining problem as detected in the current layout. As this procedure is greedy, a path can be made redundant by subsequently added paths. So, in a second cycle, we test all paths for removal, in inverse order of creation: the tested path is removed if the layout is determined to still be viable without it. Figures 9(b) and (c) show an example of the layouts after either cycle.

Patch-layout evaluation (Section 3.3). At the core of the above process, we must assess whether a given layout is viable. In addition to topological checks, this is done by attempting to morph the mesh into a spatial configuration that *approximately* meets all the geometric requirements described in Section 3.1). If no such morphing is found that adheres to the maximally tolerated displacement $e_{\max}$, then the configuration is determined not to be

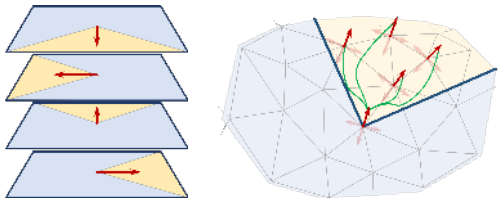


Fig. 14. We frame the problem of tracing field-aligned paths as the minimal path over a graph with four nodes for each vertex, one for each tangent direction of the cross-field (left). The graph arches are defined by connecting nearby nodes with matching cross-field directions (right).

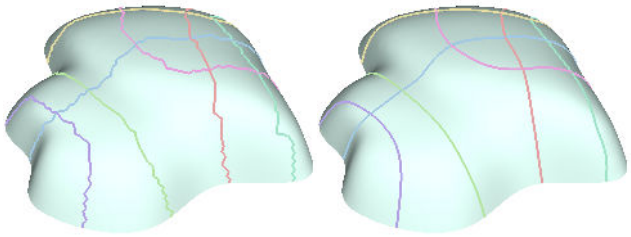


Fig. 15. Traced path as identified as minimal paths over the mesh graph (left), and after the smoothing phase.

viable. Importantly, the requirements need only to be approximated in this phase, allowing us to impose them as soft constraints that can be easily optimized globally.

Final discretization (Section 3.5). Finally, we triangulate each patch into one stripe of long and thin flat triangles, with ruling lines represented as edges. This final re-meshing meets most of the requirements (such as rule straightness and water-tightness) in full and by construction, that is, in virtue of its polygonal connectivity alone; notably, this also applies to developability as well, as observed in Section 1. At the same time, because we already determined that the 3D mesh can be morphed in a configuration close to fulfilling these requirements, we know that this final remeshing introduces a geometric error that is, at most, close to the prescribed tolerance. Finally, the patches are trivially flattened (without any distortion), obtaining a final 2D layout ready to be cut, bent, and assembled.

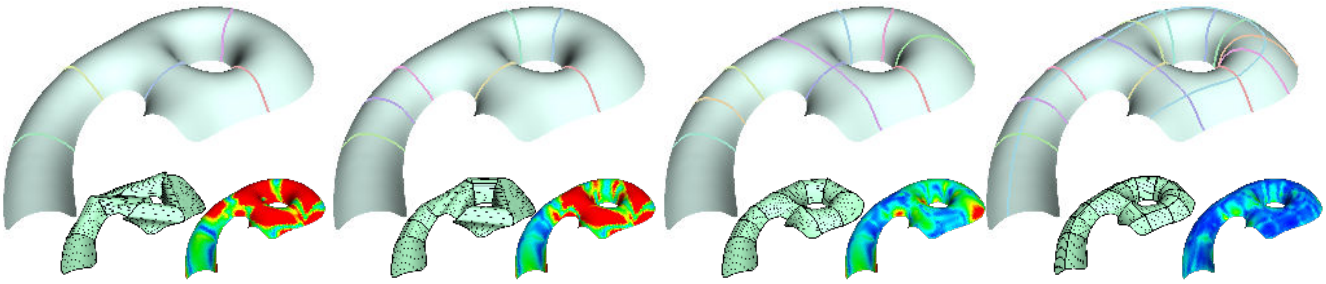


Fig. 16. A few steps of path insertion with developable 3D embedding and relative approximation error.

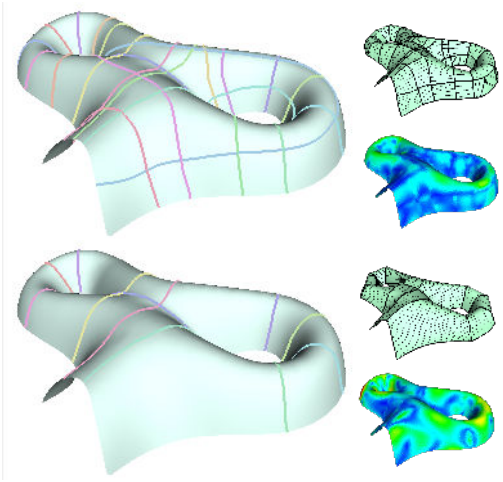


Fig. 17. Final steps of path removal, with the relative approximation error.

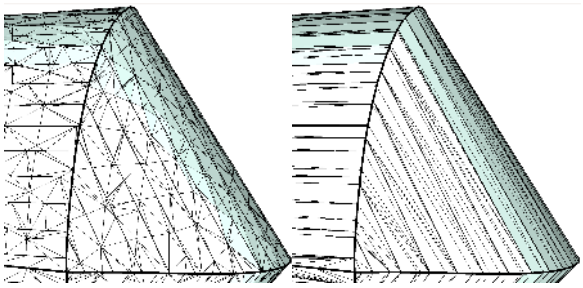
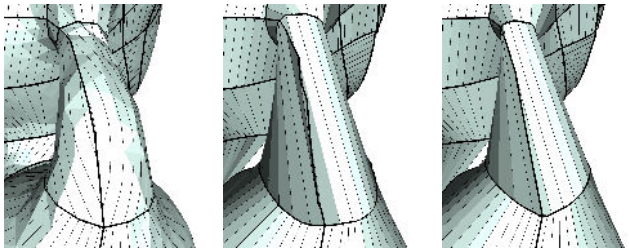


Fig. 18. The morphed original mesh (left) and the final discretization (right).

3.3 Patch-Layout Evaluation

To steer the patch layout construction, we need to estimate if the given patch layout is viable: to be viable, a patch layout must satisfy topological and geometric conditions. This assessment must be efficient, because it is repeated after each path insertion or attempted removal.

Topological conditions. Given a set of intersecting paths, each path is split into *arches* by intersection with other paths. The arches divide the mesh into patches; each patch is delimited by a number of arches, touching at the corners of the patch.



(a) (b) (c)

Fig. 19. Final shape optimization: before the morphing, ruling lines traced over a highly non-developable surface (a) can adversely affect the regularity of the patch boundary after the morphing (b); this is countered by final patch-shape optimization (c).

We check that each patch is topologically equivalent to a disk; then, we check that no patch exceeds n distinct arches on its boundary, because that would contradict the objective of simplicity. Guided by empirical evaluation, we used $n = 6$. If either condition is not met, the layout is deemed non-viable.

Geometric condition. To be viable, a patch layout must also admit a sufficiently good approximation as a set of valid ruled surfaces. To estimate this, we attempt to globally morph the surface in a way that (approximately) deforms each patch into a valid ruled surface. This is cast as a global optimization problem, minimizing the energy that penalizes the introduced displacements and promotes the geometric characteristics of ruled surfaces. The minimizer of this energy is then tested: if any introduced displacement exceeds $e_{\max}$, it is an indication that the set of patches cannot be forced into a set of ruled surfaces within the prescribed tolerance, and the layout is deemed non-viable.

To define this morphing, we first need to determine the ideal orientations for the rules in each patch, as follows.

3.3.1 Choosing the Ruling Lines. The task of determining an optimal set of ruling lines over a given patch has been tackled in prior work. The method proposed in Dev2PQ [Verhoeven et al. 2022] involves extracting straight lines from a level-set function defined by a curvature-aligned line field. This approach is elegant and optimal, but its applicability depends strongly on the presence and distribution of field singularities. In addition, this solution is not efficient enough for our purposes. Instead, we

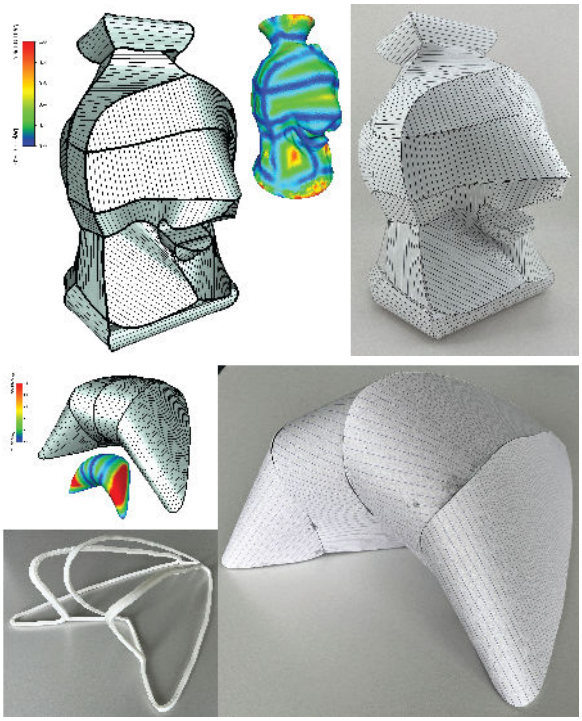


Fig. 20. Top: Fabricated lion statue composed of several complex patches exhibiting various degrees of bending and twisting. Bottom: Thin rod-like scaffolds are used to reproduce an architectural piece by attaching large pieces of developable material to cover the roof.

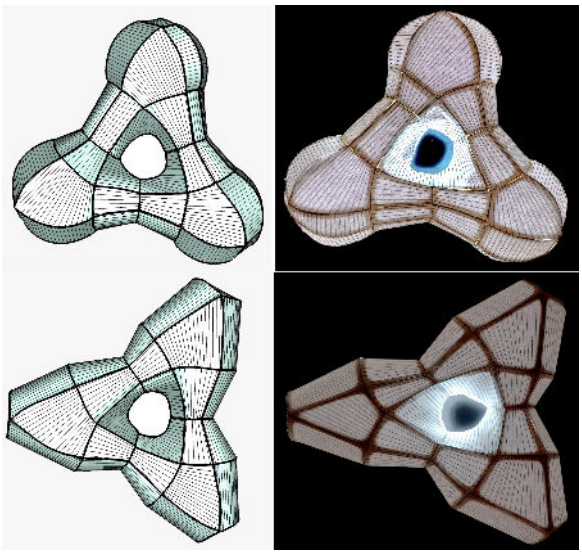


Fig. 21. Fabricated paper lamp.

have devised a different approach, that prioritizes robustness and efficiency.

We parameterize the patch in 2D, using As-Rigid-As-Possible parametrization [Liu et al. 2008], which is efficient and finds the most isometric mapping. If this mapping (which is unique up to 2D rotation and translations) presents self-overlaps, the patch layout is deemed not viable, as it infringes the condition of bijectivity (self-overlaps are detected by looking for self-intersections of the patch boundary).

Otherwise, we consider a number of candidate sets of ruling lines, each defined by a family of straight lines in parametric space. The parametrization maps each corner of the patch into a 2D position. We consider the 2D polygon defined by these n positions: for each pair of opposite sides, we generate a sequence of 2D segments sweeping the space between those two sides at regular intervals (see Figure 10(b)). If the number of corners is odd, we also generate a sweeping between each edge and its opposite corner (see Figure 10(b)). When the polygon has only two corners, we generate two candidate sets: one in the direction connecting the two corners and the other in the orthogonal direction (see 10(c)). Each line is intersected with a 2D image of the patch, giving a 2D segment traversing the patch side to side.

We pick the candidate family of lines that best preserves the straightness of all lines when mapped back to 3D space. We measure this by splitting each 2D line regularly into a number of small subsegments and averaging the norms of the cross products between consecutive seg-segments.

3.3.2 Morphing into Approximate Ruled Surfaces. Next, we want to find a morphing of the surface into a set of ruled surfaces that satisfies the necessary conditions. Strict adherence to these conditions is not necessary, as this is done solely for the purpose of evaluating the potential layout.

As a preliminary, we temporarily re-mesh the input surface, so that rules are now represented as collections of mesh edges (without affecting the shape of the surface). This is done by taking advantage of the parameterization found in the previous phase. Each edge intersecting a ruling line in 2D is split, inserting a new vertex at the intersection point; vertices at the boundary of all patches are all duplicated so that each patch becomes a disconnected component of the mesh. Figure 11(b) shows an example of this remeshing.

In the following, we group the vertices into three sets: vertices on the ruling lines $\mathbf{R}$, vertices on the border of a patch $\mathbf{B}$, and internal vertices that do not belong to any ruling line $\mathbf{I}$.

We define an energy E_{tot}, as a squared function of the positions of the vertices, designed to penalize a number of undesirable configurations; E_{tot} is the sum of a number of terms, as follows.

Approximation faithfulness term. A term E_{approx} measures the introduced deformation (thus the approximation errors), penalizing the distance of each vertex $\mathbf{v}_i$ from its original position $\mathbf{v}_t$:

$$E_{approx} = \sum_{\mathbf{v}_i \in \mathbf{R}} ||\mathbf{v}_i - \mathbf{v}_t||_2^2. \tag{1}$$

Rule straightness term. To encourage straightness of the ruling lines, we add a term that dictates that, for every triplet of consecutive vertices on a ruling line $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k \in \mathbf{R}$, the position of the middle one $\mathbf{v}_j$ must be a linear interpolation of the other two. The interpolation weight α is a constant determined by the parametric positions in 2D of the three vertices:

$$E_{straight} = \sum_{\text{consecutive } \mathbf{v}_{i,j,k} \in \mathbf{R}} ||\alpha \mathbf{v}_i + (1-\alpha)\mathbf{v}_k - \mathbf{v}_j||_2^2. \tag{2}$$

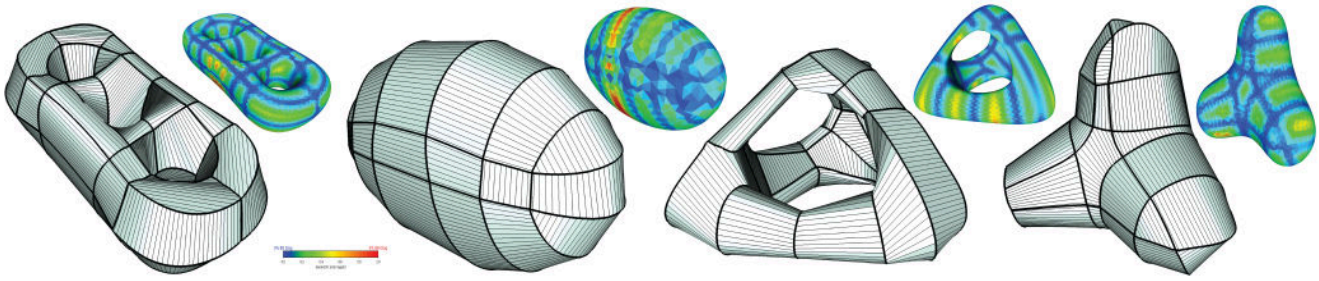


Fig. 22. The layout of ruling patches for a selection of regular geometries. Measured distances from the original mesh are reported. The maximum error is highlighted in red, corresponding to 5% of the input meshes bounding box.

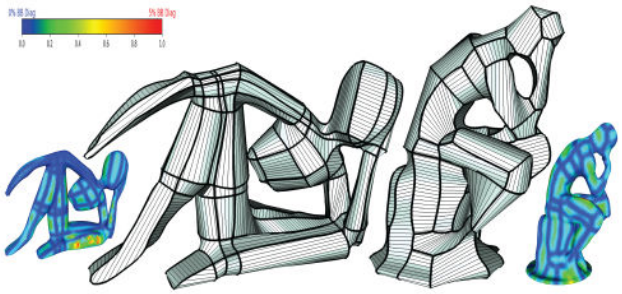


Fig. 23. Discretized versions of two complex meshes showcasing the robustness of our patch generation.

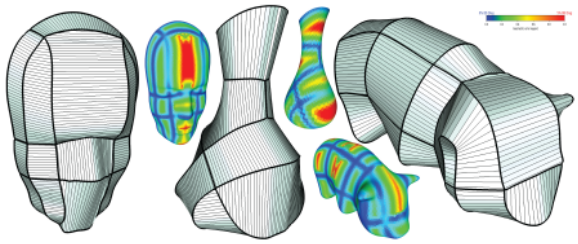


Fig. 24. The layout of ruling patches for a selection of general models.

Local planarity term. The remaining vertices should lie in the plane defined by their neighbors. An additional energy term $E_{laplace}$ strives to position each vertex $\mathbf{v}_i$ in $\mathbb{B}$ or $\mathbb{I}$ in the weighted average of its 1-ring neighbor $N(\mathbf{v}_i)$, defined by the cotangent weights [Pinkall and Polthier 1993] as computed in the 2D layout.

$$E_{laplace} = \sum_{\mathbf{v}_i \in \mathbf{B}} ||\mathbf{v}_i - N(\mathbf{v}_i)||_2^2. \tag{3}$$

C^0 continuity term. To preserve C^0 continuity between patches, it is necessary that each boundary vertex $\mathbf{v}_i \in \mathbf{B}$ is in the same position as its matching vertex $\mathbf{v}'_i$ on the boundary of a neighboring patch. This not being the case is penalized by the energy term:

$$E_{seams} = \sum_{\mathbf{v}_i \in \mathbf{B} \cup \mathbf{I}} ||\mathbf{v}_i - \mathbf{v}'_i||_2^2. \tag{4}$$

Boundaries smoothness term. To encourage the smoothness of the patch boundaries, we add an additional energy terms, E_{smooth}, as the bi-Laplacian smoothing term computed on sequences of boundary vertices. Without this term, the patch boundary can morph into unnecessarily wiggly lines, as shown in Figure 12.

Bending avoidance term. Another potential problem is that a patch can bend abruptly, folding along a ruling line. This configuration does not infringe the definition of ruled surfaces, nor its developability, but can still be undesirable for a number of reasons (including ease of construction or aesthetics), when the sudden bent is not found in the target surface. Figure 13, left, shows one example. To address this, we add an energy term, E_{flat}, that promotes the flatness of the patch for such reason. E_{flat} is defined in the same way as Equation (2), but acts on triplets of vertices $\mathbf{v}_{i,j,k}$ on three *different* consecutive rules. This term should not trigger in areas where the original surface actually bends abruptly. Therefore, we weigh the contribution for each triplet according to the local isotropy factor, the middle vertex $\mathbf{v}_j$, defined as one minus the ratio of the magnitudes of the associated minimal and maximal curvature directions.

The total energy is the sum of all the above terms, each weighted by constant factors reflecting their relative importance:

$$\begin{aligned} E_{tot} = E_{approx} + E_{laplace} + 10^5\ E_{straight} \\ + 100\ E_{seams} + E_{smooth} + 2\ E_{flat}. \end{aligned} \tag{5}$$

The weights are chosen such that the ruling lines are forced to be straight and the boundaries to match, as both $E_{straight}$ and E_{seams} have high values. The other weights were experimentally determined. Figure 11(c) shows an example of the resulting morphing.

3.4 Patch Layout Creation

Our patch creation strategy requires tracing a sequence of paths oriented with the cross-field.

Tracing Field-Oriented paths. To trace paths aligned with the field, we adopt the efficient graph-based methodology adopted in Nuvoli et al. [2019] and Pietroni et al. [2021], which we summarize for completeness. A graph is constructed with four nodes for each vertex, each aligned to one cross-field direction, see Figure 14. Then, the problem of path tracing as a shortest-path search, where a designated *source node* is connected to one of potential *destination nodes* within the graph. We trace two types of paths: those connecting border-to-border and those forming loops. When tracing a *loop* from a specific vertex, an internal node is selected as the source, with the objective of returning to the same node. For border-to-border paths, a source node is chosen on a boundary vertex whose

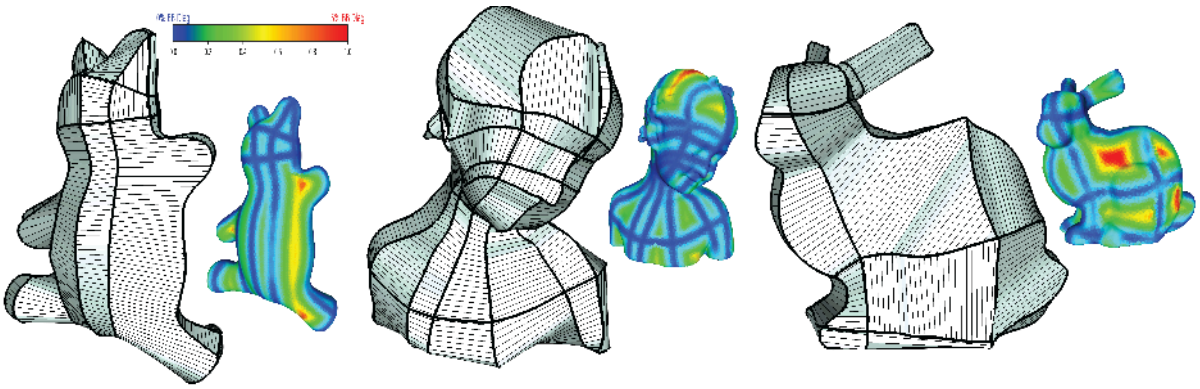


Fig. 25. The layout of ruling patches for a selection of general meshes.

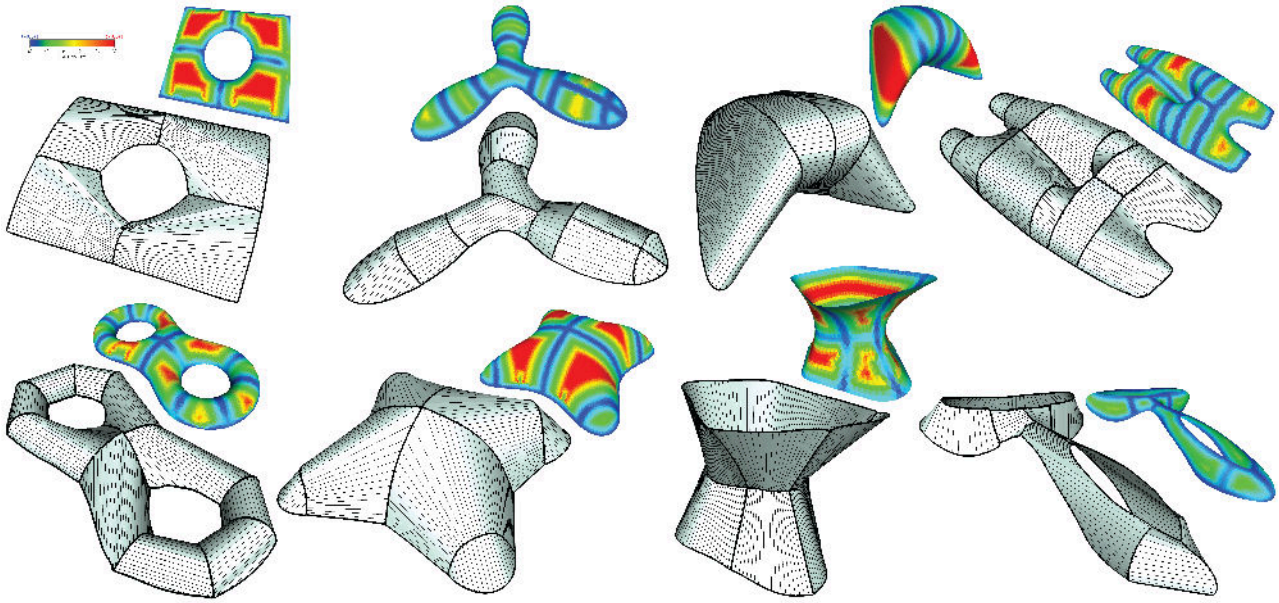


Fig. 26. The layout of ruling patches for a selection of architectural models with symmetric decomposition.

direction *enters* the mesh, and the destination nodes encompass all border nodes whose direction *exits* the mesh.

Path smoothing. Given that paths consist of sequences of edges of the triangle mesh, they often exhibit a zig-zagging shape, contrary to our objective of layout simplicity. To counter this, we employ extrinsic smoothing to all the vertices on the path, followed by their re-projection onto the original surface (see Figure 15).

Path insertion strategy. We begin by sampling a set of candidate paths that trace loops or extend from border to border. For efficiency, we uniformly subsample the source nodes both on the border and within the mesh interior. After [Livesu et al. 2020], we insert paths using a greedy approach that prioritizes paths furthest from those previously inserted. Distances for each node are computed using the $M4$ stratification of the graph and are subsequently averaged for each path.

For each insertion step, we update the entire patch layout to evaluate its viability. Then, we mark each patch failing to meet the topology (as outlined in Section 3) or geometry criteria. A candidate path is considered for insertion only if it splits a patch that does not match the specified goals and does not intersect tangentially with any previously inserted path. Notice that, in line with the approach presented in Pietroni et al. [2016] and Livesu et al. [2020], determining the tangential intersection between two candidate paths is straightforward: it involves a simple check to verify whether two paths pass through the same vertex in non-orthogonal directions.

An example of the step taken by this sampling procedure is depicted in Figure 16.

Path removal strategy. As mentioned, paths are tested for removal in inverse order of creation, until only paths deemed necessary for the viability of the patch layout remain. Figure 17 shows an example. As a variant, we can test the removal of individual arches rather than entire paths. This results in a patch layout with T-junctions (see Figure 7).

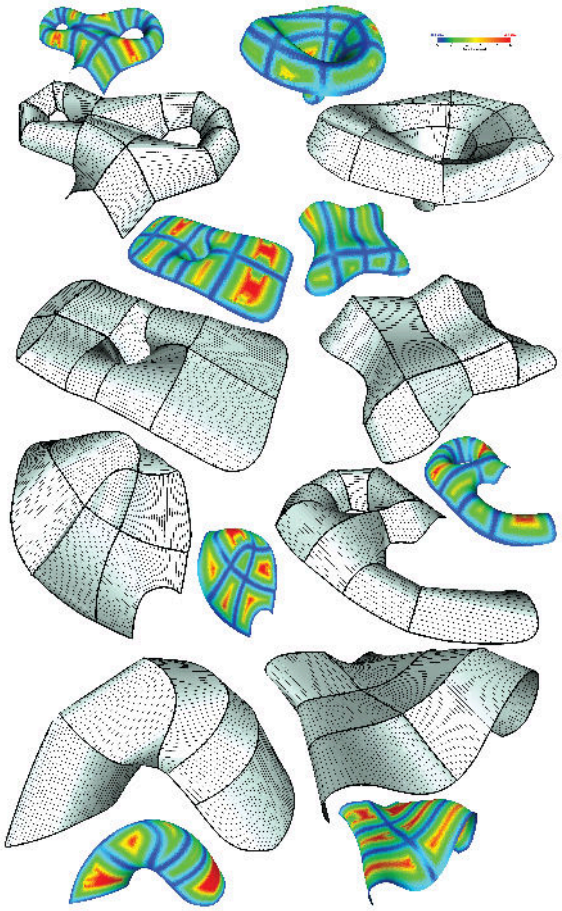


Fig. 27. The layout of ruling patches for a selection of architectural models.

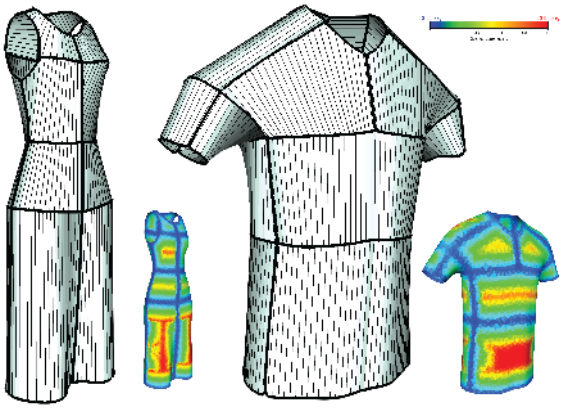


Fig. 28. The layout generated for Fashion Design.

3.5 Final Remeshing

Finally, the last produced morphed mesh is remeshed to obtain the final discretized ruled surfaces (which are developable by construction). The final meshing is obtained by sampling over each arch at the boundary between (up to) two patches, all the intersections with every ruled line on either side. Intersection points on opposite

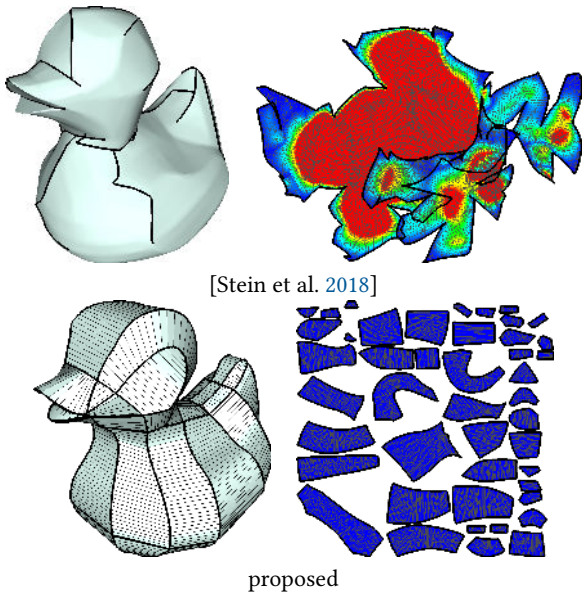


Fig. 29. Comparison of the proposed method (bottom) with that of Stein and colleagues [Stein et al. 2018] (top). Unlike the concurrent method, our approach ensures both bijectivity and full developability (red areas indicating where stretching exceeds 5%.

sides of a patch are connected, creating mesh edges that correspond to the selected ruling lines. Additional edges are inserted to triangulate the entire mesh, prioritizing edges with a direction similar to the ruling lines. See Figures 18, 11(d),(e) for examples of this remeshing, and of the corresponding isometric flattening.

Final patch-shape optimization. Over highly non-developable parts of the mesh, the orientation of the ruling lines determined by our parametrization-based approach (Figure 19(a)) will inevitably undergo high distortion during the morphing (Figure 19(b)). While this does not compromise the ability of the morphing to reliably assess the viability of a patch layout, the shape of patch boundaries can be adversely affected. As a simple way to counter this, we apply an extra optimization, only in the last iteration, consisting of repeating the entire morphing procedure a second time, including the determination of the ruling lines, this time targeting the already morphed mesh (as shown in Figure 19(c)).

4 Results

We run our test implementation on an Apple M1 Max notebook with 32 GB of RAM. The examples shown in the paper took from 6 seconds (the models in Figure 22) up to 80 seconds (for the complex models shown in Figure 23). These times allow users to adjust the parameters $e_{\max}$, searching for a preferred tradeoff (Figure 6). In all the examples, we used a maximum error ranging from 1 to 5 % of the diagonal of the input bounding box.

We successfully tested our method on a wide range of different input geometries. The results shown in Figures 22, 24, 25, and elsewhere in the paper indicate that our method performs well on fairly regular input shapes, yielding consistently good results. In the attached materials, we make all input meshes and the final design

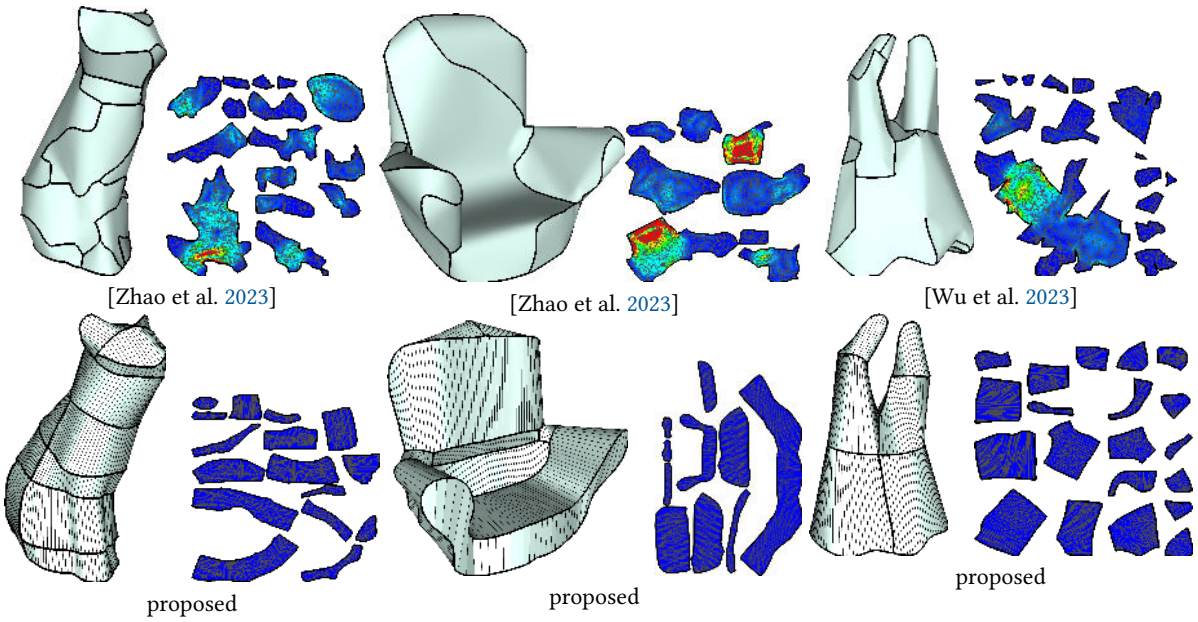


Fig. 30. Comparison of our method with the ones recently proposed by Zhao et al. [2023] and Wu et al. [2023]. The two layouts have similar boundary lengths. The last column shows the distortion in the 2D flattening of two patches from the dataset of Zhao et al. We used the As-Rigid-As-Possible deformation technique [Sorkine and Alexa 2007], where the red colour corresponds to 1% edge elongation.

(with the 2D layouts available both as UV maps of the meshes and as separated SVG files).

We test our method on shapes inspired form various specific contexts where developable ruled surfaces can find application, such as architectural design, with both symmetrical (Figure 26), and non-symmetrical (Figure 27) examples, and fashion design (Figure 28). In all cases, our algorithm generates high-quality consistent rulings that smoothly flow in the principal curvature direction. Although our method is not specialized for deriving garment pattern layouts [Pietroni et al. 2022], it produces convincing paper patterns.

Finally, Figure 23 showcases a particularly intricate models designed to highlight the robustness of our proposed method.

4.1 Input Preparation

Our method expects, as input, a manifold mesh, with reasonably uniform tesselation; whenever these conditions were not met, we uniformly re-meshed the input shape using standard tools (we use the Open Source suite [Cignoni et al. 2011]).

Our method also expects a cross-field defined on the mesh. In each experiment, we produced one by first extracting the two local principal curvature directions with Panozzo et al. [2010], then globally smoothing it with Diamanti et al. [2014], although any similarly purposed method could be employed. The smoothing is designed to align to the two principal curvature directions, where they are strongly defined, and to the mesh boundaries, if they are present, while maintaining smoothness elsewhere.

4.2 Comparison with Previous Work

In an initial set of experiments, presented in Figures 29 and 30, we compare our method to those proposed by Stein et al. [2018], Zhao et al. [2023], and Wu et al. [2023], which, similar to ours, seek to produce a decomposition into almost-developable patches. As illustrated, the decomposition produced by Stein et al. [2018] is not directly fabricable due to deformations exceeding 5% in several regions–well beyond the material limitations of stretchable media like paper [Henriksson et al. 2007]. Furthermore, the parameterization lacks bijectivity, which further impedes fabricability (see Figure 29). Minor issues are evident in other methods, as seen in Figure 30, where deformations exceed 1%. In contrast, our method inherently enforces no-stretching constraints and bijectivity, ensuring fabricability. Additionally, our decompositions are more compact and exhibit a regular structure.

Naturally, one could apply our final discretization step to a layout produced with any other method, thus obtaining a fully developable patches like ours. However, our experiments confirm the expectation that patch layouts not targeted for this result in a worse geometric approximation. In Figure 31, we compare against the layouts produced by Cohen-Steiner et al. [2004] and a simple Lloyd relaxation of restricted Voronoi diagrams. Our results are superior in terms of geometry accuracy.

4.3 Physical Realizations

We fabricated several physical examples to demonstrate the applicability of the generated design. First, we fabricated a paper covering of a vase-lion, Figure 20, top. The statue highlights the advantage of our discrete patches in allowing both bending and twisting to approximate the target mesh. The advantage of twisting can be appreciated in the region under the lion's mouth, where a single patch is split into two ends.

Our second example (Figure 20, bottom) shows a potential application of our method to construction. To physically realize the piece, we 3D printed a rod-like support following the boundary of

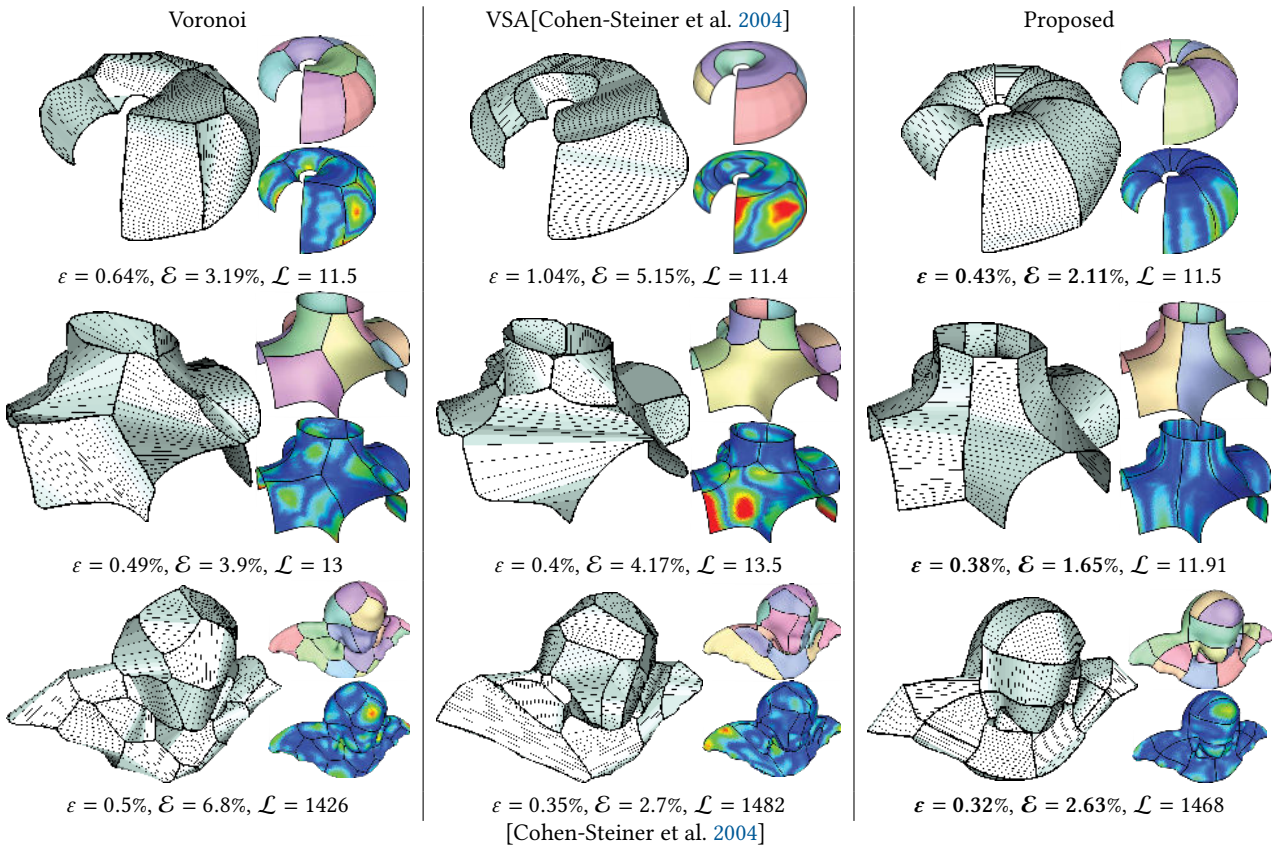


Fig. 31. A comparison between our field-aligned patch layout with the Voronoi partition with Loyd relaxation and Variational Shape Approximation [Cohen-Steiner et al. 2004]. The proposed method results in a better approximation of the target geometry, when each patch is discretized in a fully developable ruled surface, for a similar total length of the cuts.

our patch layout. The ruled patches are then cut from cardboard and glued onto this supporting, resulting in a robust model with a very limited amount of supporting material.

Our last example, in Figure 21, demonstrates an application of our method for the stylized fabrication of a lamp. In this case, the produced design naturally preserves the three-way symmetry of the input, without explicitly enforcing it.

5 Conclusion

We introduced a novel method for discretizing ruled surfaces that retains the primary advantages of ruled surfaces while overcoming the classical limitations of previous discretization methods. Building on this concept, we have developed an approach that automatically decomposes a freeform surface into a set of developable patches.

Our method imposes no stringent requirements on the input shape and ensures straight ruling lines in the final decomposition. We have demonstrated the fidelity and versatility of our method in various contexts and input variations and have successfully fabricated several physical objects.

5.1 Limitations and Future Work

One limitation stems from the use of geometric distance as the only measure of accuracy. This is robust but can potentially miss semantically important features characterized by variations in the normals (as visible in the right inset), or introduce loss of normal continuity across boundaries (as visible in many result images) or occasionally even inside patches. Although this can be tolerated in most scenarios, it would be interesting to study ways to enforce or encourage final normals to be smooth across patches or in general to adhere to the input mesh, for example, to improve aesthetics.

Other future directions may include exploring optimization techniques. Although our method is efficient for moderately complex shapes, it is not currently fast enough to enable user interaction and on-the-fly manual tuning by artists.

References

Alexandre Binninger, Floor Verhoeven, Philipp Herholz, and Olga Sorkine-Hornung. 2021. Developable approximation via Gauss image thinning. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 289–300.

Pengbo Bo and Wenping Wang. 2007. Geodesic-controlled developable surfaces for modeling paper bending. In *Computer Graphics Forum*, Vol. 26. Wiley Online Library, 365–374.

Marcel Campen. 2017. Partitioning surfaces into quadrilateral patches: A survey. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 567–588.

Paolo Cignoni, Guido Ranzuglia, M. Callieri, M. Corsini, F. Ganovelli, N. Pietroni, M. Tarini, et al. 2011. MeshLab. (2011).

David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational shape approximation. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 905–914. DOI: https://doi.org/10.1145/1015706.1015817

Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. 2014. Designing *N*-PolyVector fields with complex polynomials. *Comput. Graph. Forum* 33, 5 (2014), 1–11.

Levi H. Dudte, Etienne Vouga, Tomohiro Tachi, and Lakshminarayanan Mahadevan. 2016. Programming curvature using origami tessellations. *Nature Materials* 15, 5 (2016), 583–588.

Gerald E. Farin. 1997. *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide.*

Marco Freire, Manas Bhargava, Camille Schreck, Pierre-Alexandre Hugron, Bernd Bickel, and Sylvain Lefebvre. 2023. PCBend: Light up your 3D shapes with foldable circuit boards. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–16.

Konstantinos Gavriil, Ruslan Guseinov, Jesús Pérez, Davide Pellis, Paul Henderson, Florian Rist, Helmut Pottmann, and Bernd Bickel. 2020. Computational design of cold bent glass façades. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–16.

Lars Henriksson, Ronald Bredemo, Christer Fellers, and Greger Stubbfalt. 2007. *Mechanical Properties of Paper and Their Relation to Packaging Performance.* Appita Inc., Carlton, Vic. Retrieved from https://search.informit.org/doi/10.3316/informit.804235712193932

Alexandra Ion, Michael Rabinovich, Philipp Herholz, and Olga Sorkine-Hornung. 2020. Shape approximation by developable wrapping. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–12.

Caigui Jiang, Florian Rist, Helmut Pottmann, and Johannes Wallner. 2020a. Freeform quad-based kirigami. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–11.

Caigui Jiang, Cheng Wang, Florian Rist, Johannes Wallner, and Helmut Pottmann. 2020b. Quad-mesh based isometric mappings and developable surfaces. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 128–1.

Martin Kilian, Simon Flöry, Zhonggui Chen, Niloy J. Mitra, Alla Sheffer, and Helmut Pottmann. 2008. Curved folding. *ACM Transactions on Graphics (TOG)* 27, 3 (2008), 1–9.

Mina Konaković, Keenan Crane, Bailin Deng, Sofien Bouaziz, Daniel Piker, and Mark Pauly. 2016. Beyond developable: Computational design and fabrication with auxetic materials. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–11.

Kai Wah Lee and Pengbo Bo. 2016. Feature curve extraction from point clouds via developable strip intersection. *Journal of Computational Design and Engineering* 3, 2 (2016), 102–111.

Minchen Li, Danny M. Kaufman, Vladimir G. Kim, Justin Solomon, and Alla Sheffer. 2018. OptCuts: Joint optimization of surface cuts and parameterization. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–13.

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. 2008. A local/global approach to mesh parameterization *(SGP'08)*. Eurographics Association, Goslar, DEU, 1495–1504.

Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. 2006. Geometric modeling with conical meshes and developable surfaces. In *ACM SIGGRAPH 2006 Papers.* 681–689.

Marco Livesu, Nico Pietroni, Enrico Puppo, Alla Sheffer, and Paolo Cignoni. 2020. Loopycuts: Practical feature-preserving block decomposition for strongly hex-dominant meshing. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 121–1.

Fady Massarwi, Craig Gotsman, and Gershon Elber. 2007. Papercraft models using generalized cylinders. In *15th Pacific Conference on Computer Graphics and Applications (PG'07).* IEEE, 148–157.

Jun Mitani and Hiromasa Suzuki. 2004. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Transactions on Graphics (TOG)* 23, 3 (2004), 259–263.

Yuta Noma, Koya Narumi, Fuminori Okuya, and Yoshihiro Kawahara. 2020. Pop-up print: Rapidly 3D printing mechanically reversible objects in the folded state. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology.* 58–70.

Stefano Nuvoli, Alex Hernandez, Claudio Esperança, Riccardo Scateni, Paolo Cignoni, and Nico Pietroni. 2019. QuadMixer: Layout preserving blending of quadrilateral meshes. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–13.

Barrett O'neill. 2006. *Elementary Differential Geometry.* Elsevier.

D. Panozzo, E. Puppo, and L. Rocca. 2010. Efficient multi-scale curvature and crease estimation. In *2nd International Workshop on Computer Graphics, Computer Vision and Mathematics, GraVisMa 2010—Workshop Proceedings (2nd International Workshop on Computer Graphics, Computer Vision and Mathematics, GraVisMa 2010—Workshop Proceedings).* 9–16. 2nd International Workshop on Computer Graphics, Computer Vision and Mathematics, GraVisMa 2010; Conference date: 07-09-2010 Through 10-09-2010.

Nico Pietroni, Corentin Dumery, Raphael Falque, Mark Liu, Teresa Vidal-Calleja, and Olga Sorkine-Hornung. 2022. Computational pattern making from 3D garment models. 41, 4, Article 157 (Jul 2022), 14 pages. DOI: https://doi.org/10.1145/3528223.3530145

Nico Pietroni, Stefano Nuvoli, Thomas Alderighi, Paolo Cignoni, Marco Tarini, et al. 2021. Reliable feature-line driven quad-remeshing. *ACM Transactions on Graphics* 40, 4 (2021), 1–17.

Nico Pietroni, Enrico Puppo, Giorgio Marcias, Roberto Scopigno, and Paolo Cignoni. 2016. Tracing field-coherent quad layouts. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 485–496.

Ulrich Pinkall and Konrad Polthier. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics* 2, 1 (1993), 15–36.

Roi Poranne, Marco Tarini, Sandro Huber, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Autocuts: Simultaneous distortion and cut optimization for UV mapping. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–11.

Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2018a. Discrete geodesic nets for modeling developable surfaces. *ACM Transactions on Graphics (ToG)* 37, 2 (2018), 1–17.

Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2018b. The shape space of discrete orthogonal geodesic nets. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–17.

Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2019. Modeling curved folding with freeform deformations. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–12.

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable locally injective mappings. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1.

Faniry H. Razafindrazaka, Ulrich Reitebuch, and Konrad Polthier. 2015. Perfect matching quad layouts for manifold meshes. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 219–228.

Kenneth Rose, Alla Sheffer, Jamie Wither, Marie-Paule Cani, and Boris Thibert. 2007. Developable surfaces from arbitrary sketched boundaries. In *SGP'07-5th Eurographics Symposium on Geometry Processing*. Eurographics Association, 163–172.

Camille Schreck, Damien Rohmer, Stefanie Hahmann, Marie-Paule Cani, Shuo Jin, Charlie C. L. Wang, and Jean-Francis Bloch. 2016. Nonsmooth developable geometry for interactively animating paper crumpling. *ACM Trans. Graph.* 35, 1, Article 10 (Dec. 2016), 18 pages. DOI: https://doi.org/10.1145/2829948

Christian Schüller, Roi Poranne, and Olga Sorkine-Hornung. 2018. Shape representation by zippables. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.

Silvia Sellán, Noam Aigerman, and Alec Jacobson. 2020. Developability of heightfields via rank minimization. *ACM Trans. Graph.* 39, 4 (2020), 109.

Nicholas Sharp and Keenan Crane. 2018. Variational surface cutting. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.

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

Justin Solomon, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2012. Flexible developable surfaces. *Comput. Graph. Forum* 31, 5 (Aug 2012), 1567–1576. DOI: https://doi.org/10.1111/j.1467-8659.2012.03162.x

Olga Sorkine and Marc Alexa. 2007. As-Rigid-As-Possible surface modeling. In *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing.* 109–116.

Oded Stein, Eitan Grinspun, and Keenan Crane. 2018. Developability of triangle meshes. *ACM Trans. Graph.* 37, 4, Article 77 (Jul 2018), 14 pages. DOI: https://doi.org/10.1145/3197517.3201303

Yasaman Tahouni, Tiffany Cheng, Dylan Wood, Renate Sachse, Rebecca Thierer, Manfred Bischoff, and Achim Menges. 2020. Self-shaping curved folding: A 4D-printing method for fabrication of self-folding curved crease structures. In *Proceedings of the 5th Annual ACM Symposium on Computational Fabrication.* 1–11.

Masahito Takezawa, Takuma Imai, Kentaro Shida, and Takashi Maekawa. 2016. Fabrication of freeform objects by principal strips. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–12.

Chengcheng Tang, Pengbo Bo, Johannes Wallner, and Helmut Pottmann. 2016. Interactive design of developable surfaces. *ACM Transactions on Graphics (TOG)* 35, 2 (2016), 1–12.

Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. 2016. Directional field synthesis, design, and processing. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 545–572.

Floor Verhoeven, Amir Vaxman, Tim Hoffmann, and Olga Sorkine-Hornung. 2022. Dev2PQ: Planar quadrilateral strip remeshing of developable surfaces. *ACM Transactions on Graphics (TOG)* 41, 3 (2022), 1–18.

Charlie Wang. 2008. Computing length-preserved free boundary for quasi-developable mesh segmentation. *IEEE Transactions on Visualization and Computer Graphics* 14, 1 (2008), 25–36. DOI: https://doi.org/10.1109/TVCG.2007.1067

Charlie C. L. Wang and Gershon Elber. 2014. Multi-dimensional dynamic programming in ruled surface fitting. *Computer-Aided Design* 51 (2014), 39–49. DOI: https://doi.org/10.1016/j.cad.2014.02.004

Charlie C. L. Wang and Kai Tang. 2004. Achieving developability of a polygonal surface by minimum deformation: A study of global and local optimization approaches. *The Visual Computer* 20 (2004), 521–539.

Charlie C. L. Wang and Kai Tang. 2005. Optimal boundary triangulations of an interpolating ruled surface. *J. Comput. Inf. Sci. Eng.* 5 (2005), 291–301. DOI: https://api.semanticscholar.org/CorpusID:16088189

Hui Wang, Davide Pellis, Florian Rist, Helmut Pottmann, and Christian Müller. 2019. Discrete geodesic parallel coordinates. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–13.

Kang Wu, Zheng-Yu Zhao, Zheng Zhang, Ligang Liu, and Xiao-Ming Fu. 2023. Piecewise developable modeling via implicit neural deformation and feature-guided cutting. *IEEE Transactions on Visualization and Computer Graphics* 30, 9 (Sept. 2023), 5993–6004. DOI: https://doi.org/10.1109/TVCG.2023.3319487

Shin Yoshizawa, Alexander Belyaev, and Hans-Peter Seidel. 2004. A fast and simple stretch-minimizing mesh parameterization. In *Proceedings Shape Modeling Applications, 2004.* IEEE, 200–208.

Chao Yuan, Nan Cao, and Yang Shi. 2023. A survey of developable surfaces: From shape modeling to manufacturing. arXiv:2304.09587. Retrieved from https://arxiv.org/abs/2304.09587

Xiaoting Zhang, Guoxin Fang, Mélina Skouras, Gwenda Gieseler, Charlie C. L. Wang, and Emily Whiting. 2019. Computational design of fabric formwork. *ACM Transactions on Graphics* 38, 4 (2019), 1–13.

Zheng-Yu Zhao, Qing Fang, Wenqing Ouyang, Zheng Zhang, Ligang Liu, and Xiao-Ming Fu. 2022. Developability-driven piecewise approximations for triangular meshes. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–13.

Zheng-Yu Zhao, Mo Li, Zheng Zhang, Qing Fang, Ligang Liu, and Xiao-Ming Fu. 2023. Evolutionary piecewise developable approximations. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–14.

Received 15 November 2024; revised 11 April 2025; accepted 16 April 2025