Robby Kraft, Rupert Maleczek, Klara Mundilova, Tomohiro Tachi

From Quad Filling to Wrinkled Surfaces

Abstract: Fabricating complex geometries from flat sheets has many practical advantages such as cost-efficient fabrication and space-efficient transportation. In this paper, we explore a family of shapes that consist of two types of curved-crease molecules, which can be composed as a modular design. Alternatively, we explore how to optimize the target shape towards a global origami development, that is, a development that does not require additional slits or holes.

1 Introduction

Developable surfaces, or developables, are surfaces that can be flattened into the plane without stretching or tearing. These single-curved surfaces are often times easier and thus more cost-efficient to fabricate than doubly-curved surfaces. Therefore they have various applications in many disciplines, such as architecture, engineering and design [6, 8].

Inspired by straight crease origami, bend surfaces can be folded along curves. This enables a new space of achievable shapes that can be approximated in the straight crease origami world by only infinitely many tiny creases. Shapes obtained by folding along curves share many benefits of straight crease origami: creases can add structural stiffness, folded shapes decrease material off-cut in production, and the designs are easier to transport in their original flat state [4, 13].

Although physical fabrication of curved creases with paper may seem straightforward, the digital design of curved folded shapes is not trivial. In particular, approximating a target shape with a specific curved-crease design is challenging. Only recently, two approaches were studied. Jiang et al. [7] study principal pleatedstructures and propose a method for the reconstruction of shapes as curve pleated structures. Maleczek et al. [10] construct an edge rounded version of a polyhedral surface by first rounding each edge with a right circular cylinder and then folding the cylinders incident at a vertex into (generalized) cones.

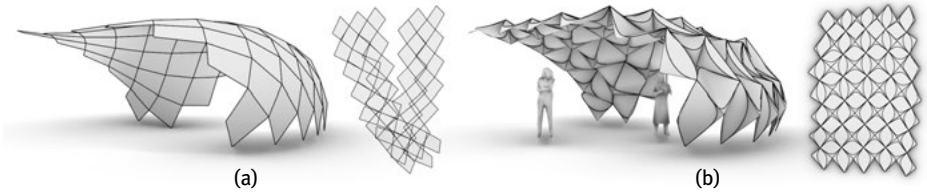

Fig. 1: A pavilion structure using the lens molecules (b), and its non-devlopable target geometry (a).

https://doi.org/10.1515/9783111162683-025

In this paper, we propose a new approach to approximate a user-specified polyhedral surface with developable quadrangular curved crease modules; modules which are able to be customized in design across a few different parameters. We demonstrate how the developable state is generated, how to target alternative developed shapes, and using this, generate a globally developable surface without cuts. Our method is implemented as a grasshopper plug-in for Rhino6/7 [1, 2].

Our paper is organized as follows. In Sec. 2, we show how to fill a quadrangular face with two curved crease designs: the cone-cone, and the lens. We then propose a subdivision scheme to prepare an arbitrary mesh for our surface-filling algorithm in Sec. 3. Finally, we explain how to target a developable shape from a range of configurations and build a global development without holes in Sec. 4.

2 Quad Filling

In this section, we show how to fill a single non-planar quad $Q = \{\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3, \mathbf{V}_4\}$ with a curved crease design, consisting of cylinders and cones[1]. In particular, we consider two developable surface layouts (see Fig. 2):

- *Cone-cone:* A cone with apex $\mathbf{V}_1$ and rulings $\mathbf{V}_1\mathbf{V}_2$ and $\mathbf{V}_1\mathbf{V}_4$ is folded into a cone with apex $\mathbf{V}_3$ such that the crease passes through the points $\mathbf{V}_2$ and $\mathbf{V}_4$.
- *Lens:* A central cylinder is folded into two cones with apices $\mathbf{V}_1$ and $\mathbf{V}_3$ such that both creases between the cylinder and the cone pass through points $\mathbf{V}_2$ and $\mathbf{V}_4$.

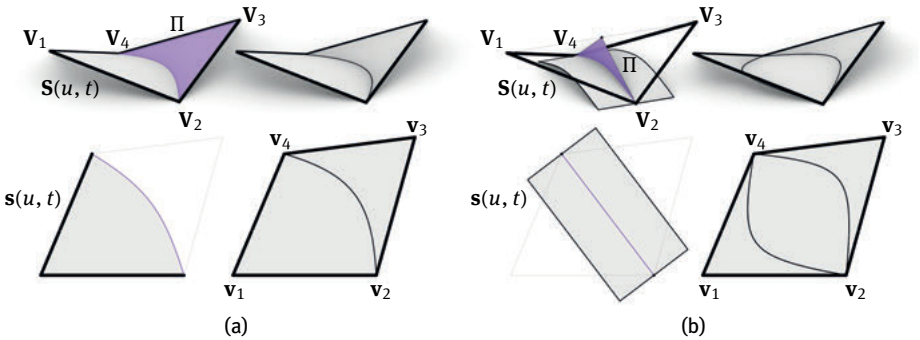

Fig. 2: Two steps of the quad filling method for the cone-cone molecule (a) and the lens molecule (b).

1 In this paper, when we refer to cones and cylinders, we are referring to the generalized versions unless specified otherwise. Cones, or conical surfaces, are surfaces composed of lines that all intersect at a single point known as the apex of the cone. Cylinders, or cylindrical surfaces, on the other hand, are surfaces composed of lines that are all parallel to each other.

In both cases, our method consists of two steps:

1. *Construct first surface:* First, we construct a cylindrical or conical surface $\mathbf{S}(u, t)$ inside the quad. A curve which defines the shape of this surface and further parameters can be specified by the user. Details are provided in Sec. 2.1.
2. *Construct remaining surface(s):* Then, we construct the fold between $\mathbf{S}(u, t)$ and a cone with apex $\mathbf{V}_4$ (cone-cone), or two cones with apices $\mathbf{V}_1$ and $\mathbf{V}_4$ (lens). Details are provided in Sec. 2.2.

2.1 Construction of the first surface

Intuitively, we obtain the first surface $\mathbf{S}(u, t)$ by extruding a not self-intersecting planar curve $\mathbf{P}(u)$ with start point $\mathbf{V}_2$ and end point $\mathbf{V}_4$. In the following, we will call $\mathbf{P}(u)$ the *base curve*. In particular, when constructing a cone, we extrude $\mathbf{P}(u)$ w.r.t. the center $\mathbf{V}_1$ by connecting every point of $\mathbf{P}(u)$ with $\mathbf{V}_1$ using a line. When constructing a cylinder, $\mathbf{P}(u)$ is extruded in the direction that is perpendicular to the curve's incident plane. The resulting surface can therefore be parametrized as $\mathbf{S}(u, t) = \mathbf{C}(u) + t\mathbf{R}(u)$, where in case of a cone we set $\mathbf{C}(u) = \mathbf{V}_1$ and $\mathbf{R}(u) = \frac{\mathbf{P}(u)-\mathbf{V}_1}{|\mathbf{P}(u)-\mathbf{V}_1|}$, and in case of a cylinder we set $\mathbf{C}(u) = \mathbf{P}(u)$ and $\mathbf{R}$ to be the normalized vector perpendicular to the curve's base plane.

The choice of $\mathbf{P}(u)$ influences not only the shape of the first surface $\mathbf{S}(u, t)$, but also the shape of the second, constructed cone $\mathbf{S}_2(u, t)$ that connects to $\mathbf{S}(u, t)$ with a curved crease. When choosing the planar curve $\mathbf{P}(u)$ arbitrarily, we might observe some undesired artifacts, such as local self-intersections of $\mathbf{S}_2(u, t)$ and "complete" or "no" folds between $\mathbf{S}(u, t)$ and $\mathbf{S}_2(u, t)$, see Fig. 3 (left).

Fig. 3: Left: Intersecting surfaces. Right: Illustration of a projective mapping used for the construction of a $\mathbf{Z}'$-central base curve of the first surface.

In the following, we discuss (1) a constraint on the rulings of the constructed surface $\mathbf{S}(u, t)$ such the curved-crease connection with $\mathbf{S}_2(u, t)$ does not result in self-intersections, and (2) how to modify a user-specified input curve $\mathbf{Q}(u)$ using projective transformations to satisfy this constraint.

2.1.1 Non-self-intersecting cones and central functions

Suppose we want to determine the fold between a developable surface $\mathbf{S}(u, t)$ and a cone with apex $\mathbf{V}$. The developable surface $\mathbf{S}(u, t)$ contains a family of lines, the so-called *rule lines*. Let us now consider the family of *ruling planes* T, that is, planes that consist of two consecutive rulings joined by a point on the curved crease, in our case a ruling of $\mathbf{S}(u, t)$ and the cone apex $\mathbf{V}$, see Fig. 4. If $\mathbf{S}(u, t)$ is a conical surface with apex $\mathbf{V}_1$, the family of planes is a bundle of planes with axis $\mathbf{V}_1\mathbf{V}$. If $\mathbf{S}(u, t)$ is a cylindrical surface with ruling direction $\mathbf{R}$, the family of planes is a bundle of planes with its axis passing through $\mathbf{V}$ and having direction $\mathbf{R}$.

As the rulings continuously vary along $\mathbf{S}(u, t)$, so do the planes in T. The second surface does not have any self-intersections if all planes in T are distinct, that is, don't "double back", see Fig, 3 (left). We argue that surfaces are not repeating by considering a planar section of the bundle of planes with a plane Π that passes through $\mathbf{V}$ and intersects all planes at least once. If all intersecting lines of $\Pi \cap T$ are unique, so are the planes. The case-specific location of plane Π is discussed in Sec. 2.1.2, construction step (a).

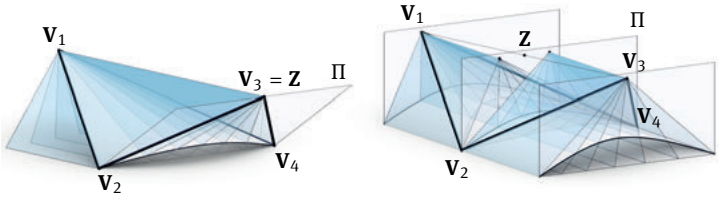

Fig. 4: Illustration of the bundle of ruling planes used in the argumentation in Sec. 2.1.1. Left: cone-cone molecule. Right: Lens molecule.

Consequently, when locating the base curve $\mathbf{P}(u)$, we want it to induce non-repeating planes, resulting in a curve that has unique connecting lines with the apex. In the following, we call such a curve a *central function*[2]. We can obtain a central function with center $\mathbf{Z}$ from the graph of a function by applying a projective transformation that maps the ideal point of the y-axis to the center $\mathbf{Z}$, see Fig. 3 (right).

In case of the cone-cone design, we construct the first surface $\mathbf{S}(u, t)$ such that its base curve is a $\mathbf{V}_3$-central function. In case of a lens design, we determine the fold between the cylinder and two cones on either side. Thus we would need a central function w.r.t. two centers (the orthogonal projections of $\mathbf{V}_1$ and $\mathbf{V}_3$ on Π). If the orthogonal projections are not the same, we observed in our experiments that it is usually sufficient to approximate this "doubly-central" function by a $\mathbf{Z}$-central function, where $\mathbf{Z}$ is the intersection of the line spanned by $\mathbf{V}_1$ and $\mathbf{V}_3$ with Π.

2 This terminology is based on the "common" 2D functions, which are central functions w.r.t. the point at infinity of the z-axis.

2.1.2 Surface fitting

On a highlevel, the construction of the first conical or cylindrical surface amounts in locating plane Π and constructing an appropriate central base curve $\mathbf{P}(u) \in \Pi$ which reassembles the shape of a given input curve.
In addition to the coordinates of the 3D quad, the inputs for the surface filling are a *shape-defining curve*, that is a curve $\mathbf{Q}(u) = (u, f(u), 0)$ where $f(u)$ is a function with $f(0) = f(1) = 0$, and a *scale parameter* s. By changing the shape of this curve, the user can influence the initial surface, which affects the location and shape of the computed crease curves, offering more design freedom. Moreover, if we are constructing the lens design, we have an additional parameter ϕ that influences the orientation of Π.

Although the following operations can be performed analytically, we explain the following in terms of our implementation using a sampling of the input curve. We adopt this discretization approach because the computation of the crease curve relies on establishing an accurate correspondence between the 3D surface and its developed representation. Furthermore, the subsequent section requires information about the opening angle of the developed conical surface or the length of the base curve of the cylinder. Both of these quantities can be readily approximated from a finely sampled mesh.

Our proposed algorithm works as follows, see Fig. 5:

(a) *Locate plane* Π*:* For a given 3D quad, we define the design-dependent base plane Π. In case of a cone, let Π be the plane containing the points $\mathbf{V}_2$, $\mathbf{V}_3$ and $\mathbf{V}_4$. In case of a cylinder, let Π be a plane containing $\mathbf{V}_2$ and $\mathbf{V}_4$, whose orientation can be influenced by a parameter ϕ (the rotation about $\mathbf{V}_2\mathbf{V}_4$). In both cases, let $\mathbf{V}$ be the intersection of Π with the line spanned by $\mathbf{V}_1$ and $\mathbf{V}_3$ (in the cone design, $\mathbf{Z} = \mathbf{V}_3$).

(b) *Scale and orient* $\mathbf{Q}(u)$*:* We scale the user defined curve so that the distance between its endpoints equals $|\mathbf{V}_2 - \mathbf{V}_4|$. In addition, we scale the curve in y-direction by the user-specified parameter s. Finally, we move the curve from the xy-plane to Π, such that $\mathbf{Q}(0) = \mathbf{V}_2$ and $\mathbf{Q}(1) = \mathbf{V}_4$, such that $\mathbf{Z}$ lies in the $y > 0$ half-plane. Note that the resulting polyline might not be $\mathbf{Z}$-central and thus not suitable for the fold construction without self-intersections.

(c) *Projective transformation:* We apply a projective mapping to transform $\mathbf{Q}(u)$ into a $\mathbf{Z}$-central function. In the following, we utilize a local 2D coordinate system where $\mathbf{V}_2$ corresponds to the origin, $\mathbf{V}_4 - \mathbf{V}_2$ to the x-axis, and $\mathbf{Z}$ lies on the half-plane with

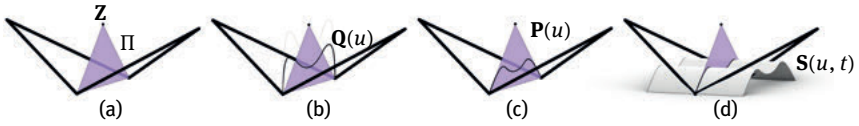

Fig. 5: Surface fitting. Illustration of the construction steps 1(a) – 1(d).

$y > 0$. Let $\mathbf{Z}'$ denote the coordinates of $\mathbf{Z}$ in this local coordinate system. Moreover, let $\mathcal{Q}' = (\mathbf{Q}'_1, \mathbf{Q}'_2, \ldots, \mathbf{Q}'_n)$ be a sampling of curve $\mathbf{Q}(u)$ in this local coordinate system. We use the following projective transformation to turn $\mathcal{Q}'$ in a $\mathbf{Z}'$-central polyline $\mathcal{P}' = (\mathbf{P}'_1, \mathbf{P}'_2, \ldots, \mathbf{P}'_n)$, that is,

$$(\mathbf{Q}'_x, \mathbf{Q}'_y) \mapsto \mathbf{P}' = \left(\frac{k\mathbf{Q}'_x + \mathbf{Z}'_x\mathbf{Q}'_x}{k + \mathbf{Q}'_y}, \frac{\mathbf{Z}'_y\mathbf{Q}'_y}{k + \mathbf{Q}'_y} \right) \quad \text{where} \quad k = \mathbf{Z}'_y - \min_i \mathbf{Q}'_{i,y}.$$

Note that because of the choice of k, the points are bounded. In particular, the lowest y-coordinate of $\mathbf{P}'_i$ is the lowest y-coordinate of $\mathbf{Q}'_i$.

(d) Finally, depending on the application, we can either smoothly or linearly interpolate the points of $\mathcal{P}$ to obtain $\mathbf{P}(u)$ (in world-coordinates) and construct the design-depended initial surface.

2.2 Construction of the remaining surface(s)

In the following, we briefly review the patch-to-cone construction presented by Mundilova [11], to construct a crease between a given developable surface patch and a cone. For more details on the implementation, see Mundilova et al. [12].

2.2.1 Preparing the development

To construct the crease that connects the given developable surface $\mathbf{S}(u, t)$ with a cone, we first need to find the unrolled surface $\mathbf{s}(u, t)$. In our implementation, we utilize the above approximation of $\mathbf{S}(u, t)$ by a discretized cone (family of triangles) or discretized cylinder (family of planar quads with parallel edges). We place the faces successively into the xy-plane and denote the 2D counterparts of $\mathbf{P}_i$ as $\mathbf{p}_i$, the 2D counterpart of the ruling direction $\mathbf{R}_i$ incident to $\mathbf{P}_i$ as $\mathbf{r}_i$ (in case of the cylinder, all $\mathbf{r}_i$ are the same). Furthermore, let $\mathbf{v}_2 = \mathbf{p}_1$ and $\mathbf{v}_4 = \mathbf{p}_n$. Note that in case of a lens design, because $\mathbf{P}(u)$ is a curve that lies in a plane orthogonal to the ruling direction, its development lies on a straight line. Finally, we find the points $\mathbf{v}_1$ and $\mathbf{v}_3$ corresponding to $\mathbf{V}_1$ and $\mathbf{V}_3$ on the appropriate side such that

$$|\mathbf{v}_1 - \mathbf{v}_2| = |\mathbf{V}_1 - \mathbf{V}_2| \quad \text{and} \quad |\mathbf{v}_1 - \mathbf{v}_4| = |\mathbf{V}_1 - \mathbf{V}_4|,$$
$$|\mathbf{v}_3 - \mathbf{v}_2| = |\mathbf{V}_3 - \mathbf{V}_2| \quad \text{and} \quad |\mathbf{v}_3 - \mathbf{v}_4| = |\mathbf{V}_3 - \mathbf{V}_4|.$$

2.2.2 Patch-to-cone construction

We construct the crease between the developable surface and a cone by considering distances between points on the same ruling. As developing a developable surface

to the plane preserves the distances between points on the same ruling, we derive a constraint from comparing two corresponding distances.

To utilize a unified expression for both cases, let $\mathbf{C}_i = \mathbf{V}_1$ and $\mathbf{c}_i = \mathbf{v}_1$ in case of a cone, and let $\mathbf{C}_i = \mathbf{P}_i$ and $\mathbf{c}_i = \mathbf{p}_i$ in case of a cylinder. Furthermore, $\mathbf{V} = \mathbf{V}_3$ and $\mathbf{v} = \mathbf{v}_3$.

Note that for a crease between a developable surface and a cone, every point on the 3D or 2D crease curve is connected to the corresponding vertex $\mathbf{V}$ or $\mathbf{v}$ with a ruling of the constructed cone. We parametrize the points on corresponding rulings in 3D and 2D by $\mathbf{L}_i(t) = \mathbf{C}_i + t\mathbf{R}_i$ and $\mathbf{l}_i(t) = \mathbf{c}_i + t\mathbf{r}_i$. For every pair of corresponding 3D and 2D rulings, we find the crease points $\mathbf{F}_i = \mathbf{L}_i(t^*)$ and $\mathbf{f}_i = \mathbf{l}_i(t^*)$ as points that satisfy

$$|\mathbf{L}_i(t_i^*) - \mathbf{V}|^2 = |\mathbf{l}_i(t_i^*) - \mathbf{v}|^2 \qquad \Longrightarrow \qquad |\mathbf{P}_i - \mathbf{V} + t_i^*\,\mathbf{R}_i|^2 = |\mathbf{p}_i - \mathbf{v} + t_i^*\,\mathbf{r}_i|^2$$

that is, the pair of corresponding points such that the distances between the corresponding points and the 3D and 2D apices are the same.

Solving the above equation for t_i^* results in

$$t_i^* = \frac{1}{2}\frac{|\mathbf{v} - \mathbf{c}_i|^2 - |\mathbf{V} - \mathbf{C}_i|^2}{(\mathbf{v} - \mathbf{c}_i)\cdot\mathbf{r}_i - (\mathbf{V} - \mathbf{C}_i)\cdot\mathbf{R}_i}. \tag{1}$$

Note, that the crease points $\mathbf{F}_i$ and $\mathbf{f}_i$ only provide the location of the crease curve as the intersection of two developable surfaces, and do not always produce a clean and useful result. As discussed in [12] in more detail, a “good" curved crease point $\mathbf{F}_i$ connecting a developable surface patch and a cone has the following characteristics:

- *Valid patch combination:* The computed crease separates a the given developable surface and the constructed cone into four surface patches. Out of the four possible combinations, only two are developable. As discussed in [12], we require that the denominator is greater zero for the values of each ruling, resulting in $(\mathbf{v} - \mathbf{c}_i)\cdot\mathbf{r}_i > (\mathbf{V} - \mathbf{C}_i)\cdot\mathbf{R}_i$.
- *Valid range:* We want to make sure that the crease exists in a suitable range of the developable surface (e. g., does not pass through the apex to the other part of the cone). This property translates to the numerator of t_i^* being greater than zero, that is, $|\mathbf{v} - \mathbf{c}_i|^2 > |\mathbf{V} - \mathbf{C}_i|^2$.

When the above inequalities are satisfied for all sampled points, we have found a valid curved crease. In case of a cone-cone design, we then only interpolate the crease and construct extrusions to $\mathbf{V}_1$ and $\mathbf{V}_2$. In case of the lens design, we compute the crease for the other cone with apex $\mathbf{V} = \mathbf{V}_1$. We then extrude both creases to the corresponding cone apices and construct the cylinder as a loft with parallel rulings between the two curves.

3 Modular Curved-Crease Designs

Equipped with the theory of the previous section, we apply the quad-filling method to each face of a non-planar quad mesh. If the faces of the mesh are planar or not all quads, we can apply the following algorithm to construct a mesh M' with non-planar faces from the given mesh M.

1. Initialize M' with the set of vertices of M and an empty face set.
2. For every face, add a new vertex at a user-specified amount in normal direction from the face's center. The corresponding parameter establishes the degree of non-planarity.
3. For every interior edge, create a face containing the edge's endpoints and the two vertices corresponding to its adjacent faces.

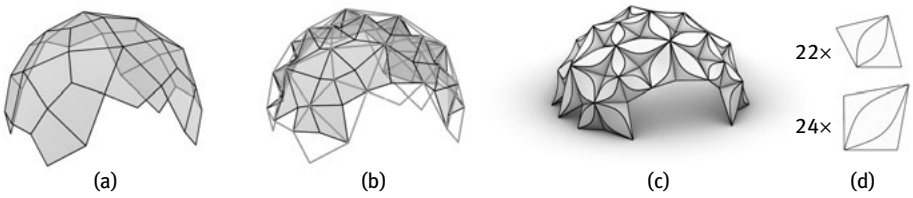

Fig. 6: Illustration of the modular curved-crease design workflow: (a) Initial mesh M. (b) Subdivided mesh M'. (c) Quads filled with curved-crease molecules. (d) Development consisting of two different types of developed molecules.

Development. Applying the quad-filling method from Sec. 2 to a non-planar quad mesh results in a shape comprised of curved-crease modules. Each curved-crease module can be unrolled, resulting in a family of curved-crease patterns with quadrangular boundaries. However there is no guarantee that the sum of developed angles incident to a common vertex is exactly 2π, and thus the pattern might not be globally developable. For fabrication purposes, the decomposition into smaller molecules can be beneficial. Alternatively, knowing the dimensions of the unrolled quads allows the use of other positioning heuristics of the quads, such as polygon nesting [3] or Origamizer-based kirigami patterns [5].

Design variations. Tiling entire surfaces with foldable modules is a design approach used in many fields. Some design variations of foldable lenses are shown in Maleczek et al. [10]. In our setup, each non-planar quad of a mesh can be filled with one of six curved crease molecules (see Fig. 7). This design freedom can be used to explore patterns generated by the composition of curved creases in quads.

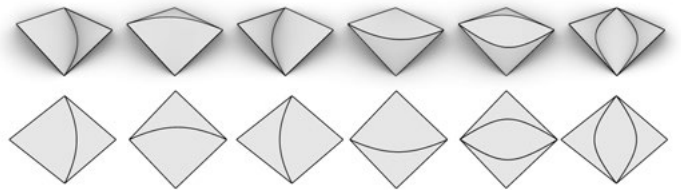

Fig. 7: For every non-planar quad, we have six different choices of the curved crease molecules induced by the same curve (four cone-cone molecules, two lens molecules).

4 Global Origami Development

In the previous sections, we considered the non-planar faces of a quad mesh as individual folded patches, the final mesh being a joining of these parts. However in special cases, the mesh faces can be treated as a connected entity which can be globally developed and fabricated from a single large sheet of material without slits. To achieve this, one must be flexible about the design parameters used to generate each quad, as neighboring quads affect each other, altering each other's interior structures. The global development is discovered through an optimization process whereby the variables are adjusted slightly until all constraints are satisfied or not feasible. A successful development, if found, often times results in a shift of the vertices away from the surface, effectively adding material, and creating a "wrinkling" of the surface.

To begin, we must generate a topologically identical mesh to our given 3D mesh to serve as our crease pattern. This mesh must be planar and contain no overlapping faces or edges. For simple geometries, this can be manually generated, or it can be obtained through a Tutte embedding [15] or the ARAP method [9, 14]. If we choose so, the vertices can be allowed to move in the xy-plane and we make their positions a set of variables in our optimization problem. If we intend for the 2D vertices to be fixed, only the coordinates of the 3D vertices are the variables in our optimization problem.

Solving a fixed-vertex crease pattern, although more heavily constrained, is a simpler case through which we can understand the solver's additional constraints. Because we are dealing with quad meshes, often times a uniform square grid makes for a good initial crease pattern, as shown in Fig. 8.

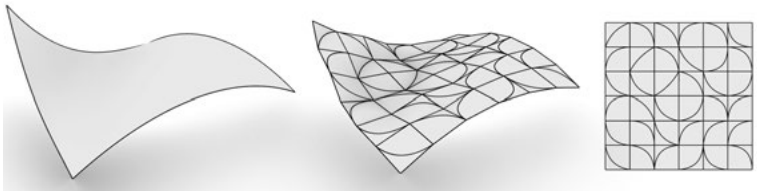

Fig. 8: A wrinkled surface based on a square grid that is globally developable.

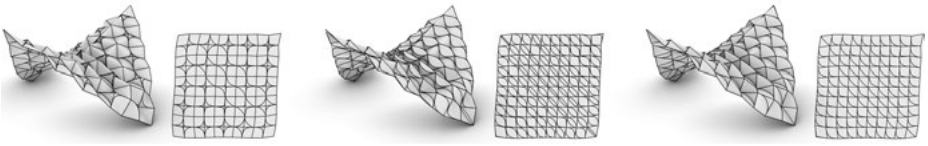

Fig. 9: Three globally developable curved-crease designs that are based on the same mesh and their crease patterns.

4.1 Constraints

To obtain suitable geometries, we impose the following constraints on the 3D (and optionally 2D) vertices of the meshes:

1. Every edge must have identical length in 3D and 2D.
2. Corresponding quad diagonals must be shorter in 3D than in 2D.
3. Every 2D quad's diagonal length should be bounded by above to ensure that the constructed first surface can be realized from a suitable **Z**-central function.

As the solver executes, vertices will be moved around, in some cases dramatically. Therefore adding regularization constraints, such as anchoring points to their original position, might be necessary. Alternatively, it might be helpful to constrain a sparse selection of vertices to the design surface, to keep closeness to the original shape, or constrain some boundary vertices to the boundary edges, to prevent the vertices from scaling down to a point.

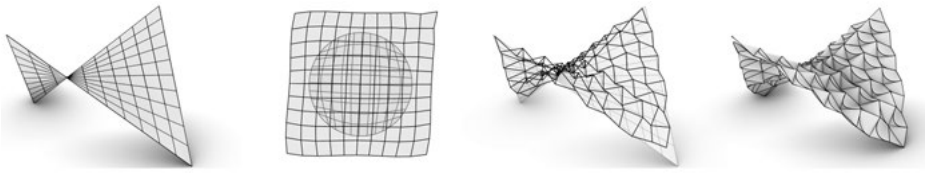

Fig. 10: Illustration of the steps for in Section 4: (a) Initial 2D mesh. (b) Constrained and scaled initial 2D mesh. (c) Wrinkled 3D mesh based on the lengths of the 2D mesh. (d) Mesh with quad faces filled with curved-crease molecules with target diagonals.

4.2 Implementation and practical considerations

We have implemented an optimization process in Kangaroo, a dynamic relaxation engine plugin for Rhino/Grasshopper. The solver uses "goals" and moves the mesh's vertices in real time during calculation, until it finds an equilibrium configuration.

We have found that it is more often the case that the solver desires more material to satisfy constraints, consequently, it is beneficial to scale the developed constrained mesh in relation to the 3D. The authors found that multiplying the edge lengths by 2 to 10 percent of their original lengths led to good results.

Due to the edge-length constraint, this scaling operation will cause the 3D vertices to move away from the surface, resulting in larger angle differences and potential challenges such as surface intersections. To maintain better control over the scaling process, pre-wrinkling the 3D mesh can be beneficial. Pre-wrinkling is a process by which the designer can displace the vertices away from the surface in a manner of their choosing.

During the optimization, vertices will start to move on the target geometry and will therefore scale down to a point if not anchored or limited in motion with respect to the original boundaries.

4.3 Filling a 3D quad with prescribed development

Upon success of the above described optimization process of the 3D and 2D mesh, we aim to fill each 3D non-planar quad with a curved crease molecule whose development perfectly fits the corresponding 2D face. The scale parameter s, as discussed in Sec. 2.1.2, provides a degree of design flexibility. Adjusting this scale parameter affects the intrinsic opening angle of the cone in the case of the cone-cone design, or the intrinsic distance between two points on a cylinder in the case of a lens design.

Determining an appropriate scale parameter that corresponds to the desired opening angle or diagonal distance of the target 2D quad can be framed as finding the root of a scalar function. We have observed that in many valid configurations, this scalar function demonstrates (almost) monotonic behavior.

5 Conclusions and Outlook

We have presented a method that enables the generation of foldable patches or globally developable shapes by filling non-planar quads with curved-crease molecules. In particular, we discussed how to fill a single non-planar quad with two types of curved-crease molecules, and how to optimize for global developability of the resulting shape. In addition, we highlighted design specific considerations, such as the generation of patterns.

As discussed above, further adjustments and investigations are necessary to improve the results when dealing with complex surfaces. Currently, achieving global developability for such surfaces remains challenging. Additionally, exploring the degrees of freedom and tangent continuity between different patches will be subject to future research. The authors are confident, that with appropriate adaptations, the presented strategy allows for different design and fabrication applications.

References

[1] Grasshopper. https://www.grasshopper3d.com/. Accessed: 2023-05-31.
[2] Rhino6/7. https://www.rhino3d.com/. Accessed: 2023-05-30.
[3] Bennell, J. A. and J. F. Oliveira. The geometry of nesting problems: A tutorial. *European Journal of Operational Research,* 184(2):397–415, 2008.
[4] Demaine, E., M. Demaine, D. Koschitz, and T. Tachi. A review on curved creases in art, design and mathematics. *Symmetry: Culture and Science,* 26(2):145–61, 2015.
[5] Demaine, E. and T. Tachi. Origamizer: A practical algorithm for folding any polyhedron. In *33rd International Symposium on Computational Geometry (SoCG 2017),* 2017.
[6] Glaeser G. and F. Gruber. Developable surfaces in contemporary architecture. *Journal of Mathematics and The Arts,* 1:59–71, 03 2007.
[7] Jiang, C., K. Mundilova, F. Rist, J. Wallner, and H. Pottmann. Curve-pleated structures. *ACM Transactions on Graphics,* 38:1–13, 11 2019.
[8] Lawrence, S. Developable surfaces: Their history and application. *Nexus Network Journal,* 13: 701–14, 2011.
[9] Liu, L., L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler. A local/global approach to mesh parameterization. In *Computer Graphics Forum,* vol. 27, pp. 1495–1504. Wiley Online Library, 2008.
[10] Maleczek, R., K. Mundilova, and T. Tachi. Curved crease edge rounding of polyhedral surfaces. *Advances in Architectural Geometry,* 2020.
[11] Mundilova, K. On mathematical folding of curved crease origami: Sliding developables and parametrizations of folds into cylinders and cones. *Computer-Aided Design,* 115:34–41, 2019.
[12] Mundilova, K., E. Demaine, R. Foschi, R. Kraft, R. Maleczek, and T. Tachi. Lotus: A curved folding design tool for Grasshopper. In *Proceedings of the 41st Annual Conference of the Association of Computer Aided Design in Architecture (ACADIA),* ACADIA 2021, pages 194–203, 2021.
[13] T. G. Nelson. Art to engineering: Curved folding and developable surfaces in mechanism and deployable structure design. 2018. PhD thesis.
[14] Sorkine, O. and M. Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing,* vol. 4, pp. 109–16. Citeseer, 2007.
[15] Tutte, W. T. How to draw a graph. *Proceedings of the London Mathematical Society,* 3(1): 743–67, 1963.