Modeling Curved Folding with Freeform Deformations

MICHAEL RABINOVICH, ETH Zurich, Switzerland
TIM HOFFMANN, TU Munich
OLGA SORKINE-HORNUNG, ETH Zurich, Switzerland

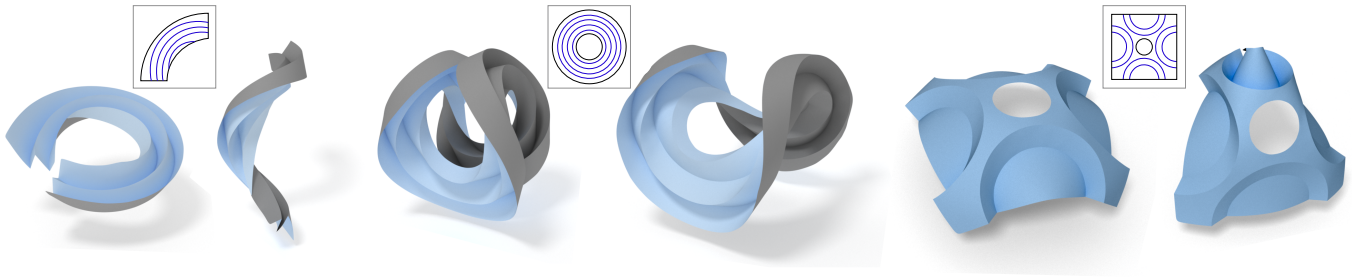


Fig. 1. Curved folded surfaces modeled with our method, along with their crease patterns. Our deformation algorithm is able to simultaneously bend and fold complicated crease patterns using only positional constraints, while automatically finding a valid mountain/valley assignment along the creases. Our framework is suitable for freeform editing and exploration of new curved folded surfaces.

We present a computational framework for interactive design and exploration of curved folded surfaces. In current practice, such surfaces are typically created manually using physical paper, and hence our objective is to lay the foundations for the digitalization of curved folded surface design. Our main contribution is a discrete binary characterization for folds between discrete developable surfaces, accompanied by an algorithm to simultaneously fold creases and smoothly bend planar sheets. We complement our algorithm with essential building blocks for curved folding deformations: objectives to control dihedral angles and mountain-valley assignments. We apply our machinery to build the first interactive freeform editing tool capable of modeling bending and folding of complicated crease patterns.

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

Additional Key Words and Phrases: curved folding, developable surfaces, discrete differential geometry, geodesic nets, shape modeling

ACM Reference Format:
Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2019. Modeling Curved Folding with Freeform Deformations. *ACM Trans. Graph.* 38, 4, Article 1 (July 2019), 12 pages. https://doi.org/10.1145/3355089.3356531

1 INTRODUCTION

There are infinitely many ways to deform a planar sheet without stretching or tearing it. One can either bend it, form sharp creases by folding it, or combine the two. Folding and bending isometries are different by nature, and historically there has been a dichotomy in the study of the two. Smooth bending deformations are typically studied in differential geometry [do Carmo 1976], whereas straight folds are explored in the field of computational origami [Demaine and O'Rourke 2007]. Curved folded surfaces [Huffman 1976] (Fig. 1) can be viewed as a combination of the two, since folding an inextensible sheet along a curve necessitates global bending around the crease. These elegant geometries have garnered the attention of architects, artists, and industrial designers [Buri et al. 2011; Demaine et al. 2011c; Gramazio and Kohler 2014; Pottmann et al. 2015; Tachi 2011, 2013].

The design of a curved folded surface is manual and time consuming and is usually done using an empirical trial and error approach [Demaine et al. 2011a,c]. The known theory on curved folds is confined to a narrow set of folds, and contrary to classical origami, bending and folding instructions are hard to write down and multiple creases must be folded simultaneously [Kilian et al. 2017]. Artists generally pre-crease the paper using a ball burnisher or a CNC plotter before carefully folding and bending, making the process of shape exploration even slower.

Although manual and slow, playing with paper is still the predominant approach for curved folded surface design. Existing works on modeling such surfaces are either limited to previously discovered surfaces [Kilian et al. 2008, 2017] or model a small, partial set of folded surfaces generated by reflections or rotational sweeps [Mitani 2009; Mitani and Igarashi 2011]. Modeling the folding process of novel forms remains a challenge [Demaine et al. 2011c].

In this paper we set out to develop the basic tools for freeform modeling of curved folds, with the objective of aiding the exploration, analysis and study of new curved folded surfaces. Our work builds upon discrete orthogonal geodesic nets (DOGs) [Rabinovich et al. 2018a,b] as a discrete model for C^2 developable surfaces. DOGs

Authors' addresses: Michael Rabinovich, ETH Zurich, Department of Computer Science, Universitätstrasse 6, Zurich, 8092, Switzerland; Tim Hoffmann, TU Munich, Department of Mathematics; Olga Sorkine-Hornung, ETH Zurich, Department of Computer Science, Universitätstrasse 6, Zurich, 8092, Switzerland.

0730-0301/2019/7-ART1 $15.00
https://doi.org/10.1145/3355089.3356531

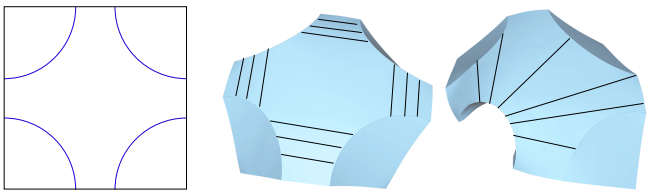


Fig. 2. Folding and bending the same curved crease pattern (left) into two different surfaces, with some of their rulings plotted. Methods that model the rulings explicitly must remesh in order to model these surfaces, since the rulings can change drastically, connecting vertices of different patches to one another. Representing the curved folded surface as a piecewise DOG avoids the need to remesh because a DOG is parameterized by intrinsic invariants – orthogonal geodesics – and does not explicitly encode the developable rulings in its mesh.

are regular quadrilateral meshes where around each vertex all angles are equal. Unlike other computational models for developable surfaces, DOGs do not suffer from locking of various deformation modes [Alessio 2012; Chapelle and Bathe 1998; Grinspun et al. 2003], are not limited by an initial choice of meshing or rulings [Kilian et al. 2008; Solomon et al. 2012; Stein et al. 2018; Tang et al. 2016] (see Fig. 2) and do not require remeshing while deforming the surface [Kilian et al. 2017; Narain et al. 2013; Schreck et al. 2017]. Therefore DOGs are particularly suited for modeling curved folds. Given an input crease pattern, we represent a curved folded model as a collection of DOGs, with boundary constraints enforcing equal discrete geodesic curvature along their intersections, as done in [Rabinovich et al. 2018b].

In practice, deforming a set of DOGs while keeping the geodesic boundary constraints does not usually result in a model that is folded along all creases (see Fig. 3). Part of the difficulty of modeling these deformations stems from the need to fold all curves simultaneously starting from a flat configuration. The primary goal of our work is to deal with this difficulty.

1.1 Contributions

- We present a discrete binary characterization for folds between discrete developable surfaces based on supporting planes along creases, motivated by a novel analysis on curved folded smooth surfaces.
- We use the previous derivation to devise an optimization algorithm capable of enforcing folds while deforming a piecewise DOG, without requiring any folding angles or mountain/valley assignments as input.
- We further derive optional objectives to control dihedral angles and mountain/valley assignments along folds.

Though we use DOGs as an underlying model for developable surfaces, our work and derivations can be applied on top of other discrete models for developable surfaces such as ruling based models [Kilian et al. 2008; Liu et al. 2006; Tang et al. 2016] or models based on discrete shells simulations[Burgoon et al. 2006; Grinspun et al. 2003].

2 RELATED WORK

2.1 Modeling developable surfaces

A smooth surface is called a *developable surface* if it is locally isometric to the plane, or equivalently has zero Gaussian curvature. Though well understood mathematically [do Carmo 1976; Pottmann and Wallner 2001; Spivak 1999], computer aided modeling of developable surfaces has been proven to be a challenge and is an active research area. The primary difficulty lies in finding a discrete model that is able to capture the full set of deformations while keeping the surface developable. Deformations can be extrinsic as well as intrinsic. The latter stretch the surface while keeping it developable, and are used for geometry exploration tasks where the size and shape of the flattened developable surface is unknown [Liu et al. 2006; Rabinovich et al. 2018b; Tang et al. 2016]. A failure of a discrete model to represent the full range of smooth deformations is often termed *locking* [Chapelle and Bathe 1998; Solomon et al. 2012] and is the bane of most discrete developable models. Ruling based models [Kilian et al. 2008; Liu et al. 2006; Solomon et al. 2012; Stein et al. 2018; Tang et al. 2016] are limited to a partial set of extrinsic deformations, while isometry based methods [Burgoon et al. 2006; Fröhlich and Botsch 2011; Goldenthal et al. 2007; Grinspun et al. 2003] do not model intrinsic deformations by design, and are also prone to locking of various bending deformations [Alessio 2012; Chapelle and Bathe 1998], and often must be coupled with dynamic remeshing [Kilian et al. 2017; Narain et al. 2013, 2012; Schreck et al. 2015].

Our work is based on modeling a developable surface as a discrete orthogonal geodesic net (DOG) [Rabinovich et al. 2018a], a model that has been shown both theoretically and empirically to avoid extrinsic and intrinsic deformation locking. We further rely on [Rabinovich et al. 2018b] to explore the shape space of DOGs, but we replace their Laplacian flow based deformation with a sequential quadratic programming (SQP) based algorithm, detailed in Sec. 6.

In addition to the above literature, it is worth mentioning the large body of works focused on designing developable surfaces by fitting a developable to a target shape [Pottmann and Wallner 1999; Stein et al. 2018; Tang et al. 2016] or to a set of boundary curves [Bo et al. 2019; Frey 2002, 2004; Rose et al. 2007]. Rather than modeling freeform developable deformations, these works deal with the challenges of ambiguity when fitting to sparse inputs, as well as approximation quality.

2.2 Curved folding

Curved folded sculptures are beautiful works of art almost a hundred years old, dating back to the 1920's works of Josef Albers in the Bauhaus art school [Adler 2004], and continuing with the investigations of David Huffman and Ron Resch in the 1970's [Huffman 1976; Resch 1974]. This direction in art, though intimately linked to the mathematics of developable surfaces, is mostly driven by physical experiments with paper [Demaine et al. 2011c]. As opposed to smooth developable surfaces [do Carmo 1976] or straight fold origami [Demaine and O'Rourke 2007], the mathematics of curved folding is lagging behind the manual craft and mostly concerns the local behavior of a single folded curved crease [Demaine et al. 2011c; Duncan and Duncan 1982; Fuchs and Tabachnikov 2007]. Notable

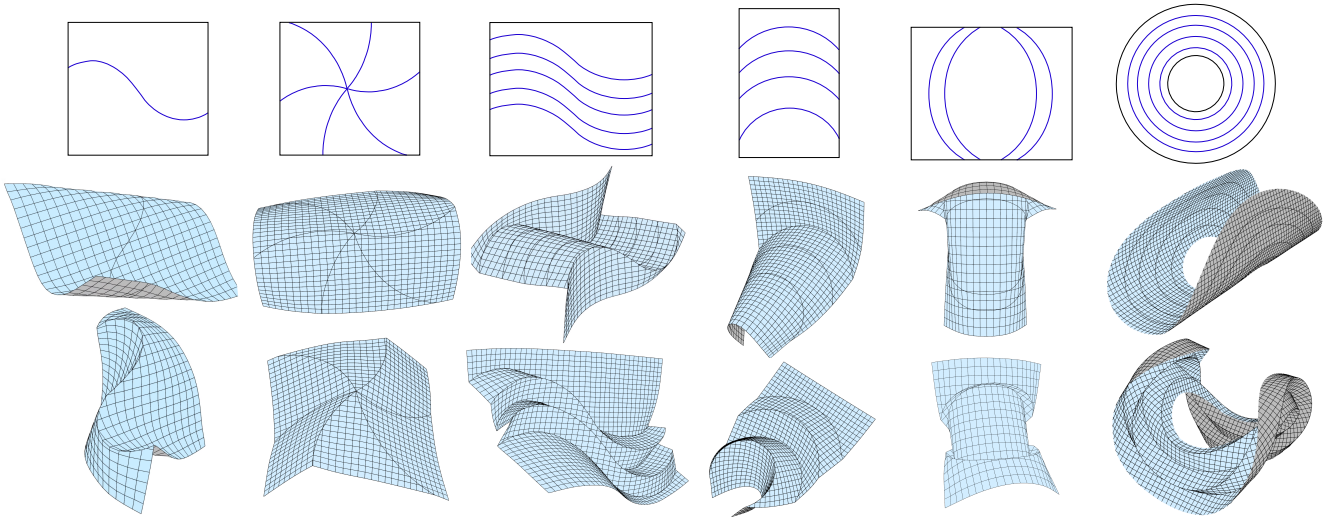


Fig. 3. Comparison of the same deformation objective with and without our folding algorithm. Up: Crease patterns given as input. Center: Applying a positional based deformation objective of the crease patterns without our folding algorithm results in most crease curves being ignored, i.e. not folded. At this stage, one cannot bend these creases without first flattening the surface. Down: Result of applying the same deformation with our folding algorithm. Our folding algorithm simultaneously folds all crease points while deforming a surface by adding a bias that effectively push flat points towards a folded configuration while not affecting already folded points. The crease mountain/valley assignments, which in these examples are in fact fixed given one choice, are determined automatically without any input. The objective of all of these deformations were the same positional constraints defined on mesh vertices or on a part of a crease curve based on a *curve-constraining flow* [Rabinovich et al. 2018b].

crease patterns, such as the Huffmann Tower [Demaine et al. 2011a; Wertheim 2004], are not yet understood [Demaine et al. 2018] and the known mathematics on the folding and bending of multiple curved creases is limited to few particular cases, coupled with a specific folding movement and guided by fixed rulings [Demaine et al. 2015, 2018; Mundilova 2019]. In essence, we do not know much about which crease patterns can fold, and we do not know in which ways they can fold. Unlike straight origami, there are often infinitely many ways of folding and bending a curved crease pattern by varying the dihedral angles as well as the developable rulings along the different creases.

Curved folding was introduced to the geometry processing community by the work of Kilian and colleagues [2008], where the authors devised an algorithm to reconstruct scanned paper curved folded surfaces by estimating their ruling directions, resulting in a mesh with a fixed torsal/planar patch decomposition and mountain/valley assignments. Our work directly deals with the difficulty of folding starting from a flat configuration (Fig. 3) and can also be applied to the model of [Kilian et al. 2008], possibly combined with remeshing to accommodate the locking issue (Fig. 2).

Several works on modeling curved folded surfaces focus on a given subset of folding deformations, such as planar creases generated by reflection [Mitani 2012; Mitani and Igarashi 2011], surfaces generated by rotational sweeps [Mitani 2009] or surface folded with a fixed ruling pattern [Tang et al. 2016]. In [Kilian et al. 2017] the authors simplify the process of fabrication for a wide range of curved folded models using a network of strings, solving the problem of which surface points to pull in order to actuate a folding movement. To model the folding deformation the authors of [Kilian et al. 2017] employ the model of [Botsch et al. 2006] with the remeshing algorithm in [Narain et al. 2012]. Their deformation is guided by mountain/valley assignments of all creases, which are provided as input, as well as prescribed soft constraints on folding angles, as well as a bending objective. We note that prescribing dihedral angles on a single curve results in an undetermined system, while prescribing the folding angles of multiple creases often results in an overdetermined system. There are infinitely many ways to fold a surface with the same prescribed folding angles [Duncan and Duncan 1982; Fuchs and Tabachnikov 1999], however dihedral angles across multiple folds must be compatible with each other [Demaine et al. 2018].

To the best of our knowledge, the first freeform handle based system for curved folding deformations was introduced by Rabinovich et al. [2018b]. They employed multiple DOGs bound by a set of stitching and flattability constraints to model curved crease deformations. Our work builds and extends upon [Rabinovich et al. 2018b] by deriving a discrete characterization for a fold along a crease, allowing us to devise a point handle based editing system that ensures folding of the creases of a given pattern rather than smoothly ignoring them, all without requiring the user to specify dihedral angles or mountain/valley assignments (see Fig. 3). In cases where more explicit control is desired, we also derive simple quadratic constraints to control dihedral angles along folds, as well as mountain/valley assignments – a degree of freedom that is often only available along one curved crease (see Fig. 13 and Fig. 14).

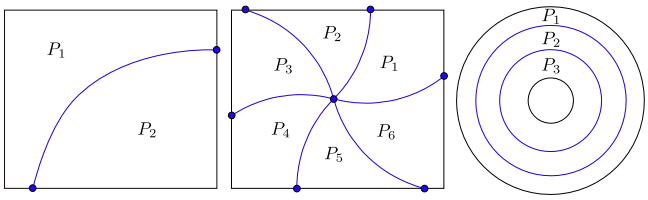


Fig. 4. Curved crease patterns, decomposing a pattern into multiple components P_i and intersecting at crease vertices. Boundary curves in black, crease curves in blue.

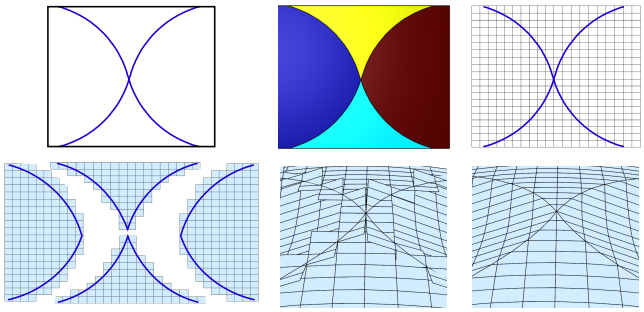


Fig. 5. Representation of a discrete curved folded surface, as done in [Rabinovich et al. 2018a]. Top left: A given curve arrangement, representing the domain via its boundary (in black), and two curves intersecting at their inflection point in the center of the domain (in blue). Top center: The curves segment the domain into four patches, each intersecting with two other patches along a segment of a crease curve. Top right: Placing an orthogonal grid on top of the crease pattern. Bottom left: We model the surface by a separate DOG for each patch, where faces intersecting the crease curves are duplicated for the different patches. Each pair of intersecting DOG patches satisfies the continuity constraints specified in [Rabinovich et al. 2018a]. Bottom center: A closeup on a deformation of the planar model. Bottom right: Culling the extraneous parts of the duplicated faces.

3 SETUP

3.1 Definitions

Throughout the paper, we use the following definition for a curved folded surface:

Definition 3.1. A surface S is called a curved folded surface if it is locally isometric to the plane and can be written as a finite union $S = \bigcup S_i$ where each S_i is a C^2 developable surface termed a *patch*, and the intersections of different patches $S_i \cap S_j$ are either empty or are C^2 curves.

By this definition, a smooth developable surface is a curved folded surface with a single patch. The definition is suitable for various topologies, such as a cylinder, but throughout this paper we work with surfaces that are isometric to a subset of $\mathbb{R}^2$. Borrowing terms from [Demaine et al. 2011b; Demaine and O'Rourke 2007], we often refer to the surface *crease pattern* as the planar domain P isometric to S, subdivided into patches P_i (flattened S_i), whose intersection curves are the flattened creases (see Fig. 4). Flattened creases with nonzero curvature are said to be curved, while those with vanishing curvature are straight. A crease might be partly curved and partly straight, or curved almost everywhere but with inflection points where the curvature vanishes (see Fig. 3, the first and third models from the left). The flattened domain boundaries together with the flattened crease curves form a planar arrangement [Grünbaum 1972], inducing a planar graph that decomposes P into the planar faces P_i. The vertices of this graph are the intersection points of the curves with each other or the boundary curves, which we call *crease vertices*. The *edges* of this graph are the pairwise intersections of the various patches, and we refer to the inner points of these curves as *crease points*, i.e., the points on these curves that are not *crease vertices*. We say that S is *folded* at a crease point p if at that point the patches S_i, S_j sharing it have a tangential discontinuity (see Fig. 3).

We are interested in deformations of curved folded surfaces that keep them curved folded. Viewed separately on each patch P_i, these deformations are C^2, though they often introduce folds along the creases as tangent plane discontinuities of neighboring patches. In particular, we are interested in continuous deformations, or deformation flows [Rabinovich et al. 2018b], which we refer to as curved folding flows. We denote these flows by a continuous map $S(t), 0 \le t \le 1$, where each $S(t)$ is a curved folded surface, and the flow is C^2 when restricted to each patch. We often look at the case where the starting point $S(0)$ is planar. We apply our tools to model isometric curved folding flows, which we also refer to as *folding*. Such flows can be used to model physical paper or sheet metal folding, though most of our observations and tools can also be used to model curved folding flows that stretch a developable surface while keeping it developable. Non-isometric developable deformations can be useful for design tasks where the a priori flattened shape is unknown [Rabinovich et al. 2018a,b; Tang et al. 2016].

3.2 Model

We follow the work of [Rabinovich et al. 2018b] by modeling each patch S_i as a discrete orthogonal geodesic net, together with alignment constraints on the shared boundaries. Our input is an arrangement of curves representing our crease pattern. On top of this arrangement we place an orthogonal grid while ensuring that every vertex of the arrangement lies on a grid line. We then split the grid into overlapping patches, sharing the faces where curves pass. Finally, we compute the intersections of the curves with the grid edges and represent the resulting curve points as linear combinations, one for each patch sharing that curve point. See Fig. 5 for an illustration. Following [Rabinovich et al. 2018a] we maintain continuity of curve points along edges while penalizing deviation of the edge lengths across duplicated quads.

3.3 Desiderata

Our goal is to develop tools for the exploration of curved folded shapes on top of piecewise DOGs by means of deformations. Our choices are guided by the two following ground rules for deforming DOGs: (1) Perform homotopy based optimization, and (2) Minimally constraining the DOGs.

Homotopy based optimization is motivated both theoretically and empirically: Modeling DOGs requires solving highly constrained

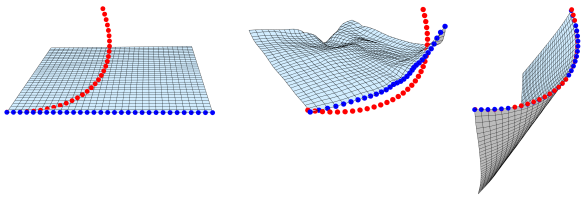


Fig. 6. Setting soft positional constraints on a curve (left). The same optimization algorithm fails completely when setting all constraints at once, returning a mesh that does not satisfy the non-linear DOG constraints (center). In contrast, using a *curve-constraining flow* [Rabinovich et al. 2018b] returns a smooth DOG (right). The latter approach is a homotopy based method that interpolates the positional constraints.

and nonlinear optimization problems, yet the theory of DOGs guarantees the existence of nearby solutions if one starts at a feasible point. In fact, generally the shape space of DOGs is a smooth manifold [Rabinovich et al. 2018b]. This observation is useful in practice: DOGs exploration has been demonstrated to perform well using smooth flows or homotopy based optimization methods both for handle based deformation tasks as well as more complicated deformations such as curve-constraining flows [Rabinovich et al. 2018b] (see Fig. 6).

Minimally constraining the DOGs. Since DOGs are already heavily constrained objects, one needs to carefully choose which quantities to fix by hard constraints, and which to optimize using soft constraints. This is essential in order to avoid locking or ill-posed problems in case the constraint gradients are linearly independent [Rabinovich et al. 2018b]. In particular, the rigidity analysis in [Rabinovich et al. 2018a] demonstrates that one cannot fix all edge lengths, or likewise demand a DOG to also be a Chebyshev net. We note however that this can be done approximately and to a low tolerance, because a smooth orthogonal geodesic net is Chebyshev, and it admits a rich set of exact isometries. Our folding constraints in Sec. 4 are chosen such that they can be satisfied *exactly*. They capture an important characteristic of curved folded surfaces: a folded crease point remains folded under small deformations.

4 FOLDING CREASE PATTERNS

In this section we explore the different ways one can fold a given crease pattern. Our result is a discrete combinatorial characterization for the local existence of a fold in a piecewise DOG.

4.1 The smooth and combinatorial degrees of freedom around a single curved crease

Straight creases are rather boring, mathematically speaking. Straight lines can only be folded as in classical origami, i.e., by keeping them straight [Demaine et al. 2015], unless one first folds a crease by 180 degrees, such that the two incident sheets coincide. Hence a folding of a single straight crease can be described by a single real number representing the constant dihedral angle between the incident planes. There are infinitely many ways, or degrees of freedom, to fold a curved crease. If S is a surface with a folded crease, and P is its flattened isometric reference, then one can locally deform the curved surface S by freely deforming the crease curve, as long as

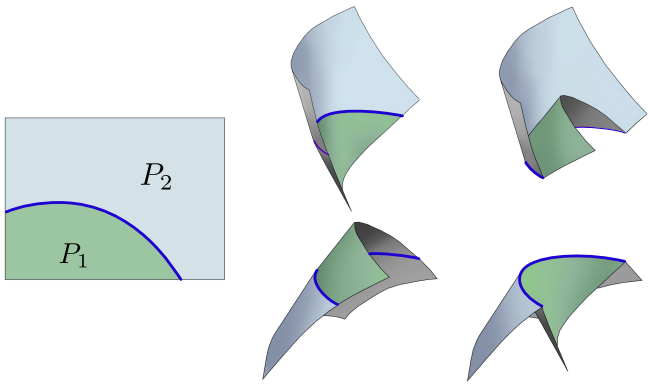


Fig. 7. Illustration of the combinatorial degrees of freedom in curved folding. If a crease pattern of a curved folded surface (left) is isometrically folded such that a given curve lies in some configuration in $\mathbb{R}^3$, then there are only two smooth surfaces that isometrically flatten into the crease pattern (center). One can also choose a different surface for each patch P_1, P_2, resulting in two other, curved folded surfaces (right).

the absolute value of the crease curvature stays greater than its flattened curvature in P [Fuchs and Tabachnikov 1999]. Up to a rigid motion, a curve is defined by its curvature and torsion functions.

One can flip this point of view: Given a planar domain P, a curve on the domain $\gamma(t)$ and a deformed, isometric space curve $\Gamma(t)$ with greater absolute curvature than that of $\gamma(t)$, there are only two smooth surfaces isometric to P passing through $\Gamma(t)$ such that the unfolding of the surface to the plane maps $\Gamma(t)$ to $\gamma(t)$ [Fuchs and Tabachnikov 2007] (see Fig. 7 left and center).

If one permits the surface S to have a fold along $\Gamma(t)$ but remain smooth around it, then there are four possible surfaces: two of them are smooth, while the other two are folded along the curve (see Fig. 7). If $S = S_1 \cup S_2$ is folded along $\Gamma(t) = S_1 \cap S_2$ then the angle between the tangent planes of S_1, S_2 along the curve is called the folding angle, which we denote by $\theta(t) > 0$. Unlike the case of a straight crease, $\theta(t)$ often varies along the curve. The bigger the curvature of $\Gamma(t)$, the bigger the folding angle. If $\kappa(t)$ is the curvature of the space curve $\Gamma(t)$, $\kappa_g(t)$ is its geodesic curvature, which is also the curvature of $\gamma(t)$, then $\kappa_g(t) = \kappa(t) \cos \frac{\theta(t)}{2}$, implying that the osculating plane of the crease $\Gamma(t)$ bisects the tangent planes of the smooth patches intersecting at $\Gamma(t)$ [Duncan and Duncan 1982; Kilian et al. 2008]. The folding angle does not dictate the shape of the surface, as different surfaces can be generated with the same $\theta(t)$ by varying the torsion of $\Gamma(t)$, thereby changing the ruling pattern of each developable patch. The connection between $\theta(t)$, the curvature and the torsion of $\Gamma(t)$, the curvature of $\gamma(t)$ and the rulings of each incident patch is further detailed in [Demaine et al. 2018].

To summarize, the shape of a curved folded surface S with a single curved crease $\Gamma(t)$ can be locally described by two real functions for the curvature and torsion of $\Gamma(t)$ under the condition of sufficient absolute curvature, as well as an additional combinatorial parameter distinguishing between four possible surfaces, two of which have a fold.

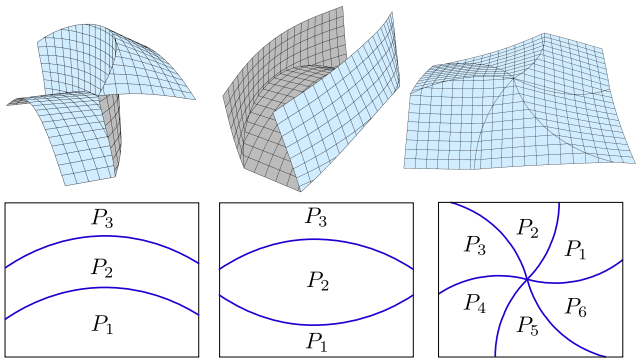


Fig. 8. Propagation of constraints in crease patterns. Bottom row: Crease patterns. Top row: Curved folding of the crease patterns. Left and center columns: A deformation in the patch P_1 dictates most of the shape of the patch P_2, which in turn dictates most of the patch P_3. The propagation of deformation is generally global, and depends on the directions of the rulings. In these cases one can choose a mountain/valley assignment for one fold, which already determines the M/V assignment of the next fold (left: valley-mountain, center: valley-valley). Right column: In more complicated crease patterns, for instance those with a vertex, the process is more involved, as there are also some compatibility conditions the patches must satisfy.

4.2 The combinatorial parameters of multiple creases

The previous analysis explains the local behavior of curved folding around a single curve. Understanding crease patterns globally still remains a challenge. In essence, deforming one patch propagates a global deformation of the patch on the other side of the crease, a process that depends on the locations of the creases and the possibly changing ruling lines along the developable. When there are multiple creases, the propagation dictates the shape of other patches. The process becomes more complicated when some creases intersect, due to compatibility constraints (see Fig. 8).

Generally speaking, one may be able to choose between four different configurations of the surface at one crease, but this choice already fixes the patch shape for nearby creases. The combinatorial degrees of freedom that remain are whether each crease is folded or not (see Fig. 3). The difficulty in modeling folding of a planar surface stems from the fact that these combinatorial choices often need to be enforced at the beginning of the folding process, as explained by the following theorem.

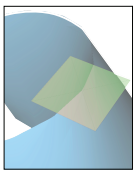
Theorem 4.1. *Let $S(t)$ be a curved folding flow and let $p(t_k)$ be a point on a curved crease of $S(t)$ lying on two patches $S_1(t), S_2(t)$ at a given time in the flow, $t = t_k$. If $p(t_k)$ is not a planar point on $S_1(t_k)$ (or equivalently on $S_2(t_k)$), then there exists an $\epsilon > 0$ such that one of the following holds:*

(1) $S(t)$ is folded at $p(t)$ for every $t \in (t_k - \epsilon,\ t_k + \epsilon)$;
(2) $S(t)$ is not *folded at $p(t)$ for every $t \in (t_k - \epsilon,\ t_k + \epsilon)$.*

Proof. Let $\kappa(p(t))$ be the crease curvature at the point $p(t)$, and let $\kappa_g(p(t)) = \kappa(p(t)) \cos\frac{\theta(p(t))}{2}$ be the crease's flattened (geodesic) curvature. If $\theta(p(t)) \neq 0$, the claim follows from the discontinuity of the two tangent planes at $p(t)$: a folding corresponds to a different choice of the tangent planes, forming an angle of $\theta(p(t))$ with each other, and any small continuous deformation cannot move from a folded to a non-folded configuration or vice-versa. Finally, the non-planarity of $p(t_k)$ implies $\theta(p(t_k)) \neq 0$, since $\theta(p(t_k)) = 0$ would mean that the normal curvature of the crease curve is 0, and therefore the tangent of the crease curve is parallel to the ruling direction. But by Lemma 12 and Corollary 16 in [Demaine et al. 2015] this implies that the curve has a kink at $p(t_k)$, contradicting the fact that $S(t)$ is C^2 when restricted to the patches $S_1(t), S_2(t)$. □

Therefore it is impossible to fold a crease point that is non-planar. For a non-planar point that is not folded, any small deformation keeps it that way. Folding can only happen after flattening the point, and if the surface is already folded, any small deformation keeps it folded. Thus, the decision whether to fold or not can only be done when the crease points are planar, and no extra care needs be taken if the crease is already folded. With this in mind, we note the following observation.

Theorem 4.2. *A non-planar curved crease point p on a curved folded surface S is folded if and only if the osculating plane of the crease curve at p is locally a supporting plane for the patches S_1, S_2 intersecting at p.*



This follows directly from the fact that the tangent planes on both sides of a crease curve coincide if the surface is smooth there, but along a folded crease they are reflections of one another through the crease curve's osculating plane. Planar crease points along a curved crease, while not folded, still satisfy this constraint, since around these points the tangent planes are exactly the same as the osculating plane of the curve, though even the slightest surface deformation might change that.

4.3 Discretization

We saw that folding happens exactly when both sides of the surface around a crease are in the same half-space of the osculating plane of the crease curve. We discretize this condition by constraining the tangents of the discrete parametric (grid) lines of the two DOG patches to be on the same side of the crease curve's discrete osculating plane. See Fig. 9 for the notation. The DOG edges intersecting the crease can be considered as discrete surface tangents originating at the crease points. In the notation of Fig. 9, each of the two patches has its own duplicate of the edge (e_1 or e_2) intersecting the crease curve. In the starting, flat configuration the two edges coincide, but a folding movement creates a discontinuity between them. We denote the discrete surface tangents on both sides of the crease curve as $t_1 = \frac{e_1}{\|e_1\|}, t_2 = \frac{e_2}{\|e_2\|}$. The binormal of the crease curve, i.e., the normal of its osculating plane, is $B = \frac{e_f \times e_b}{\|e_f \times e_b\|}$, noting that e_f, e_b always coincide for both patches.

The supporting plane constraint can be written as

$$\mathrm{sgn}(\langle t_1, B\rangle) + sgn(\langle t_2, B\rangle) = 0 \tag{1}$$

with

$$\mathrm{sgn}(x) = \begin{cases} -1 : \text{if } x < 0, \\ 0 : \text{if } x = 0, \\ 1 : \text{if } x > 0. \end{cases} \tag{2}$$

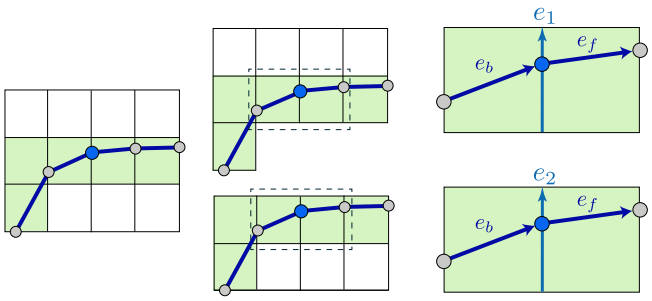


Fig. 9. Notation for edges in a discrete crease pattern. Left: a flat DOG with a discrete crease curve. Center: Following [Rabinovich et al. 2018b], we represent creases by duplicating the quads that contain the crease curve, which results in different connected components, or patches. The positions of the vertices on the curve, i.e., its intersection points with the grid edges, are constrained to match on both patches. Right: Notation for a duplicated grid edge (e_1, e_2) intersecting the crease curve at the blue point, and the two crease edges e_f, e_b.

Constraint (1) can be simplified by replacing B with the cross product $e_f \times e_b$. Furthermore, if one assumes an isometric deformation, it is possible to replace t_1, t_2 by the non-normalized edges e_1, e_2, arriving at:

$$\text{sgn}(\langle e_1, e_f \times e_b \rangle) + sgn(\langle e_2, e_f \times e_b \rangle) = 0. \tag{3}$$

In Sec. 6 we show how to plug this constraint into an optimization framework to model folding and bending of curved folded DOGs (see Fig. 3).

4.4 Discussion

There are multiple equivalent characterizations for a folded crease over a curved folded surface. We now point out some key properties of our chosen discretization (1) and briefly discuss how these properties are not satisfied by other possible constraint choices.

Suitable for homotopy based optimization methods. Our constraint is satisfied on a flat mesh. In this sense we consider a point along a curve with zero normal curvature as both folded and not folded.

Minimal and generally non-intrusive. Once a curved crease on a piecewise DOG is folded, one no longer needs to take explicit care for it to stay folded. The effect of Eq. (3) on an already folded surface is null. The tangent discontinuity caused by the folding implies that a folded crease remains folded under local deformations, and in order for it to become unfolded, one needs to first flatten it, as is the case for a piecewise smooth curved folded surface. We also capture the converse: a discrete curved crease can only be folded when starting from a planar point (Theorem 4.1).

An alternative constraint for folding could be e.g. enforcing discontinuities along the tangents t_1, t_2, but this results in losing the feasibility of the flat models. Moreover, in the discrete case a minor discontinuity can still arise even though there is no fold, i.e., where Eq. (1) is not satisfied, thus numerically giving the impression of a fold when visually there is none. Another option is to define folded configurations as those satisfying a similar but simpler smooth constraint:

$$\langle t_1, B \rangle + \langle t_2, B \rangle = 0. \tag{4}$$

This condition is satisfied exactly in flat models and in any piecewise smooth curved folded surface, since tangent planes along a folded creased curve are reflections of each other w.r.t. the osculating plane of the crease curve. However, this condition is not satisfied *exactly* on every folded piecewise DOG (as evident by all models in this paper). An exception to this is the class of curved creases with zero torsion, in which case the folds are simply formed as a global plane reflection [Mitani and Igarashi 2011]. Therefore, enforcing constraint (4) as a hard constraint is too restrictive in practice, while enforcing it softly creates a condition that, unlike the smooth case, does not vanish once a crease is folded and hence is not minimal in our sense.

5 FOLDING ANGLES AND MOUNTAIN-VALLEY ASSIGNMENTS

In this section we propose tools to constrain the folding angles and their direction (mountain or valley) during deformation, in order to provide designers with additional expressive and intuitive control. We first show how to constrain folding angles, i.e., the angles between the tangent planes around a crease. We implement this by constraining DOG tangent angles, resulting in a simple quadratic constraint. We then devise a tool to differentiate between mountain and valley folds. Our derivations work for both curved and straight origami creases.

5.1 Folding angle

A folding deformation can be seen as a rotation of the surface patches' tangent planes hinged on the tangent of the crease curve. The tangent of a straight fold is constant, and so is the folding angle, while on a curved crease, the tangent varies and often the folding angles change along the crease. In both cases, if the folding angle at a given point is θ, then the surface tangent vectors on both sides of the crease that are orthogonal to the crease tangent form an angle of θ, while the surface tangent vectors that are parallel to the crease remain parallel to each other. The following lemma shows the relation between the angle formed by surface tangent vectors

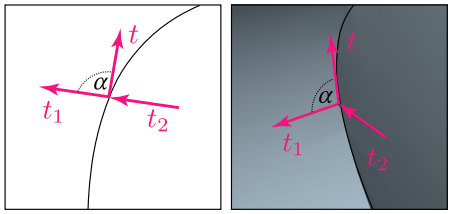


Fig. 10. Left: A flattened configuration of a crease curve and its tangent t, forming an angle α with surface tangents t_1, t_2, which are equal in this flat state ($t_1 = t_2$). Right: A folded isometric configuration with a tangent discontinuity $t_1 \neq t_2$. Lemma 5.1 shows the connection between the angle α and the folding angle θ in the smooth case, stating that $\langle t_1, t_2 \rangle = \cos^2\alpha + \sin^2\alpha \cos\theta$.

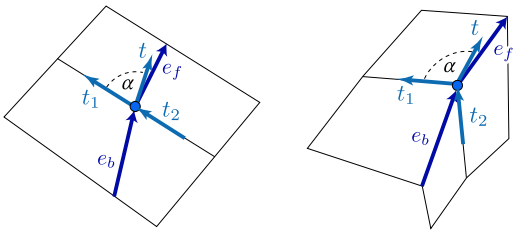


Fig. 11. Discretizing Lemma 5.1 (see Fig. 10) at a crease point by using the normalized DOG edges emanating from the crease point as surface tangents.

that are equal in the flattened configuration and the folding angle (see Fig. 10):

Lemma 5.1. *Let t_1, t_2 be surface tangent vectors on two sides of a crease curve at a given point p that are equal to each other in the isometrically flattened state of the developable surface. Let t be the crease curve tangent at p. Assuming the surface went through a curved folding isometric deformation and the folding angle at crease point p is θ, the surface tangent vectors satisfy:*

$$\langle t_1, t_2 \rangle = \cos^2\alpha + \sin^2\alpha \cos\theta, \tag{5}$$

where α is the angle between t and t_1. Note that this angle is preserved under isometry.

Proof. We denote by t, n, b the vectors of the Frenet frame of the curved crease at p. We wish to express the surface tangent vectors t_1, t_2 in the local coordinates of this Frenet frame. In the flat isometric configuration, t_1, t_2 coincide and can be written as $\cos(\alpha)t + \sin(\alpha)n$. A folding angle of θ means that relative to the Frenet frame of the curve, the surface tangent on one side of the curve was rotated by angle $\frac{\theta}{2}$ about the crease curve tangent t, and the surface tangent on the other side by $-\frac{\theta}{2}$. Thus w.l.o.g.

$$t_1 = \cos(\alpha)t + \sin(\alpha)\left(\cos\left(\tfrac{\theta}{2}\right)n + \sin\left(\tfrac{\theta}{2}\right)b\right),$$
$$t_2 = \cos(\alpha)t + \sin(\alpha)\left(\cos\left(\tfrac{\theta}{2}\right)n - \sin\left(\tfrac{\theta}{2}\right)b\right).$$

The proof is concluded by computing $\langle t_1, t_2 \rangle$ and plugging in the trigonometric identity $\cos\theta = \cos^2\frac{\theta}{2} - \sin^2\frac{\theta}{2}$. □

We discretize Lemma 5.1 by looking at angles between edges of the DOG emanating from crease points (see Fig. 11). Using the notation of Fig. 11, we discretize the tangent of the crease curve at a given point by looking at the incident edge vectors e_f, e_b:

$$t = \frac{\|e_b\|e_f + \|e_f\|e_b}{\left\|\|e_b\|e_f + \|e_f\|e_b\right\|}. \tag{6}$$

If the two edge vectors e_b, e_f are not collinear, t as above is the tangent at the point to the unique circle passing through the point and its two neighbors.

Under an isometric deformation, t_1, t_2 are linear in the vertex positions, α is constant and Eq. (5) is quadratic.

5.2 Mountain/valley assignments

As mentioned in Sec. 4.1, there are a few combinatorial degrees of freedom in choosing the type of fold, i.e., the choice of surfaces on each side of the curve (Fig. 7). We follow [Demaine et al. 2015] to distinguish between the two types of folded configurations in Fig. 7 by calling one choice a *mountain fold* and the other a *valley fold*. We would like to emphasize that for straight folds, this degree of freedom always exists, but on curved creases it often does not. In fact, in many crease patterns it is often only possible to choose one mountain/valley (M/V) assignment, and the remaining assignments are determined by the propagation of the rulings, leaving only the combinatorial degrees of freedom of whether a crease is folded or not, embodied by Eq. (1). We distinguish M/V folds by looking at whether a tangent of one surface patch is above or below the tangent plane of the second surface patch at the crease point, for a consistent choice of orientation. This can be achieved by the following constraint:

$$\langle t_1, t \times t_2 \rangle \leq 0, \tag{7}$$

where t is the tangent of the oriented crease curve and $t \times t_2$ is the normal of the tangent plane of the second surface patch (the one that has t_2 as a tangent vector). The orientation of t determines whether a mountain or a valley fold is chosen. By the cyclic property of the triple product, the left hand side of Eq. (7) is also equal to $\langle t_2, t_1 \times t \rangle$. As we are only interested in the sign of the left side of (7), we can replace t with the simpler $t^* = \|e_b\|e_f + \|e_f\|e_b$, which is linear under isometry.

To simplify notation for Sec. 6, we reformulate our mountain/valley constraint with an equality by using the Heaviside step function:

$$\mathrm{H}(x) = \begin{cases} 0 : \text{if } x \leq 0, \\ 1 : \text{if } x > 0, \end{cases} \tag{8}$$

and write the mountain/valley condition as:

$$H(\langle t_1, t^* \times t_2 \rangle) = 0. \tag{9}$$

6 OPTIMIZATION

We employ the tools developed in Sec. 4 and Sec. 5 to devise a simple folding and bending algorithm for deforming piecewise DOGs. The algorithm aims to minimize an objective function while keeping the DOG constraints and ensuring the formation of folds along all crease curves and a specific M/V assignment on a crease when such an assignment is given as input.

6.1 Problem setup

We model our curved folded surfaces as a quad mesh, with a separate connected component for each patch. We denote the set of n mesh vertices in $\mathbb{R}^3$ by V, the vertex positions (variables) by $x \in \mathbb{R}^{3n}$, and the quad mesh faces by F. Each connected component is a DOG, i.e., it has the connectivity of a subset of $\mathbb{Z}^2$ and satisfies the DOG angle constraints [Rabinovich et al. 2018a], which we denote as $\phi_{d_i}(x) = 0, 1 \leq i \leq m$.

We are interested in deformations that fold the surface along all crease curves in a given crease pattern using Theorem 4.2 and enforcing Eq. (3) and optionally the mountain/valley assignment Eq. (9). We enforce these constraints on all crease points, which

are points on crease curves that are not crease vertices, with the exception of crease points that have the following degeneracies on the flattened mesh (see Fig. 12):

(1) degenerate osculating plane: crease points with a curvature smaller than a threshold κ_ϵ;
(2) degenerate edge: crease points on an edge, splitting it into two parts where one is shorter than ε_r% of the other;
(3) degenerate angle with the intersecting DOG tangent: crease points where the tangent directions t_1, t_2 form an angle with one of the edges e_f, e_b that is smaller than ε_α.

We use the constants κ_ϵ = 1e-5, $\varepsilon_r = 5, \varepsilon_\alpha = 3^\circ$. We denote the set of all folding constraints and the mountain/valley constraints (Eq. (9)) by $\phi_{f_j}(x) = 0, 1 \le f_j \le n_f$.

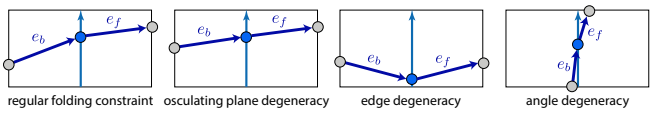


Fig. 12. A folding edge constraint defined on a blue crease point splitting the blue edge and degenerate cases where we do not enforce the constraint. From left to right: A regular folding edge constraint, instabilities in the osculating plane's normal as $\frac{e_b \times e_f}{\|e_b \times e_f\|}$ caused by e_b, e_f being almost collinear, degenerate edges as one part of the edge split by the blue crease point is comparably very short, and lastly a very small angle between e_b and the DOG edge crossing the blue point. An angle degeneracy often occurs before or after an edge degeneracy.

We only enforce the DOG angle constraints and the folding and mountain/valley constraints as hard constraints. The objective function, which we denote by $f(x)$, is composed of a weighted sum of a bending objective, soft positional constraints, soft dihedral constraints and soft patch-continuity constraints.

Isometry is enforced as a soft constraint as advised by the degrees of freedom analysis in [Rabinovich et al. 2018a,b], but we emphasize that all our results have an average relative edge stretch that is less than 0.003, and a maximum stretch below 0.004, where our surfaces are normalized to have an average edge length of 1. As opposed to [Rabinovich et al. 2018b], we also encode the linnear continuity constraints between patches as a soft constraint, as we have noticed a significant improvement in the quality and smoothness of crease patterns when these are enforced as a soft penalty with a large weight, and our results have an average continuity deviation of 0.0002 and a maximum of 0.0035. We note that the constrained shape space analysis in [Rabinovich et al. 2018b] only concerns the DOG angle constraints, and complicated crease patterns give rise to a large set of additional linear constraints.

The problems we solve in this paper can be written in the form:

$$\begin{aligned} &\arg\min_x f(x) \\ &\text{subject to} \\ &\phi_{d_i}(x) = 0, \quad i = 1, \dots, m, \\ &\phi_{f_j}(x) = 0, \quad j = 1, \dots, n_f, \end{aligned} \tag{10}$$

where f is specified in Sec. 6.2. We handle the combinatorial constraints ϕ_{f_j} by using a penalty based method (Sec. 6.3), and solve our problem with an iterative sequential quadratic programming (SQP) solver with a line search (Sec. 6.4). The line search strategy ensures that the DOG angle constraints ϕ_{d_i} are satisfied numerically while the combinatorial constraints ϕ_{f_j} are satisfied exactly.

6.2 Objectives and convex Hessian approximations

Our objective f is composed of a weighted sum of various functions measuring bending, stretch, positional constraints and dihedral angles. We use an integrated squared mean curvature bending objective taken from [Rabinovich et al. 2018b], and we exploit the fact that it is quadratic and convex under isometric deformations [Bergou et al. 2006]:

$$f_{\mathrm{H}}(x) = 0.5 x^t (L^t M^{-1} L) x, \tag{11}$$

where L is the DOG Laplacian and M is a diagonal mass matrix defined by the DOG vertex area [Rabinovich et al. 2018b].

We employ Lemma 5.1 to constrain the folding angle at a given crease point using the constraint

$$\phi_{\mathrm{d}_{c^i}}(x) := \langle t_1^i, t_2^i \rangle - \cos^2(\alpha^i) - \sin^2(\alpha^i)\cos(\theta^i) = 0, \tag{12}$$

where c^i is the index of the crease point along the edge defined as a linear combination of two vertices, t_1^i, t_2^i, α^i are as defined in Sec. 5.1 and Fig. 11, and θ^i is the desired dihedral angle at the crease point c^i. Under isometry t_1^i, t_2^i are linear in the net vertex locations, α^i is fixed, and the constraint is quadratic.

Let e be an edge on the net mesh, l_e its length and l_e^0 the length in the reference net mesh. We define the following quadratic isometry constraints:

$$\phi_{\mathrm{iso}}(x)_e := l_e^2 - {l_e^0}^2 = 0. \tag{13}$$

We maintain continuity along the patches with a set of linear equality constraints on duplicated crease points [Rabinovich et al. 2018b], which we denote by $\phi_{\mathrm{cont}}(x) = 0$. Lastly, we allow the user to specify positional constraints on vertices or crease edge points, including constraints requiring two points to have the same coordinate, as used in the creation of the ring at Fig. 15 and the annulus at Fig. 1 (also see accompanied video). We denote this user defined set of constraints by $\phi_{\mathrm{pos}}(x) = 0$. We enforce the dihedral, positional, isometry and patches continuity constraints in a soft manner by using a penalty on their squared deviation, denoted accordingly by $f_{\mathrm{D}}(x), f_{\mathrm{pos}}(x), f_{\mathrm{iso}}(x), f_{\mathrm{cont}}(x)$. These sum of squared objectives are not convex, and we replace their Hessian in our optimization with their Gauss-Newton's Hessian approximation. We do the same for $\sum \|\phi_{f_i}^*(x)\|_2^2$.

The objective we optimize is then:

$$f(x) = w_{\mathrm{H}} f_{\mathrm{H}} + w_{\mathrm{pos}} f_{\mathrm{pos}} + w_{\mathrm{D}} f_{\mathrm{D}} + w_{\mathrm{iso}} f_{\mathrm{iso}} + w_{\mathrm{cont}} f_{\mathrm{cont}}. \tag{14}$$

Throughout the paper, unless stated otherwise, we use $w_{\mathrm{H}} = 1$, $w_{\mathrm{pos}} = 5$, $w_{\mathrm{D}} = 100$, $w_{\mathrm{iso}} = 20000/|E|$, $w_{\mathrm{cont}} = 1\mathrm{e}4$, where $|E|$ is the number of edges in the net mesh (i.e., for a mesh with 1000 edges $w_{\mathrm{iso}} = 20$). Our meshes are always scaled to have an average edge length of 1 and therefore using a different resolution for the same geometry keeps our bending objective the same, but scales the isometric objective by the number of edges.

6.3 Folding constraints

Motivated by the fact that in the smooth case, one cannot move from a folded to a non-folded configuration around a non-planar point, we strive to always satisfy $\phi_{f_j}(x) = 0$ exactly. The common starting point of a flat surface is an interesting case, as it is a bifurcation point between surfaces satisfying Theorem 4.2 and those that do not, which also holds for the discretization Eq. (1). To that end, we solve our problem with an iterative sequential quadratic programming (SQP) solver with a line search, complemented with two simple strategies to handle the constraints $\phi_{f_i}(x)$:

(1) a penalty term [Nocedal and Wright 2006] punishing deviation from the constraints;
(2) a line search method that backtracks if the resulting mesh does not exactly satisfy $\phi_{f_j}(x) = 0, i = 1, ..., n_f$.

Since the functions $\mathrm{sgn}(x), H(x)$ involved in the constraints $\phi_{f_j}(x)$ are not C^1, we replace them by the approximations:

$$\begin{aligned} \mathrm{sgn}(x) &\approx \tanh(hx) \\ H(x) &\approx \begin{cases} 0 & : \text{if } x \le 0, \\ \frac{x^2}{x^2+\delta} & : \text{if } x = 0, \delta > 0 \end{cases} \end{aligned} \tag{15}$$

using the fixed parameters $h = 1000, \delta =$ 1e-5. Our approximation for $H(x)$ is taken from [Li et al. 2012; Poranne et al. 2017]. The use in a homotopy based optimization necessitates an approximation for $H(x)$ that vanishes on a flat mesh, and therefore we do not use the common approximation for the Heaviside function $H(x) \approx \hat{H}(x) = \frac{1+\tanh(hx)}{2}$ because $\hat{H}(0) = \frac{1}{2}$.

We refer to the approximated constraints as $\phi^*_{f_j}(x)$ and replace the optimization problem (10) with the following problem:

$$\begin{aligned} &\arg\min_x f(x) + \omega \sum \|\phi^*_{f_j}(x)\|^2 \\ &\text{subject to} \\ &\phi_{d_i}(x) = 0, \quad i = 1, \ldots, m. \end{aligned} \tag{16}$$

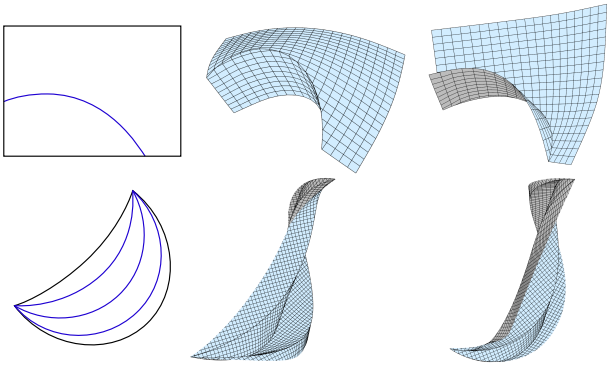


Fig. 13. Using the optional mountain/valley assignment input (Sec. 5.2) on a single crease curve. Each crease pattern is deformed with the same positional constraints, induced by a curve constrained flow, but with a different mountain/valley assignment along one crease, enforced by Eq. (9). In the banana shaped model (bottom row), the rest of the mountain/valley assignments are then uniquely determined.

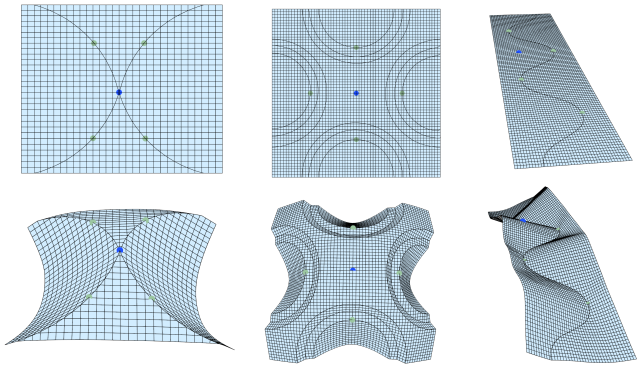


Fig. 14. Using the optional folding angle constraints (Sec. 5.1) on a sparse set of crease points. These examples are deformed by constraining the folding angle of a set of points (in green), without specifying their folding orientation, and by setting a single positional constraint (in blue).

Here, $\omega > 0$ is a metaparameter initialized as $\omega_0 = 1$, which doubles its value if the line search cannot find a point satisfying the supporting plane conditions exactly. In practice, the penalty term only affects points that are very close to being planar, while approaching zero very quickly around already folded points.

6.4 Equality constrained SQP

For ease of notation, we use the following to refer to the objective of Eq. (16):

$$f_\omega(x) = f(x) + \omega \sum \|\phi^*_{f_j}(x)\|^2. \tag{17}$$

We minimize (16) using SQP with a line search [Nocedal and Wright 2006]. Given a set of variables at a given iteration x^k and current values of Lagrange multipliers λ^k, a line search equality constrained SQP algorithm iteratively finds the next direction for a line search of Eq. (16), by which it sets the next variables x^{k+1} by solving a KKT system of the form:

$$\begin{aligned} &K \begin{pmatrix} d^{k+1} \\ \lambda^{k+1} \end{pmatrix} = b, \text{ where} \\ &K = \begin{pmatrix} \Delta^2_{xx}\mathcal{L}(x^k, \lambda^k) & J^{\mathsf{T}}(x^k) \\ J(x^k) & 0 \end{pmatrix}, \quad b = \begin{pmatrix} \nabla f_\omega(x^k) \\ -\phi_{d_i}(x^k) \end{pmatrix}, \end{aligned} \tag{18}$$

where $J(x)$ is the Jacobian of the equality constraints in Eq. (16), $\Delta^2_{xx}\mathcal{L}(x, \lambda) = H_{f_\omega}(x) + \sum \lambda^k_i \Delta^2_{xx}\phi_{d_i}(x)$ is the Hessian of the Lagrangian of the problem and $H_{f_\omega}(x)$ is the Hessian of $f_\omega(x)$.

Following [Rabinovich et al. 2018b], we use a minimally modified Jacobian $J^*(x)$ to deal with singularities in DOGs. We also replace the Hessian of the objective $H_{f_\omega}(x)$ by a convex approximation, which we denote by $H^*_{f_\omega}(x)$, as detailed in Sec. 6.2, and thus replace the system (18) by:

$$\begin{aligned} &K \begin{pmatrix} d^{k+1} \\ \lambda^{k+1} \end{pmatrix} = b, \text{ where} \\ &K = \begin{pmatrix} H^*_{f_\omega}(x^k) + \sum \lambda^k_i \Delta^2_{xx}\phi_{d_i}(x^k) & J^{*^{\mathsf{T}}}(x^k) \\ J^*(x^k) & 0 \end{pmatrix}, \quad b = \begin{pmatrix} \nabla f_\omega(x^k) \\ -\phi_{d_i}(x^k) \end{pmatrix}. \end{aligned} \tag{19}$$

We note that in [Rabinovich et al. 2018b] the authors discretize Laplacian metric flows by solving a similar system with a Laplacian instead of the Lagrangian's Hessian. However, we found that replacing the Laplacian by the Lagrangian followed by convexifying the Hessian performs significantly better, especially on larger models. As common in SQP algorithms, we use a merit function to guide our line search, defined as a combination of the objective and the constraints. The line search chooses step sizes that reduce the objective and keep the DOG angle constraints numerically feasible, while backtracking if a point does not satisfy the constraints ϕ_{f_j} exactly.

This removes the need for the slower LBFGS constraints' projection used by [Rabinovich et al. 2018a,b]. We use the L_2 merit function [Nocedal and Wright 2006]:

$$\psi(x;\mu) = f_\omega(x) + \mu \sum \|\phi_{d_i}(x)\|_2, \tag{20}$$

where we update the parameter μ^k at each iteration using the absolute values of the Lagrange multipliers [Nocedal and Wright 2006]:

$$\mu^k = \max\{c_\mu \cdot \max\{|\lambda_i^k|\},\ \mu_0\}, \tag{21}$$

with $c_\mu = 1.1$ and $\mu_0 = 0.05$.

7 RESULTS

We employ the optimization described above (see Eq. (14)) in an interactive freeform editing system. The input to our system is the flat domain boundary and the curves of the crease pattern, represented by polylines. These can be easily generated from any standard vector graphics format by sampling the smooth curves therein. Our system computes an arrangement of the input curves using CGAL's arrangement model [The CGAL Project 2019; Wein et al. 2019; Zukerman et al. 2019] and solves the symmetric indefinite linear systems as required by the optimization (Eq. (19)) using Pardiso [De Coninck et al. 2016; Kourounis et al. 2018; Verbosio et al. 2017]. We ran our experiments on a 16-core Ryzen Threadripper 1950X clocked at 3.4 GHz. Our editing system supports setting point handle positional constraints, as can be seen in Figures 1, 8, 15, and the second and last model from the left in Fig. 3. A stress test for our algorithm is shown in Fig. 15, with a crease pattern containing 20 different creases that all automatically bend and fold, driven only by point handle positional constraints. We also support constraining a crease curve by specifying its curvature and torsion in a curve constrained flow, see Figures 3 and 13, as well as prescribing a sparse set of dihedral angles along crease points, see Fig. 14.

Fig. 13 is the only case where we supply a mountain/valley assignment as input, while Fig. 14 displays the only models designed by constraining dihedral angles. The curve constrained positional constraints, as well as the dihedral angles, are interpolated for improved quality (see Fig. 6). To maintain interactive frame rates in handle based editing tasks, we run a fixed number of SQP iterations per frame, which we set to 5. On models with 500, 1000, 2000, 4000 vertices these 5 SQP iterations run on average at more than 39, 19, 9, 4 frames per second. Hence our system is able to handle realtime interaction of meshes with around 2000 vertices. The two concentric circles folds in Fig. 1 originate from a mesh with about 5500 vertices. They are designed by simply penalizing the distance of a single pair of vertices while interpolating the penalty weight, and their final forms are reached in about 30 seconds. We refer the reader to our supplementary video for further results, including interactive editing examples.

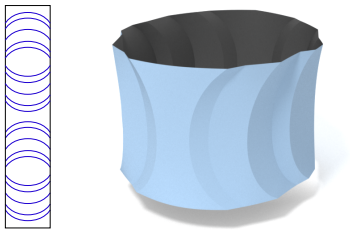


Fig. 15. A curved folded ring. The crease pattern is taken from [Mitani 2019] and contains 20 different creases. It was deformed by enforcing our folding constraints (Sec. 6), without specifying any folding angles or mountain/valley assignments, together with positional constraints pushing the vertices on the right and left boundary to match such that a loop is formed, while also minimizing an additional bending energy term for the glued area to smooth it.

8 CONCLUSIONS AND FUTURE WORK

This paper is a first step towards unhindered freeform modeling of curved folded surfaces. Basing our models on DOGs [Rabinovich et al. 2018a] allows us to capture the full set of curved folded deformations, and our discretization in Sec. 4, together with the folding algorithm in Sec. 6, allows us to steer the modeled deformations towards those that simultaneously fold and bend crease curves. Our deformation algorithm is able to model bending and folding of complicated crease patterns by merely using positional constraints, making it highly suited for exploration of new curved folded surfaces. We supply further optional objectives to constrain dihedral angles and mountain/valley assignments in Sec. 5, providing designers additional expressiveness.

Similar to other works on modeling DOGs [2018a; 2018b], the most obvious limitation of our algorithm is speed. Our optimization framework allows us to interactively model up to 2000 vertices. We leave scaling of the optimization to future work, possibly by using a multigrid solver on the DOG grids. In addition, we find that we lack tools and objectives to enforce symmetry of the designed shapes. In particular, we would like to look at folding of curved symmetric plane wallpapers and tessellations [Demaine et al. 2015; Mundilova 2019]. We also do not take physical reality constraints into account, such as collisions, material thickness and elasticity properties, making the models created in our system not necessarily realizable. Incorporating our model into a physically accurate design system could potentially alleviate this limitation. Finally, we note that we model deformations of a given fixed input crease pattern. Optimizing and changing an input crease pattern, as done in origami modeling tools [Tachi 2010], could offer new and exciting ways to discover and design curved folded surfaces.

ACKNOWLEDGMENTS

The authors would like to thank Oliver Glauser and Justin Solomon for illuminating discussions, Martin Kilian and Niloy Mitra for clarifications about their prior work, and Katja Wolff and Phillip Herholz for their help with results, figures and video production. The work

was supported in part by the Deutsche Forschungsgemeinschaft-Collaborative Research Center, TRR 109, "Discretization in Geometry and Dynamics" and gifts from Facebook, Adobe and Snap, Inc.

REFERENCES

E. D. Adler. 2004. *" A New Unity!" The Art and Pedagogy of Josef Albers.* Ph.D. Dissertation.

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

M. Bergou, M. Wardetzky, D. Harmon, D. Zorin, and E. Grinspun. 2006. A quadratic bending model for inextensible surfaces. In *Symposium on Geometry Processing*. 227–230.

P. Bo, Y. Zheng, X. Jia, and C. Zhang. 2019. Multi-strip smooth developable surfaces from sparse design curves. *Computer-Aided Design* 114 (2019).

M. Botsch, M. Pauly, M. H. Gross, and L. Kobbelt. 2006. PriMo: coupled prisms for intuitive surface modeling. In *Symposium on Geometry Processing*. 11–20.

R. Burgoon, Z. J. Wood, and E. Grinspun. 2006. Discrete Shells Origami. In *Computers and Their Applications*.

H. U. Buri, I. Stotz, and Y. Weinand. 2011. Curved Folded Plate Timber Structures. In *IABSE-IASS 2011 London Symposium*. IABSE-IASS.

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

A. De Coninck, B. De Baets, D. Kourounis, F. Verbosio, O. Schenk, S. Maenhout, and J. Fostier. 2016. Needles: Toward Large-Scale Genomic Prediction with Marker-by-Environment Interaction. *Genetics* 203, 1 (2016), 543–555. https://doi.org/10.1534/genetics.115.179887 arXiv:http://www.genetics.org/content/203/1/543.full.pdf

E. D. Demaine, M. L. Demaine, V. Hart, G. N. Price, and T. Tachi. 2011b. (Non) existence of pleated folds: how paper folds between creases. *Graphs and Combinatorics* 27, 3 (2011), 377–397.

E. D. Demaine, M. L. Demaine, D. A. Huffman, D. Koschitz, and T. Tachi. 2015. Characterization of curved creases and rulings: Design and analysis of lens tessellations. *Origami* 6 (2015), 209–230.

E. D. Demaine, M. L. Demaine, D. A. Huffman, D. Koschitz, and T. Tachi. 2018. Conic Crease Patterns with Reflecting Rule Lines. arXiv preprint arXiv:1812.01167. (2018).

E. D. Demaine, M. L. Demaine, and D. Koschitz. 2011a. Reconstructing David Huffman's legacy in curved-crease folding. *Origami* 5 (2011), 39–52.

E. D. Demaine, M. L. Demaine, D. Koschitz, and T. Tachi. 2011c. Curved crease folding: a review on art, design and mathematics. In *Proceedings of the IABSE-IASS Symposium: Taller, Longer, Lighter*. 20–23.

E. D. Demaine and J. O'Rourke. 2007. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra.* Cambridge University Press.

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

J. P. Duncan and J. Duncan. 1982. Folded developables. *Proc. R. Soc. Lond. A* 383, 1784 (1982), 191–205.

W. H. Frey. 2002. Boundary triangulations approximating developable surfaces that interpolate a closed space curve. *International Journal of Foundations of Computer Science* 13, 02 (2002), 285–302.

W. H. Frey. 2004. Modeling buckled developable surfaces by triangulation. *Computer Aided Design* 36, 4 (2004), 299–313.

S. Fröhlich and M. Botsch. 2011. Example-Driven Deformations Based on Discrete Shells. *Comput. Graph. Forum* 30, 8 (2011), 2246–2257.

D. Fuchs and S. Tabachnikov. 1999. More on paperfolding. *The American Mathematical Monthly* 106, 1 (1999), 27–35.

D. Fuchs and S. Tabachnikov. 2007. *Mathematical omnibus: thirty lectures on classic mathematics.* Springer.

R. Goldenthal, D. Harmon, R. Fattal, M. Bercovier, and E. Grinspun. 2007. Efficient simulation of inextensible cloth. *ACM Trans. Graph.* 26, 3 (2007), 49.

F. Gramazio and M. Kohler. 2014. *Made by robots: challenging architecture at a larger scale.* John Wiley & Sons.

E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. 2003. Discrete Shells. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 62–67. http://dl.acm.org/citation.cfm?id=846276.846284

B. Grünbaum. 1972. *Arrangements and spreads.* American Mathematical Society.

D. A. Huffman. 1976. Curvature and creases: a primer on paper. *IEEE Trans. Computers* 25, 10 (1976), 1010–1019.

M. Kilian, S. Flöry, Z. Chen, N. J. Mitra, A. Sheffer, and H. Pottmann. 2008. Curved folding. *ACM Trans. Graph.* 27, 3 (2008), 75:1–75:9.

M. Kilian, A. Monszpart, and N. J. Mitra. 2017. String actuated curved folded surfaces. *ACM Trans. Graph.* 36, 3 (2017), 25:1–25:13.

D. Kourounis, A. Fuchs, and O. Schenk. 2018. Towards the Next Generation of Multiperiod Optimal Power Flow Solvers. *IEEE Transactions on Power Systems* PP, 99 (2018), 1–10. https://doi.org/10.1109/TPWRS.2017.2789187

Z. Li, S. Wang, and X. Lin. 2012. Variable selection and estimation in generalized linear models with the seamless L_0 penalty. *Canadian Journal of Statistics* 40, 4 (2012), 745–769.

Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang. 2006. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.* 25, 3 (2006).

J. Mitani. 2009. A design method for 3D origami based on rotational sweep. *Computer-Aided Design and Applications* 6, 1 (2009), 69–79.

J. Mitani. 2012. Column-shaped origami design based on mirror reflections. *Journal for Geometry and Graphics* 16, 2 (2012), 185–194.

J. Mitani. 2019. *Curved-Folding Origami Design.* CRC Press.

J. Mitani and T. Igarashi. 2011. Interactive design of planar curved folding by reflection. In *Proc. Pacific Graphics, Short Papers*.

K. Mundilova. 2019. On mathematical folding of curved crease origami: Sliding developables and parametrizations of folds into cylinders and cones. *Computer-Aided Design* (2019). accepted for publication.

R. Narain, T. Pfaff, and J. F. O'Brien. 2013. Folding and Crumpling Adaptive Sheets. *ACM Trans. Graph.* 32, 4 (2013).

R. Narain, A. Samii, and J. F. O'Brien. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM transactions on graphics (TOG)* 31, 6 (2012), 152.

J. Nocedal and S. J. Wright. 2006. *Sequential quadratic programming.* Springer.

R. Poranne, M. Tarini, S. Huber, D. Panozzo, and O. Sorkine-Hornung. 2017. Autocuts: Simultaneous Distortion and Cut Optimization for UV Mapping. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)* 36, 6 (2017).

H. Pottmann, M. Eigensatz, A. Vaxman, and J. Wallner. 2015. Architectural geometry. *Computers & graphics* 47 (2015), 145–164.

H. Pottmann and J. Wallner. 1999. Approximation algorithms for developable surfaces. *Comput. Aided Geom. Des.* 16, 6 (1999), 539 – 556. https://doi.org/10.1016/S0167-8396(99)00012-6

H. Pottmann and J. Wallner. 2001. *Computational line geometry.* Springer, Berlin, Heidelberg, New York.

M. Rabinovich, T. Hoffmann, and O. Sorkine-Hornung. 2018a. Discrete Geodesic Nets for Modeling Developable Surfaces. *ACM Trans. Graph.* 37, 2 (2018), 16.

M. Rabinovich, T. Hoffmann, and O. Sorkine-Hornung. 2018b. The shape space of discrete orthogonal geodesic nets. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 228.

R. D. Resch. 1974. Portfolio of shaded computer images. *Proc. IEEE* 62, 4 (1974), 496–502.

K. Rose, A. Sheffer, J. Wither, M.-P. Cani, and B. Thibert. 2007. Developable Surfaces from Arbitrary Sketched Boundaries. In *Proc. Symposium on Geometry Processing*. 163–172. http://dl.acm.org/citation.cfm?id=1281991.1282014

C. Schreck, D. Rohmer, and S. Hahmann. 2017. Interactive paper tearing. *Computer Graphics Forum (Eurographics)* 36, 2 (2017), 95–106.

C. Schreck, D. Rohmer, S. Hahmann, M.-P. Cani, S. Jin, C. C. L. Wang, and J.-F. Bloch. 2015. Nonsmooth Developable Geometry for Interactively Animating Paper Crumpling. *ACM Trans. Graph.* 35, 1 (2015), 10. http://dblp.uni-trier.de/db/journals/tog/tog35.html#SchreckRHCJWB15

J. Solomon, E. Vouga, M. Wardetzky, and E. Grinspun. 2012. Flexible developable surfaces. *Comput. Graph. Forum* 31, 5 (2012), 1567–1576.

M. Spivak. 1999. *A Comprehensive introduction to differential geometry, 1.* Publish or Perish, Houston, TX. http://opac.inria.fr/record=b1117511

O. Stein, E. Grinspun, and K. Crane. 2018. Developability of triangle meshes. *ACM Trans. Graph.* 37, 4 (2018).

T. Tachi. 2010. Freeform variations of origami. *J. Geom. Graph* 14, 2 (2010), 203–215.

T. Tachi. 2011. One-DOF rigid foldable structures from space curves. In *Proceedings of the IABSE-IASS Symposium*. 20–23.

T. Tachi. 2013. Composite rigid-foldable curved origami structure. In *1st International Conference on Transformable Architecture (Transformables 2013), Seville, Spain, Sept.* 18–20.

C. Tang, P. Bo, J. Wallner, and H. Pottmann. 2016. Interactive design of developable surfaces. *ACM Trans. Graph.* 35, 2, Article 12 (2016), 12 pages.

The CGAL Project. 2019. *CGAL User and Reference Manual* (4.14 ed.). CGAL Editorial Board. https://doc.cgal.org/4.14/Manual/packages.html

F. Verbosio, A. D. Coninck, D. Kourounis, and O. Schenk. 2017. Enhancing the scalability of selected inversion factorization algorithms in genomic prediction. *Journal of Computational Science* 22, Supplement C (2017), 99 – 108. https://doi.org/10.1016/j.jocs.2017.08.013

R. Wein, E. Berberich, E. Fogel, D. Halperin, M. Hemmer, O. Salzman, and B. Zukerman. 2019. 2D Arrangements. In *CGAL User and Reference Manual* (4.14 ed.). CGAL Editorial Board. https://doc.cgal.org/4.14/Manual/packages.html#PkgArrangementOnSurface2

M. Wertheim. 2004. Cones, curves, shells, towers: He made paper jump to life. *The New York Times* 2, 4 (2004), 5.

B. Zukerman, R. Wein, and E. Fogel. 2019. 2D Intersection of Curves. In *CGAL User and Reference Manual* (4.14 ed.). CGAL Editorial Board. https://doc.cgal.org/4.14/Manual/packages.html#PkgSurfaceSweep2