Title:
Fitting Single Crease Curved-Fold Model to the User Specified Points

Authors:
Yuka Watanabe, ywatanabe@cgg.cs.tsukuba.ac.jp, University of Tsukuba
Jun Mitani, mitani@cs.tsukuba.ac.jp, University of Tsukuba

Keywords:
Curved Folding, Developable Surface Modeling, Optimization

DOI: 10.14733/cadconfP.2021.144-148

Introduction:
We propose an approach to realize an intuitive manipulation of a developable surface with a single curved fold. The surface shape is discretized as planar quad strips, as shown in Fig. 1 and Fig. 2. The user manipulates the shape by specifying the destinations of some points placed on the surface. The shape is determined by the optimization method to approximate the destinations while maintaining the developability.

There has been extensive research on the design of developable surfaces due to their industrial usefulness. It has an ideal property in manufacturing that its shape is formed from a flat sheet of material only by bending, without stretching nor contraction. Curved folding is also made from a flat sheet by folding it along the curved crease. However, the control of the shape with folds is a difficult problem, and much work remains to be done. This is because the direction of the surface bending may change in many ways while being folded. This makes the rulings, which are the straight lines on a developable surface, to transit, as shown in Fig. 1 middle and bottom row. Other than the paper folding, there are projects to curved-fold a metal plate using machines, such as RoboFolds [1]. For the automation, it is essential to understand, model, and to predict its folding motion.

To support such work, we had developed GUI systems, shown in Fig. 2, to model and visualize the shape and the folding motion of the curved folding [7],[8]. The user can edit the shape interactively by changing the parameters on the curve control points. The parameters are the curvature, the torsion, and the folding angle, which defines the shape of the curve, the ruling directions, and the shapes of the adjacent curved surfaces. While editing, the user checks the calculated 3D shape to make sure it is reshaped as intended. The user also needs to check that the rulings are not intersecting in the 2D space, which are numerically possible but never occurs in the real world. This interactive manipulation is not intuitive since the user cannot change the 3D shape directly, requiring some experience.

In our new system, we developed more intuitive and goal-oriented user interface which fits the surface to some target positions. Given the curved-fold model with a 2D curved crease, the user controls the target 3D shape by specifying some surface control points on the surface and the corresponding 3D target points, as shown in Fig. 3. The positions of the control points and the target points are defined according the application. In direct shape editing the user sets the positions freely through the GUI. For combining multiple curved fold models, the control and the target points are set on the joint parts. Then the system reshapes the curved-folded surface by optimizing the torsion and the folding angle to minimize the distances between the surface control points and the target points, while the 2D curved crease on the paper is fixed. For efficiency, our optimization method is performed in two steps. First, the rulings are randomly changed by small amount and checked if there are any rulings intersecting. Then, only if there is no intersection, torsions and a set of folding angles are calculated from the ruling directions. Finally, after some iterations, the parameters with the smallest distances are adopted.

As related work, there are software to simulate the folding motions of origami by intuitive user manipulation, such as Free-form Origami by Tachi [6] and Origami Simulator by Amanda Ghassaei [3]. Their model assumes fixed geometry of the polygon faces and therefore not suitable for the folding motion of the curved folding with the rulings transition. In the area of curved folding, Kilian's method models the folding motion by re-meshing the triangular mesh depending on the local surface curvatures [4]. Their methods are able to represent various shapes, but resulting mesh data may contain thousands of faces, not suitable for observing the rulings. Rabinovich et al. represented curved-folded surfaces by a set of discrete orthogonal geodesic nets connected along the curved creases by the positional constraints [5]. Their system allows the users to deform the shape interactively and intuitively, taking different approach from ours in the modeling and the optimization method.

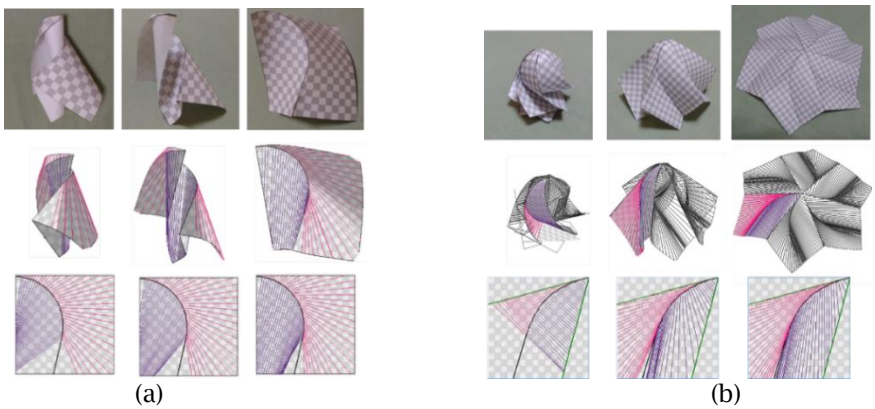

Fig. 1: Curved fold models and the folding motions generated by our previous systems: (a) Curved folding with single crease [7], and (b) Rotational symmetric curved folding [8]. Top: photo of real paper. Middle: 3D model. Bottom: crease and rulings projected on the 2D space of flattened paper.

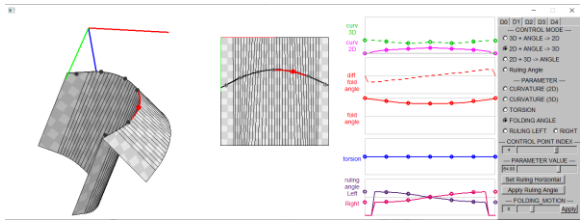

Fig. 2: GUI system from our previous works, with four panes showing 3D model, 2D space of unfolded paper with a crease and rulings, curve charts of crease curve parameters, and the control panel to edit the parameters. The curve control points are shown as small dots on 2D and 3D crease curves.

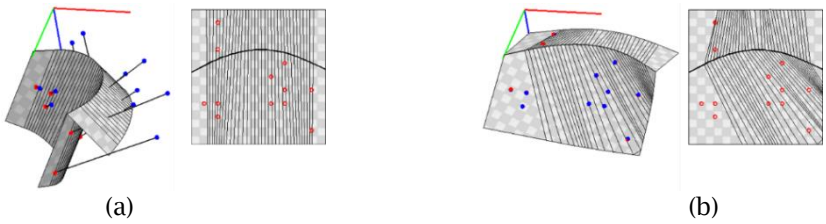

Fig. 3: New user interface on 2D and 3D pane. Surface control points are shown as red dots. The target points are shown as blue dots connected to the corresponding surface control points in 3D pane. (a) Initial state. (b) Result of optimization.

Main Idea:
Our method optimizes the torsion and the folding angle on the crease curve to minimize the displacement of the original shape to the target. The cost function is calculated as the average distance between the pairs of the surface control points and their corresponding target points. The distance between the two points, rather than the distance to the surface, is used to avoid the tangential drift. A simple discrete gradient descent method would be inefficient as its parameter space has many local minima and invalid states. With seven curve control points, it optimizes 14 parameters: the torsions and the folding angles on each control point. Small changes of the parameters cause large differences in the rulings and the 3D shape, easily generating ruling intersections in the 2D space. Instead of this direct optimization, we introduce a ruling based optimization method which searches for the ruling directions that best fits the targets. In every optimization step, the method first tries some change in the ruling angles and ensures the ruling to be nonintersecting. Then the torsions and a set of folding angles are calculated from the ruling angles. Finally, the rulings and the folding angles which generates the best fitting 3D model are adopted. Our method has 15 parameters: the ruling angles on the left and the right side of the seven curve control points and the folding angle with 1-DoF. Although having larger number of parameters, the optimization process is more stable as the output shape is less sensitive to the parameters. Also, because the rulings are changed by small amount sequentially from a valid state in the previous iteration, there are smaller probabilities of ruling intersections compared to setting new random parameters in every iteration. The geometry and the procedure are described in the following sections. Our new method enables the applications such as (1) editing the shape of a curved folding so that the user specified points on the surface fit the target 3D positions and (2) stitching two surface patches by deforming them to share a boundary curve.

Geometry of Curved Folding
Our system uses the following equations introduced by Fuchs and Tabachnikov [2] to calculate the ruling angles on the left and the right side of the crease curve, $\beta_L(s)$, $\beta_R(s)$, from the 2D curvature $k_{2D}(s)$, torsion $\tau(s)$, and the folding angle $\alpha(s)$, with the parameters represented as arc-length parametric function,

$$\cot\beta_L(s) = \frac{\alpha(s)' - \tau(s)}{k_{2D}(s)\tan\alpha(s)}, \tag{1}$$

$$\cot\beta_R(s) = \frac{-\alpha(s)' - \tau(s)}{k_{2D}(s)\tan\alpha(s)}, \tag{2}$$

as illustrated in Fig. 4. To derive the torsion and the folding angle from the ruling angles, used in our optimization process, Eqn. (1) and (2) are deformed as

$$\tau(s)/\tan\alpha(s) = \frac{1}{2}\,k_{2D}(s)\big(\cot\beta_L(s) + \cot\beta_R(s)\big), \tag{3}$$

$$\alpha(s)'/\tan\alpha(s) = \frac{1}{2}\,k_{2D}(s)\big(\cot\beta_L(s) - \cot\beta_R(s)\big). \tag{4}$$

By expressing the parameters in the discrete form and setting a folding angle α_0 on the starting point of the curve, the torsions and the folding angles on the rest of the points are calculated sequentially by

$$\alpha_i = \alpha_{i-1} + d\alpha_{i-1}dx, \tag{5}$$

$$\tau_i = \frac{1}{2}\,k_{2D\,i}(\cot\beta_{L\,i} + \cot\beta_{R\,i})\tan\alpha_i, \tag{6}$$

$$d\alpha_i = \frac{1}{2}\,k_{2D\,i}(\cot\beta_{L\,i} - \cot\beta_{R\,i})\tan\alpha_i. \tag{7}$$

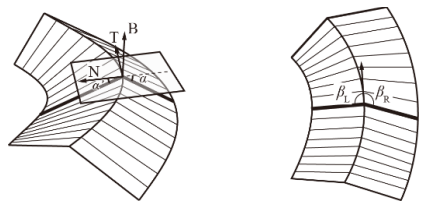

Fig. 4: Parameters of a crease curve.

Optimization Process

As an initial state, a curved-folded surface with one crease is given. Then the parameters are optimized by the following process:

1. Change the ruling angles at some curve control points randomly by small amounts. Interpolate them with spline curves of degree three to obtain the ruling angles on all points on the curve. The interpolation is processed by the same method as torsion and curvature in our previous works.
2. Check if there are any ruling intersections in the 2D space. If there are, go back to step 1.
3. Calculate the torsions and a set of folding angles using Eqn. (5)-(7). Since there are many solutions to the folding angles according to α_0 on the starting point, calculate all possible solutions in the discrete sampling and discard the result which the folding angle range out of $[-\pi/2, \pi/2]$.
4. Calculate the cost function. The folding angle with the smallest cost is adopted.
5. Check if the cost is smaller than the previous iteration, which is derived from the previous ruling angles. If it is, update the ruling angles. If it is not, go back to step1.
6. Check if the cost is smaller than the predefined threshold. If it is not, go back to step 1.

The iteration is repeated until the cost is below the threshold, 0.3mm, or the number of iterations exceeds the predefined maximum number, 500 in our implementation.

Fitting Single Crease Model

We have implemented the new feature on the GUI system of [7] for editing a curved folding with a single crease. A result is shown in Fig. 3. The coordinates of the surface control points and the target points are given from an input file or by mouse input. The input file includes a list of 2D coordinates of the surface control points and the corresponding 3D target positions. As the mouse input, the surface control points are first added, moved, or deleted in 2D pane to specify the position on the surface, and then the 3D target points are dragged in the 3D space in the 3D pane. The optimization is performed by pressing a button on the control panel.

Stitch the Boundaries of Two Surface Patches

The feature is also implemented on the GUI system of [8], which is for designing and visualizing the rotational symmetric curved folding, to stitch the adjacent surface patches. Fig. 5 shows a result of the stitching in one of the frames in the folding motion of the origami model shown in Fig. 1(b). As the initial state, the six patches, each having one curved crease, are placed in rotational symmetry, rotated 60 degrees from the adjacent patch (Fig. 5(a)). To minimize the gap between the patches, their locations and orientations are first optimized without changing their shape, keeping the rotational symmetry (Fig. 5(b)). Then the surface control points are placed evenly on the boundaries. The target points are set to be the midpoints of the pairs of surface control points on the adjacent patch boundaries. Finally, the torsion and the folding angle are optimized so that the two corresponding boundaries form the same shape, approximating the series of the midpoints (Fig. 5(c)). This process had been performed manually in the previous work, but the new feature enabled it to be semi-automated.

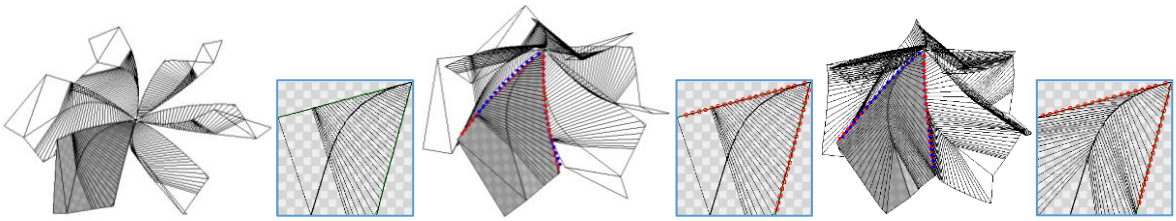

Fig. 5: Stitching the boundaries of surface patches: (a) Initial state of the six patches, (b) After alignment, (c) After the optimization of torsion and folding angle. Left: the 3D model. Right: 2D crease and rulings of one patch.

Evaluation

We evaluated our method using the synthetic data. The curved-fold model of 200mm square sheet is deformed by random parameters, such as torsions, folding angles, and rulings, to make various 3D shapes. Then the pairs of surface control point and the target point are generated from the 3D shapes. Fig. 6 shows some results of the optimization process given the initial state shown in Fig. 2 and the target points generated as the synthetic data. For 14 random target shapes and 10 surface control points alignments, the average distance between the surface control points and the target points are 1.14 mm in average and 5.03 mm at maximum, while a simple discrete gradient descent method result in 7.13 mm in average and 24.9 mm at maximum. The average process time was 4.39 sec, using Core i7-1165G7 @ 2.80GHz and 16 GB RAM. This is a practical process time for an interactive system, significantly faster than manually editing the parameters.

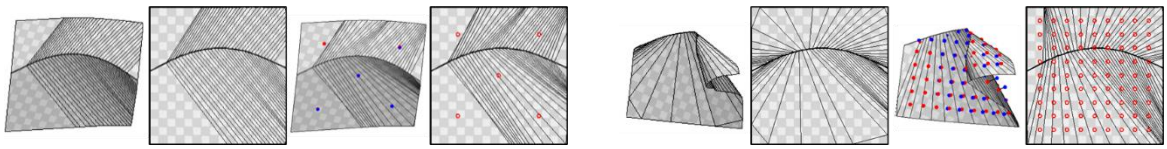

Fig. 6: Optimization results. From left to right, target shape, target rulings, result shape, result rulings.

Conclusions:

We developed an intuitive manipulation method of a curved-folded shape with a given 2D crease by optimizing the curve parameters to minimize the displacement of the surface control points to the target points. To avoid the inefficiency caused by the invalid states with rulings intersections, our method takes two steps: trials of the rulings without the intersections, and the derivation of the torsions and the folding angles that best approximates the target points. The method is applied to the manipulation of the curved folding and the stitching of two curved-folded surface patches. The evaluation shows that our method well approximates the target shapes with average gap of 1.14 mm in a practical process time, with better accuracy than a simple discrete gradient descent method.

Acknowledgements:

This work was supported by JSPS KAKENHI Grant Number JP19K12677, Japan.

References:

[1] Epps, G.: Robofold and robots. IO, Made by Robots: Challenging architecture at a larger scale, Architectural Design, 2014, 68–69. https://doi.org/10.1002/ad.1757
[2] Fuchs, D. Tabachnikov, S.: More on paperfolding, The American Mathematical Monthly, 106(1), 1999, 27-35. https://doi.org/10.2307/2589583
[3] Ghassaei, A.; Demaine E. D.; Gershenfeld, N.: Fast, interactive origami simulation using GPU computation, Origami 7: The 7th International Meeting on Origami in Science, Mathematics and Education, 2018, 1151-1166.
[4] Kilian, M.; Monszpart, M.; Mitra, N. J.: String actuated curved folded surfaces, ACM Transactions on Graphics, 36(3), 2017, 25:1-13. https://doi.org/10.1145/3072959.3015460
[5] Rabinovich, M.; Hoffmann, M.; Sorkine, O.: Modeling Curved Folding with Freeform Deformations, ACM Transactions on Graphics, 38(6), 2019. https://doi.org/10.1145/3355089.3356531
[6] Tachi, T.: Generalization of Rigid-Foldable Quadrilateral-Mesh Origami, Journal of the International Association for Shell and Spatial Structures (IASS), 50(3), 2009, 173-179.
[7] Watanabe, Y.; Mitani, J.: Modelling the Folding Motions of a Curved Fold, Origami 7: The 7th International Meeting on Origami in Science, Mathematics and Education, 2018, 1135-1150.
[8] Watanabe, Y.; Mitani, J.: Visualization of Folding Motion of Rotationally Symmetric Curved Folding, Computer-Aided Design & Applications, 17(3), 2019, 513-524. https://doi.org/10.14733/cadaps.2020.513-524