IASS 2020/21
SURREY 7
International Conference on
Spatial Structures 2020/21

Proceedings of the IASS Annual Symposium 2020/21 and the 7th International Conference on Spatial Structures
Inspiring the Next Generation
23 – 27 August 2021, Guilford, UK
S.A. Behnejad, G.A.R. Parke and O.A. Samavati (eds.)

How to use parametric curved folding design methods- a case study and comparison

Riccardo FOSCHI*, Robby KRAFT[a], Rupert MALECZEK[a],
Klara MUNDILOVA[b], Tomohiro TACHI[c]

*Department of Architecture | University of Bologna, ITA
riccardo.foschi2@unibo.it

[a] i.sd | Structure and Design | Department of Design, University of Innsbruck, AUT
[b] CSAIL | MIT | Massachusetts, USA
[c] Graduate School of Arts and Sciences | The University of Tokyo, JPN

Abstract

Designs based on developable surfaces can be convenient for many reasons, however designing developable patterns that make use of curved creases is a challenge. Many studies propose new methods to tackle the problem but sometimes these methods do not generate a parametric model which is easily modifiable by changing the input parameters. Furthermore, the known methods are applicable only to certain families of curved folded models, because there is no generalized method for curved folding yet. Thus, sometimes, it is hard for designers to decide which method is more suitable for their needs. This paper shows how to use different well-known and newer approaches to produce parametric curved folded designs. The potentialities and criticalities of three approaches are compared by applying them to the same case study, namely the "curved folded tripod". The aim, thus, is to make the design of curved folded geometries more accessible to designers without a background in origami theory.

Keywords: curved folding, applied origami, computational design, developability enforcement.

1. Introduction

Designs based on curved folding, i.e., developable surfaces with curved creases, can be convenient for structural stiffness, fabrication, and durability. The curved surfaces formed by active bending can structurally act as a shell system. Folding from a flat panel simplifies production, reduces the need for molds, reduces offcut, and decreases assemble time and thus the cost of the product.

For actual design of curved folded shapes, designers often adopt an approach based on trial-and-error (inspired by origami artists) starting from the manipulation of a piece of paper (or any other flexible material). This approach is convenient because there is no need to understand the actual geometric properties of the folded surface, such as ruling direction, developability, or curvature. The designer can simply manipulate the piece of paper, and the material itself guarantees the developability by rearranging the ruling and the curvature dynamically by simply following its natural behaviour. However, after this first trial-and-error phase, according to the aim of the project, it is often useful to digitize the model by a sculpting and post-rationalizing approach (Koschitz [8]).

Model digitization allows designers to check the accuracy and integrate the model into other projects already designed exclusively within the digital space while verifying the tolerances. Because a paper model can have errors generated by the elasticity of the material, it is particularly important to create a mathematically correct digital model for engineering applications such as a foldable panel for space or architectural applications. The digital model should be editable and thus parametric, so that the model would be easily manipulated to fit the required design constraints and needs. Thus, computational parametric strategies for curved creases need to be adopted.

Many researchers proposed new methods for computational curved folding (Demaine *et al.* [4], Tachi [14]), however methods that can take versatile input often do not provide parametric results (Kilian *et al.* [7], Ghassaei [6]). Other studies provide parametric design methods for curved folding, each of which, however, is restricted to certain families of curved folds, or is restricted to discretized surfaces (Mitani [10]). In other words, there is no "general tool for curved folding" yet. Because of that, it is not obvious how to apply these methods when we start from a physical model (or a digital simulation model) and convert it to a parametric model. In design practice, a designer needs to choose the proper tool depending on the family of patterns and the design needs; however, the process of identifying which method to use for each case is not trivial.

In this paper, we show how to use three computational methods for creating parametric models of curved folding through a case study. We highlight their potentialities and limitations to provide designers examples and help them choose the most suitable approach for their needs. As a case study, we chose a specific curved folded model that we call "tripod" as introduced in Section 2. Some researchers previously referred to similar structures as "three-crease mesh" or "triangle-shaped curved-line folding component" (Bhooshan [1], Brancart *et al.* [2]). The first step of the design is to start from a folded sheet of paper and extract the geometric characteristics from the model by observation and application of geometric knowledge (Section 3). This leads to the estimation of the families of surfaces and creases and the geometric constraints needed, which are then parametrically modelled by three methods explained in Section 4, namely methods (a) isometry enforcement based on discretized mesh, (b) constructive method with various geometric constraints, and (c) mirror reflection of developable surfaces (Section 4). Then, we compare the methods and show the limitation and trade-off between accuracy and versatility (Section 5).

2. Problem description

2.1 Curved folding characteristics guessed/identified from an instance.

To create a parametric model from an instance of curved folding either obtained as a physical paper model or a digitally simulated model, we need to identify the underlying principle of the folded model based on the geometry of curved folding, i.e., how developable surfaces and curved creases work.

The identification of the characteristics of the curved creased surfaces could be carried out by observing the mock-up and rationalize it based on its shape and folding behaviour. We divide the model into surfaces that are separated by creases. Each surface can then be further divided into patches of flat planes, cylinders, cones, and tangent developable surfaces. Each surface patch is characterized by ruling directions which can be estimated through observation and corrected using geometric knowledge. Then we characterize each crease in the folded state by roughly estimating its curvature and torsion. In particular, it is important to locate the inflection points, the points where the sign of curvature changes. We can also look for points that can be used to split the crease into simpler sub-curves (e.g., arcs, segments, ellipses, parabolas). Lastly, we determine if the curved crease lies in a plane, implying it has zero torsion.

After identifying the geometric characteristics of the surfaces and creases, each surface patch and crease is modelled such that they satisfy geometric constraints, such as developability. Also, the smoothness of the surface (e.g., G0 or G1 continuity) needs to be considered. In addition to origami constraints, there are outside influences and design requirements. Each method has its own process to handle these constraints.

2.2 Case study

We outline three approaches to digitize a curved crease design, which we use to model and parametrize the "tripod" (Bhooshan [1], Brancart *et al.* [2]). The tripod is a developable model with three curved creases forming a triangular shape enclosing one central face which we call the top surface. As the creases are folded, the top surface bends. If we divide the top surface into patches according to curvature, we recognize a flat surface in the middle, connected to three cylindrical surfaces (refer to Figure 1). The surfaces adjacent to the top surface are also bent and if we divide them according to curvature and rule lines, we can have planar, conical, cylindrical, or tangent developable patches. The surface patches will be tangent continuous (G1).

We can add extra properties to tripod design. For example, we can extend the side surfaces to let the boundary fit to a plane (Figure 1 Middle). Then, we can also constrain the boundary edges to be straight lines, so that it fits to a triangle (Figure 1 Right). We will construct tripods with straight boundary edges whenever possible, but one of the methods does not allow for this additional constraint.

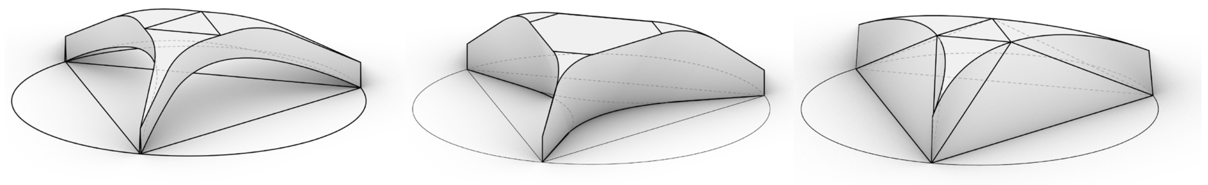

Figure 1: Tripod variations.

3. Estimating types of surfaces and creases by observation

3.1. Estimating surfaces

To digitize a curved folded model, it is necessary to observe and understand the geometry of the model. Developable surfaces are ruled surfaces (but not all ruled surfaces are developable) and one of the most important characteristics to observe is the direction and distribution of the rule lines.

A sheet of paper in the real world has a nonzero amount of material elasticity making it possible that a physical model might manifest non developable surfaces such as doubly curved or warped surfaces (Demaine *et al.* [3]). In the parametric model, designers often want to avoid this imperfection, especially if the aim is to fabricate the design with metal, wood, or any other less elastic material. Thus, it is important to mathematically interpret the paper model before digitizing. It is useful to understand in advance if a perfectly developable digital twin of a physical paper model is feasible at all.

One possible method to estimate the rule lines is making note of the straight lines at the contour while rotating the model. Another method is lighting the model with a light that generates sharp shades (small light sources, such as a flashlight, or very far sources, such as the sun). The shade line on a single curved surface always runs along the rulings; curved shade lines suggest that the model is not developable. An alternative method is to move a ruler along the surface looking for positions where there are no gaps between the surface and the ruler; the rule line lies along the ruler. In addition, one can build a physical model with a glossy metallic surface and observe the distortion of objects reflected on its surfaces. Objects will be undeformed on a planar patch, stretched in one direction on a cylindrical patch, or bent if the surface is conical. The ruling direction is perpendicular to the stretching direction.

When we know roughly where the rule lines are placed, we can estimate how to subdivide the model into multiple surface patches according to the main four types of developable surfaces: planes, cylinders, cones, and tangent developables. The division into patches is estimated through observation which can only be verified by using geometric background knowledge. For example, the existence of a planar patch on the top surface is not necessarily self-evident from observation only, but geometric knowledge that rulings cannot cross each other (Demaine *et al.* [4]) suggests the existence of planar patch. Kilian *et al* [7] propose a method to retrieve the surface patch decompositions from a 3D scan of a mock-up model.

We use these observed developed surface types as starting points to build parametric algorithms and to set up necessary input geometry and constraints. For example, if we identify conical surfaces, we can constrain rulings to pass through apices, or enforce parallel rule lines in the case of cylindrical surfaces.

3.2. Estimating curved creases

It is helpful to classify curves into planar or non-planar, as planar creases are easier to digitize. Planar curves can be identified by rotating the model into a perspective that visually collapses the curve into a straight line. Alternatively, if the curve is near the exterior of the model, placing a model crease-down onto a flat surface tests for planarity. If a crease is planar, the parametric model could simply be generated by mirror reflecting a developable surface along a plane, see Section 4.3.

We can judge if a crease is planar or not by looking at the ruling directions on both sides. A curved crease is planar, if and only if the incident rulings are aligned in the development, see Fuchs Tabachnikov [5]. For example, a crease between a cylinder and a cone cannot be planar.

4. Comparison of three methods to generate digital parametric curved creased models

In the following sections we will present three methods for digitizing a model following the initial geometric examination. As discussed later, these three approaches are not specific for the tripod example but can be candidate methods for other curved crease models.

Method a): *isometry enforcement using error minimization based on dynamic relaxation* (Figure 2.a). We create a polygonal mesh based on the estimated rulings with thin triangles and/or planar quads, and optimize for equal lengths in the 2D and 3D patterns, similarly to the method by Tang *et al.* [15].

Method b): *constructive approach for generating a curved crease folding from one patch of a developable surface given additional geometric constraints in 3D space* (Figure 2.b) *based on the method by* Mundilova [11]. From the one surface represented by a NURBS surface, we construct the other surface (either cylinder or a cone) to pass through apex positions or to have ruling directions by analytical solutions per each ruling.

Method c): *mirror reflection of a set of developable surfaces* (Figure 2.c). Given NURBS surfaces, we slice them with a plane and mirror reflect the other part, primarily adopted by Huffman and Resch (Demaine [4]).

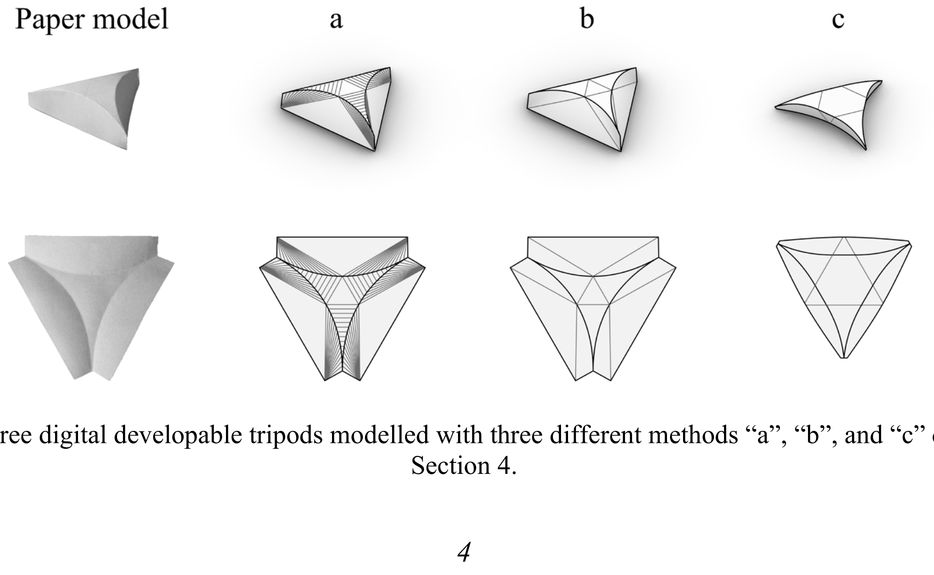

Figure 2: Three digital developable tripods modelled with three different methods "a", "b", and "c" described in Section 4.

4.1. Method a: Isometry Enforcement

Method "a" (Figure 2.a) is based on the developability enforcing of an approximated 3D folded polygonal mesh. We implemented our algorithm with Kangaroo and Grasshopper (Piker [13]). With these tools the geometrical constrains are assigned through a set of predefined "goal nodes" which generate moving vectors on each vertex of the mesh. The solver then iteratively advances until it reaches an input tolerance.

We generate a parametric tripod with a straight boundary as follows. First, we roughly model the folded configuration as a polygonal mesh divided along the guessed ruling of the paper model (Figure 3 step 1). The obtained mesh is not necessarily developable. Then, we forcefully flatten the polygonal mesh into a plane to generate an approximated 2D planar crease pattern with the same graph topology (Figure 3 step 2). This flattening is done by pulling every vertex to the ground plane (OnPlane Goal) while permitting elastic deformation of the edges (Length Goal). In some special cases, e.g., if the mesh folds on itself, it might be useful to guide the flattening at the beginning of the simulation by applying an additional angle constraint that equalizes the angle between consecutive faces to zero (Hinge Goal). Finally, the lengths of each pair of corresponding edges (in the folded and unfolded models) are corrected to have matching lengths (EqualLength Goal) (Figure 3 step 3), while keeping the 2D vertices on the ground plane (OnPlane Goal). It is often useful to give a bending stiffness (Rod Goal) to the curved foldline, so it will be a smooth curve in 3D. Because the bending stiffness leaves residual energy in the equilibrium it needs to be reduced to zero at the end of the simulation to guarantee the developability of the mesh. The algorithm will iteratively modify the 2D pattern and the 3D model until they are isometric to each other and thus converge to a developable mesh.

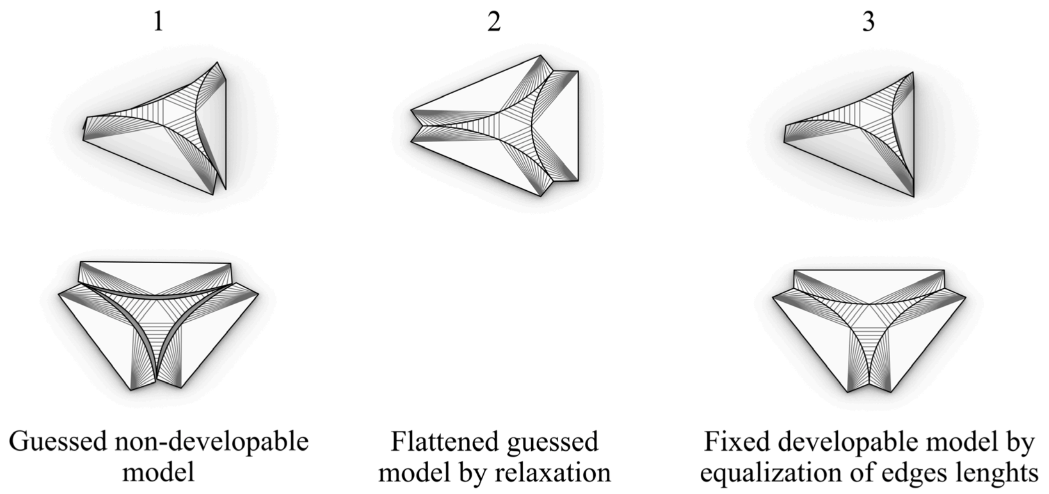
1
2
3
Guessed non-developable model
Flattened guessed model by relaxation
Fixed developable model by equalization of edges lenghts

Figure 3: Isometry enforcement based on a discretized mesh; from left to right 1) generating the model guessing the ruling regardless of its developability, 2) flattening the model forcefully by relaxation, 3) iteratively correcting the edge lengths of the initial guessed model and the pattern by equalizing 2D and 3D lengths, additional constraints are added to induce specific properties to the final model (such as snapping vertices at the base).

By only imposing the aforementioned constraints, the 3D shape might change in unpredictable ways, especially if the initial guess is far from being developable. Therefore, additional constraints need to be added to more carefully guide the positions of the 3D vertices. In this case, we want the planar top patch to keep its position, so we lock its vertex positions (Anchor Node). If the user has a different design intention, they could lock other vertex positions, e.g., all vertices along a planar triangle as can be seen in Figure 4.

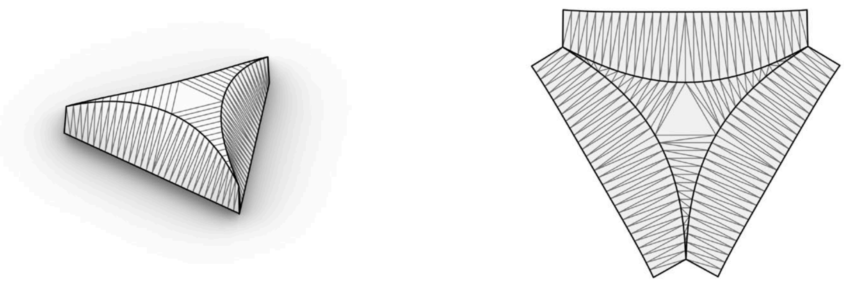

Figure 4: Wrong ruling guessing, which still gives a developable result due to dense triangulation, however the surfaces do not approximate smooth surfaces.

In this method, it is crucial to start with a good guess of the ruling distribution of the initial 3D folded mesh, otherwise it might never reach a perfectly developable result, or it might find a developable solution that is undesired. The ruling configuration in Figure 3 is a result of geometric constraints concerning developable surfaces. As it is not possible to generate a developable surface between a space curve and a straight line, the rulings form a large triangle and two conical configurations at the end of the straight curve. Figure 4 exemplifies a perfectly developable solution with a wrong ruling assignment. This type of ruling is not possible with non discretized surfaces; however, it becomes possible when the surface is pleated along the triangulation, even though the mesh does not approximate a smooth surface.

Moreover, the definition of additional geometric constraints must be carefully evaluated to avoid unexpected results or incompatibilities which might prevent finding a developable solution.

4.2. Method b: Constructive Method

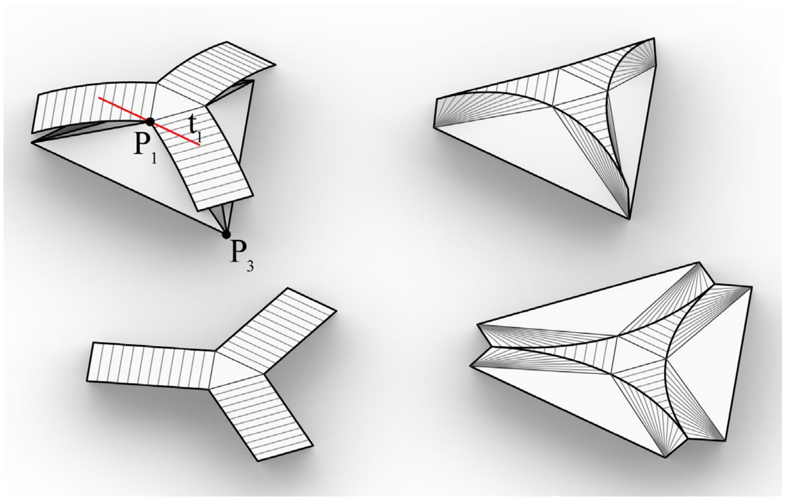

Figure 5: Generating a tripod with constructive approach and lotus plugin for Grasshopper.

The second method uses the newly developed set of components by the authors called "Lotus" for Grasshopper for Rhino based on Mundilova [12]. Mundilova proposes a constructive method that takes an arbitrary developable surface as an input and computes the curved folding composed of the input developable surface and a generalized cylinder or a cone with a specified 2D and 3D ruling direction or apex, respectively. This method considers corresponding rulings in 2D and 3D and finds the unique crease point on each ruling that satisfies length preserving distance constraints. As such, the location of a point on a ruling can be computed analytically. In particular, its explicit local evaluation does not require optimization.

"Lotus" is implemented as a Grasshopper plug-in for Rhino6 (McNeel [9]) and works for any developable surface given as a NURBS surface patch of degree $1 \times n$ with two types of user input. The user input is either (1) two points $\mathbf{P}_1$ and $\mathbf{P}_2$ that are interpolated by the crease curve and the point $\mathbf{P}_3$ specifying the generated cone apex or the ruling direction $\mathbf{P}_1\mathbf{P}_3$ of the generated cylinder; or (2) a point $\mathbf{P}_1$ and intended tangent direction $\mathbf{t}_1$ at $\mathbf{P}_1$ on the crease curve and the point $\mathbf{P}_3$ specifying the generated cone apex or the ruling direction $\mathbf{P}_1\mathbf{P}_3$ of the generated cylinder.

To construct a tripod with a straight boundary, we first design the top surface by constructing a base triangle in the *xy*-plane and tangent continuous cylindrical surfaces at the appropriate locations, see Figure 5. Then, we create three planar triangle patches on the side. We attach cones between the triangle patches and the top surface using Lotus. In order to maintain the tangent continuity (G1) between the generated cones and the side triangles, we choose the tangent lines of the three creases at the top to be the intersection of the planes containing the side triangles with the *xy*-plane. Using Lotus with the *xy*-plane triangle points as $\mathbf{P}_1$, appropriate tangent directions and the remaining corners of the side triangles as apices $\mathbf{P}_3$, we find the crease curves. This results in a closed shape as the angles between the tangent and triangle sides are same in 2D and 3D. Furthermore, the creases are tangent continuous by construction. The location of the cone apices can be adjusted to change the curved crease layout. Note, that this procedure can be applied to not only rotationally symmetric designs.

Lotus generates parametric models. It can accept as inputs any type of developable surface; however, it can only generate cylinders and cones as outputs. Specifically, it cannot handle a crease between two tangent developables, in which case, either of the surfaces needs to be approximated by the combination of cylinders and cones.

4.3. Method c: Mirror Reflection

The mirror reflection method has been widely studied; the earlier examples are from Huffman and Resch (as explained in Demaine [4] and applied in Mitani [11]). It consists of slicing and reflecting a surface with a plane, which results in a planar crease. If the surface is developable, this process guarantees preservation of developability. If the starting surface is curved and has no kinks, the crease will be a smooth curve as well. Furthermore, mirror reflection does not change the ruling directions across a crease and thus results in aligned rulings in the development.

Because of the planarity limitation, the obtained model cannot satisfy all properties existed in the original model. In particular, we cannot achieve a straight boundary. The tripod has a planar top face and three legs with continuous curved creases; thus, we first construct the top surface composed of multiple developable patches connected with at least G1 continuity and apply the mirroring procedure three times with three planes, see Figure 6.

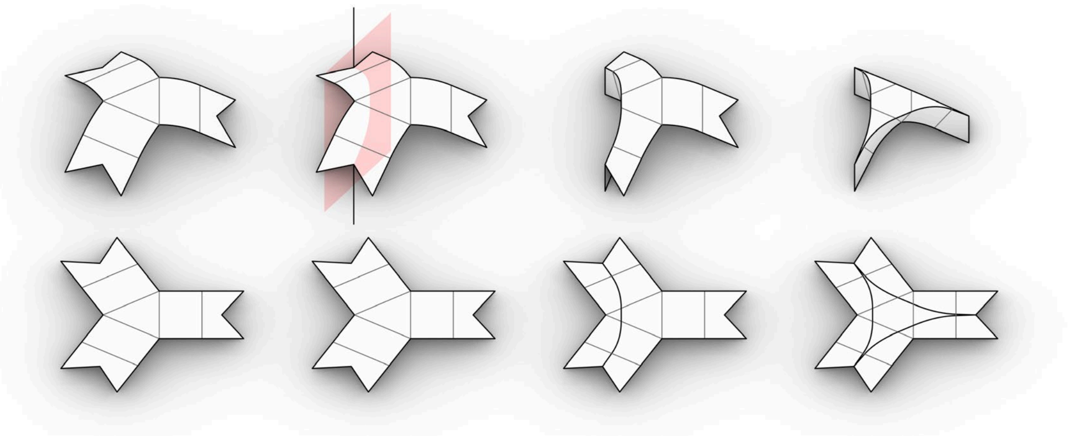
Figure 6: Generating a tripod with the mirror reflection method.

The first step is building the top surface with the required properties. The tripod that we consider is symmetric thus we focus on one crease only. The crease passes through two adjacent legs and the top flat surface and intersects the perimeter of the paper on two points, so the reflection plane passes through those two intersection points (Figure 6 step 2). By tilting the plane about the axis between the intersection points, we obtain variations of tripods. Because it is not possible to make the reflected cylinders to be vertically aligned at the end, we add an extra planar surface at the legs.

The mirror reflection method is limited when applied to this tripod model in contrast to methods "a" and "b". For example, the top planar patch with this method cannot be a triangle but needs to be a truncated triangle (hexagon), and thus the model contains a straight-line segment in the crease. This is because the method requires to have unfolded original developable surface, where the rulings of adjacent cylinders would intersect if there were no truncation. Another limitation of this method is that depending on the proportion of the original triangle, the boundary surface will not reach the ground plane. Additionally, it is not possible to generate a tripod with a boundary of straight lines. This is because we would require starting with a completely planar preliminary surface to have straight edges, and thus there would not be any curved crease. Figure 7 shows a tripod generated with method "c" compared to a tripod generated with method "b".

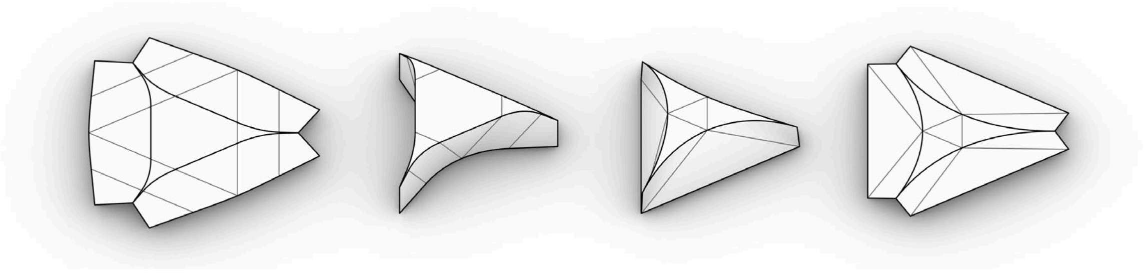
Figure 7: Comparison of the tripod generated with method "c" (left, development and folded model) and the tripod generated with method "b" (right, folded model and development).

5. Discussion / Comparison

According to the case studies proposed we observe that each of three methods has its own limitations regarding their input geometry, the parameterizability of the model, and the prerequisite knowledge required.

5.1 Limitation on Input Model

When the input model contains non-planar creases, we cannot apply method "c", and need to compromise the design. If the input model contains creases between tangent developables method "b" cannot be applied, though, it is possible to approximate tangent developables by a series of cylinders or cones. Because method "b" computes one surface from the other, a cycle of surface patches cannot be computed (see Figure 8 Left).

5.2 Limitation on Output Model

Method "c" computes the most accurate result as it would only suffer from the error given by the floating-point representation in the CAD system. Method "b" is nearly as accurate; the algorithm requires sampling rulings, which has the potential to introduce error, but a large enough number of ruling samples increases the accuracy as needed. Method "a" is the least accurate of the three as it discretizes the curved surface into planar faces. Increasing the number of samples could increase the accuracy.

However, we observed simulating very dense meshes is often unstable and slow to simulate. Also, because of its optimization scheme, some error residual may still exist for a large number of faces.

As for interactivity, methods “b” and “c” compute the results immediately when users change the input parameters, while method “a” requires more computational time to solve the geometric problems and can be time consuming for a complex model. Method “c” has fewer parameters and more difficult to adapt, while method “b” offers more design freedom. Method “a” can take additional design and geometric constraints as needed.

Methods “b” and “c” work with continuous NURBS surfaces, method “a” only works with discretized meshes. Method “c” is the most accurate as it only suffers from the error given by the tolerance of the software itself. Method “b” is nearly as accurate; the algorithm requires discretizing the curve, which has the potential to introduce error, but a large enough number of ruling samples increases accuracy. Method “a” is the least accurate of the three as it discretizes the curved surface into planar faces. However, if the discretization is dense enough the final mesh might be converted into a NURBS surface by interpolation. Nevertheless, it must be pointed out that error distribution with very dense meshes might be unstable or too slow to simulate.

5.3 User Knowledge

Designing models based on any of the three methods require some intuition and geometric knowledge to handle surface patch types, however method “a” is probably the most forgiving because the error distribution process could fix potential ruling errors. Methods “b” and “c” are more precise. Method “a” requires more set up time (conversion from NURBS to mesh, set up of Kangaroo goal nodes, definition of the 2D pattern) and the knowledge about the constraint-based system (Kangaroo and goal nodes). In the design process, method “a” can be used as a stepping stone to method “b” or “c” if the result of “a” shows a promising comparison to a more precise construction method.

5.4 General Comparison

Method “a” is probably the most versatile in terms of possible shapes it can handle even if it cannot give smooth surfaces as a result; on the contrary, method “c” is probably the least versatile because it is restricted to planar curved creases. Method “b” has the potential to be both versatile and accurate, but it is limited to sequential computation. Figure 8 showcase a series of selected curved fold examples from various artists and applicable methods.

Note that only method “a” can handle the first example. In particular, method “b” cannot handle the problem because they form a cycle of surfaces that cannot be sequentially solved. However, note that the result from method “a” is still an approximation, i.e., the planarity of each quadrangle is not satisfied, so the ruling directions are not correct. It is still open if the pattern exists or how it works. The second and fourth examples are effectively computed by method “c”. Creases created by method “c” can also be made with method “b” except for folds adjacent to two tangent developables, as in the fourth example. Method “b” (but not method “c”) can solve models composed of cylinders and cones which intersect at non-planar creases, as in third example. Note that serial computation in mimicked in the third model using method “b” by reflecting one corner.

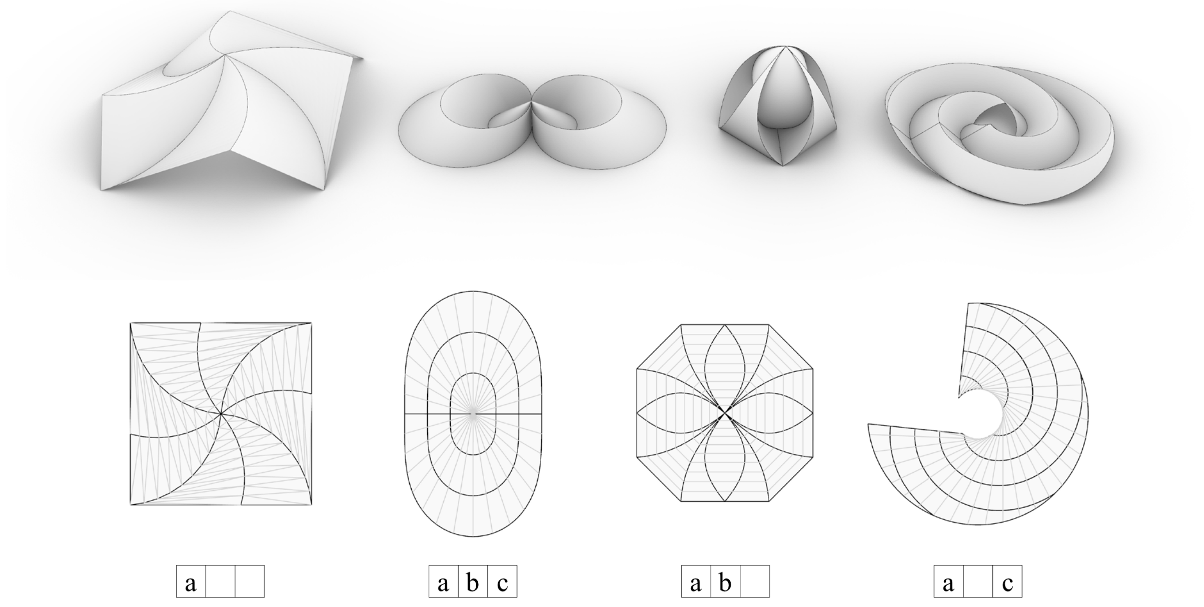

Figure 8: Modified models designed by (left to right) David Huffman, Ron Resch (Yellow Cones Kissing), Jeannine Mosely (Origami Bud), Riccardo Foschi (Taijitu) and digitized with our presented methods. The models were chosen for their specific properties (left to right): creases with torsion and tangent developable surfaces, flat creases and cone surfaces, creases with torsion and cylinder surfaces, planar creases and tangent developable surfaces.

6. Conclusion

In conclusion, we observe that all three compared methods need a certain amount of background knowledge to be able to apply them properly and achieve advanced and correct results. We describe how to analyze in advance the paper model by observation and apply our findings to digitize the model, including evaluating ruling direction and identifying surface patch types. We also describe in detail three methods for digitizing models, noting that there is a trade-off between accuracy and versatility. The limitations of each method discussed in the paper will guide the designer not only to identify the method to represent the input exactly but also to find the proper and necessary design modification to comply with each method.

Acknowledgements

This research started at a Structural Origami Gathering 2020+1/2, and we thank the participants for their input and feedback.

This work was also supported by the Department of Architecture of Bologna University, the JST PRESTO JPMJPR1927 and the FFG Austria, Project #864731.

References

[1] Bhooshan S., *"Interactive Design of Curved-Crease_Folding",* Master Thesis. Bath: University of Bath, 2015.

[2] Brancart S., Vergauwen A., Roovers K., van den Bremt D., de Laet L., de Temmerman N., "*UNDULATUS: design and fabrication of a self-interlocking modular shell structure based on curved-line folding", in* Conference proceedings: International Association for Shell and Spatial Structures (IASS) Symposium 2015: Future Visions, 2015

[3] Demaine E., Demaine M. et al. *"(Non)Existence of Pleated Folds: How Paper Folds Between Creases"*. Graphs and Combinatorics. 2011, Bd. 27, 377.

[4] Demaine E., Demaine M., Koschitz D., and Tachi T., *"A review on curved creases in art, design and mathematics"*, in Symmetry: Culture and Science, volume 26, number 2, 2015, pages 145–161.

[5] Fuchs D. and Tabachnikov S., "*More on Paperfolding*". The American Mathematical Monthly, Vol. 106, No. 1, pp. 27–35, 1999.

[6] Ghassaei A., Demaine E., and Gershenfeld N., *"Fast, Interactive Origami Simulation using GPU Computation"*, in Origami7: Proceedings of the 7th International Meeting on Origami in Science, Mathematics and Education (OSME 2018), volume 4, Oxford, England, September 5–7, 2018, pages 1151–1166, Tarquin.

[7] Kilian M., Flöry S., Chen Z., Mitra N. J., Sheffer A., Pottmann H., *"Curved Folding"* in ACM Transactions on Graphics, 2008, volume 27, number 3, 2008, pages 75, 1-9

[8] Koschitz D., *"Designing with curved creases"* in Advances in Architectural Geometry 2016, H. E. Z. vdf, Hrsg., Zurich, vdf, Hochschulverlag, AG ETH Zürich, 2016, pp. 82-103.

[9] McNeel. Grasshopper - Generative Modeling for Rhino, http://grasshopper.rhino3d.com/

[10] Mitani J., *"A Design Method for 3D Origami Based on Rotational Sweep*" in Computer-Aided Design & Applications 6(1), 2009, 69-79.

[11] Mitani J. and Igarashi T., "*Interactive Design of Planar Curved Folding by Reflection"* in Pacific Graphics Short Papers, 2011.

[12] 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

[13] Piker D. "*K2-Goals"* in https://github.com/Dan-Piker/Kangaroo-Documentation

[14] Tachi T., *"One-DOF Rigid Foldable Structures from Space Curves"* in Proceedings of the IABSE-IASS Symposium 2011.

[15] Tang C., Bo P., Wallner J., and Pottmann H. "*Interactive design of developable surfaces"*. ACM Trans. Graphics 35/2 (2016), #12, 1-12. [doi].