Journal for Geometry and Graphics
Volume 16 (2012), No. 2, 185–194.

Column-shaped Origami Design Based on Mirror Reflections

Jun Mitani

Graduate School of Systems and Information Eng'g, University of Tsukuba / JST ERATO
Tsukuba Ibaraki 305-8573, Japan
email: mitani@cs.tsukuba.ac.jp

Abstract. Origami, which is the construction of an object by folding a single sheet of paper, has been studied in the fields of mathematics and engineering. Several software programs for designing origami shapes have been recently developed, and so we can now obtain a variety of shapes. However, the variations designed with them are still limited. In this paper, we propose a new method for designing origami based on the sweep and reflection operations. As an example, a column-shaped origami object with a relief structure is designed using our method. The shape is defined by a profile polyline and a trajectory polyline that are input by the user. We developed a system on which the user inputs and edits the polylines to generate both the crease pattern and the 3DCG model of the origami object in real time. With the tools implemented in our system, the user can design a wider variety of shapes in a trial-and-error manner while seeing the effects of their operations. Thus, users can easily design stable and geometrically attractive shapes. We indicate the effectiveness and the potential of our system by showing some examples.

Key Words: Origami, crease pattern, mirror reflection, computational origami
MSC 2010: 52B70, 51N05, 68U05

1. Introduction

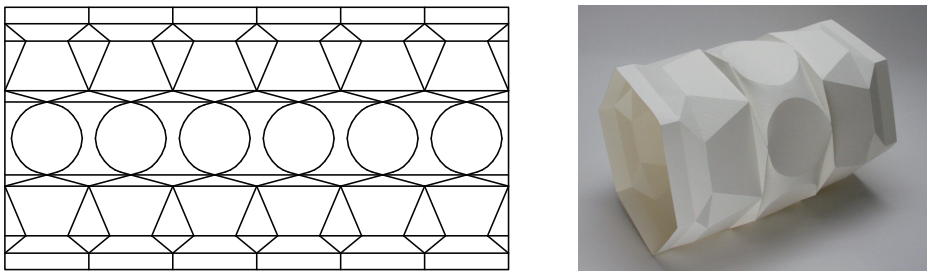
Origami is traditionally viewed as the art of making a 3D object by folding a single sheet of paper. However, the technology of origami is useful for engineering since it helps to reduce the volume of products by folding and fabricating products efficiently from flat sheet material. Several software programs for designing the shapes of origami have been recently developed, and so we can now obtain a variety of origami shapes. However, the variations designed with them are still limited.

In related studies, the author proposed a method for designing three-dimensional origami based on rotational sweep [3]. The internal shapes of the origami objects designed with this

ISSN 1433-8157/$ 2.50

method are limited to a solid generated with rotational sweep. The author together with T. IGARASHI proposed an interactive user interface for designing curved origami having planar curved folds [4]. This approach is based on the fact that when a part of a developable surface is reflected with a simple mirror operation, the resulting shape can be reconstructed with a single sheet of paper without tearing. The user can interactively move a point on a surface by mouse dragging and see the shape change accordingly.

In this paper, we propose a new system which combines the advantages of both rotational sweep and mirror reflection approaches. Column-shaped origami objects with a relief structure, as shown in Fig. 1, are designed with a simple user interface on our system. The algorithm is based on the abovementioned two approaches. The user inputs only the profile curve and the trajectory curve, which are represented by polylines. When the user inputs the curves on the screen, a crease pattern and a 3DCG model are generated automatically in real time. The shapes designed with this approach cover not only the cylindrical shapes designed with *ORI-REVO* [5] but also other variations, such as the diamond pattern (Fig. 9). It is possible to design origami objects that have the same topology as a disc or a tube. By approximating a smooth profile curve by a set of tiny line segments, curved surfaces can be used in the design. The theory behind our system is simple and not new. However, we can explore a variety of geometrically attractive origami shapes quickly and easily by providing a well-conceived user interface. The proposed approach is flexible and applicable for a wider range of origami objects.



(a) (b)

Figure 1: (a) The crease pattern. (b) A photo of the fabricated origami.

2. Related work

When an origami object is unfolded, mountain and valley creases appear on the sheet of paper. The relation between the crease pattern and the folded shape has long been the target of studies on the design of origami. Most of the methods for designing origami objects focus on the geometry of the crease pattern. After computers became popular, it was possible to design sophisticated origami objects by using software programs. A number of software programs that generate crease patterns for origami have been developed, and now some useful tools are available on the Internet. The research field of designing origami with a computer is now called *"computational origami"*.

TreeMaker is software used to design flat foldable origami [2]. This software generates crease patterns from a graph tree that represents the base structure of the object by using the circle-river packing algorithm. *Tess* is software for designing a mosaic pattern that appears when a sheet of translucidus paper is folded [1]. In this software, a crease pattern is generated

from a basic tiling pattern, selected from a predefined pattern list by the user, and some parameters. *Origamizer* is software that generates a crease pattern from an input 3D polygonal model [6]. By using *Origamizer*, an arbitrary 3D polygon model that is topologically the same as a disc can be fabricated with a sheet of paper.

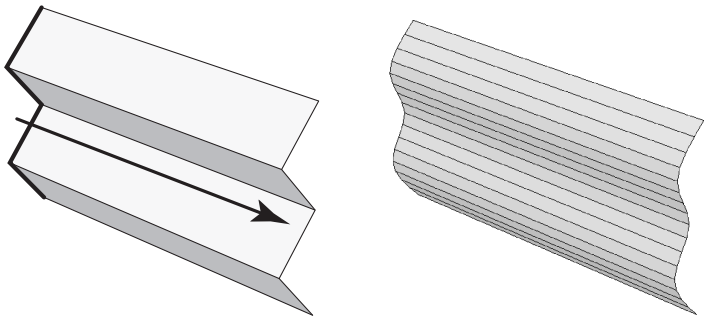
The software programs introduced in this section handle only straight creases and flat surfaces. However, *ORI-REVO* [5] and our system can design an approximated curved surface.

3. Theoretical concept

The shape of the origami objects designed with our system is generated by applying reflection operations to the initial developable surface. In this section, we describe the details of the basic theoretical concept on which our system stands.

3.1. Initial developable surface

It is well known that surfaces that can be made with a single sheet of paper are limited only to developable surfaces. We start with a developable surface to design the final shape. The surface we use here as the initial surface is a cylindrical surface, which consists of an infinite number of parallel straight lines. The cylindrical surface is defined as a swept surface whose trajectory is a straight line. In our system, the profile curve, which is input by the user, is represented as a polyline lying on a vertical plane. We call this polyline the "profile polyline" hereinafter. The trajectory is defined as a straight line lying on a horizontal plane perpendicular to the plane on which the profile polyline lies. Therefore, the initial surface is defined as a set of rectangles, as shown in Fig. 2. A smooth curved surface can be approximated by using a fine profile polyline that has many tiny line segments.



(a) (b)

Figure 2: (a) An initial surface generated by sweeping a profile polyline (thick line) along a trajectory line (arrow). (b) A curved initial surface approximated by a set of thin rectangles.

3.2. Applying reflection operations

The surface obtained by applying a reflection operation to a part of the initial surface is a surface which has a crease (Fig. 3). The crease corresponds to the intersection between the initial surface and the reflection plane. This surface can be physically made with a single sheet of paper by folding the initial surface along the crease without tearing and distortion.

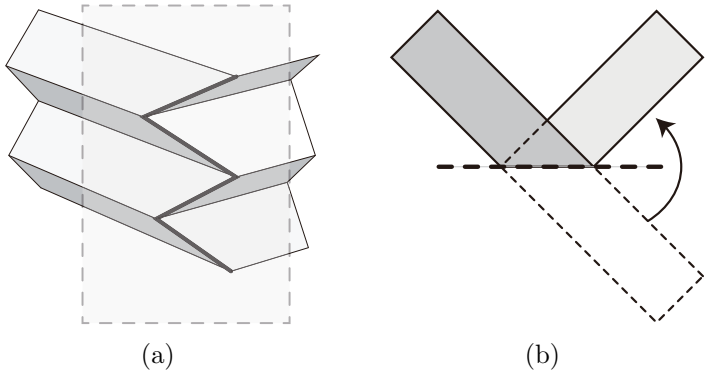


Figure 3: (a) A surface obtained by the reflection operation of a plane (denoted as a dashed square). (b) The top view of the surface.

Since any part of the surface even after this folding is still developable, the surface generated by applying the reflection operation several times is always developable and can be made with a single sheet of paper. This is the basic operation we use for designing a new origami model.

3.3. Location and orientation of reflection planes

To design a shape, the reflection planes have to be appropriately located and orientated. However, providing the user interface to control them intuitively is not straightforward. Furthermore, it is sometimes difficult to recognize the relation between the generated shape and the location or orientation of the reflection plane, because a slight difference of the orientation causes a significantly different resulting shape. These problems motivated the author and Igarashi to develop a system on which the location and the orientation of a reflection plane are implicitly set when a user drags a point on the surface [4]. Our approach is similar to their approach. In our system, the location and the orientation of a reflection plane are automatically and implicitly defined by a polyline that the user inputs as a trajectory.

When the user inputs a polyline as a trajectory on a horizontal plane, the system locates and orientates the reflection planes so that a ray reflected by them coincides with the trajectory. This means that each reflection plane is located so that a corner of the polyline lies on it. Then the reflection plane is orientated so that its normal vector coincides with the bisector of the corner angle. Figure 4 illustrates the relation between the trajectory polyline (user input) and the reflection planes (system output). Since we constrain the trajectory path so that it lies on a horizontal plane, all reflection planes are orientated vertically.

3.4. Construction of a 3D model

The system places the reflection planes by referring to the trajectory polyline. Then the system constructs a 3D model by reflecting the initial developable surface at every reflection plane. Since the initial surface consists of a set of parallel lines, as shown in Fig. 2, the 3D model is generated by reflecting it at every reflection plane, as shown in Fig. 5. To help the user designing a new shape in a trial-and-error process, the system updates the 3D model in real time as the user edits the trajectory.

The number of polygons constituting the 3D model is the product of the number of line segments in both the profile polyline and the trajectory polyline. Therefore, the number can

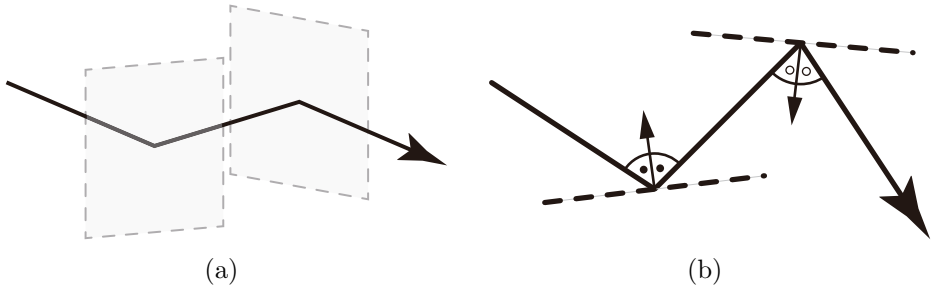


Figure 4: (a) Reflection planes placed so that the reflected path coincides with the user input of the polyline. (b) The path (large arrow) and the reflection planes (dashed line) in the top view.

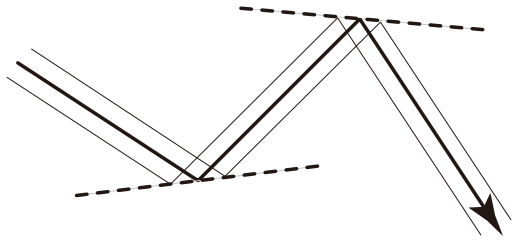


Figure 5: Relation of the trajectory (arrow) and the line elements (thin lines) of the 3D model reflected by the reflection planes (dashed lines) in the top view.

be a thousand at most, and it is small enough to construct and display the model in real time on today's PCs.

3.5. Generating a crease pattern

Since the 3D model is represented as a polygonal model, the development of the model is obtained by placing the polygons onto the 2D plane one by one.

However, the crease pattern of the 3D model generated with our system is obtained with simpler algorithm. Because the polygons composing the 3D model generated our system are all trapezoids, the crease pattern can be generated by simply adding skew line segments in the parallel crease pattern of the initial model, as shown in Fig. 6. The distances between neighboring skew lines are defined as the distances between the points where a line element is reflected by a reflection plane. Note that the reflection operations do not change the initial crease pattern: these operations just add new skew creases. Therefore, the pattern is generated quickly enough to follow interactive editing.

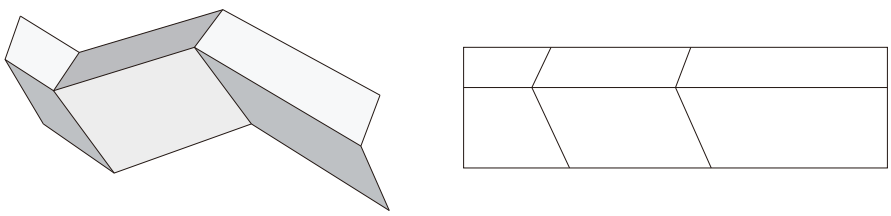


Figure 6: Relation of the 3D model (left) and the crease pattern (right). The crease pattern fits inside the rectangle.

4. User interface

Since just two polylines, the profile polyline and the trajectory polyline, determine the 3D shape, the system is quite simple. However, to produce easily both stable and attractive shapes, a user interface for precisely controlling the polylines is required. In this section, we describe the details of our system from the viewpoint of the user interface.

4.1. Rotational symmetry

Symmetry is essential for geometrical attractiveness. When the trajectory polyline has rotational symmetry and is closed (the start point and the end point are located at the same position), the resulting shape becomes a column-like shape whose cross section has rotational symmetry. Therefore, providing commands for designing the symmetrical trajectory is important. We prepared some templates, as shown in Fig. 7, so that the user can input a symmetrical trajectory polyline easily. By controlling parameters such as the radius and the number of corners, the user can obtain a variety of geometrically attractive shapes. Since the concavity and convexity are flipped at the reflection plane located at a corner of the trajectory polyline, the number of corners has to be even so that the start point and the end point meet.

The variation of shapes generated with this approach covers the cylindrical shapes designed with *ORI-REVO*.

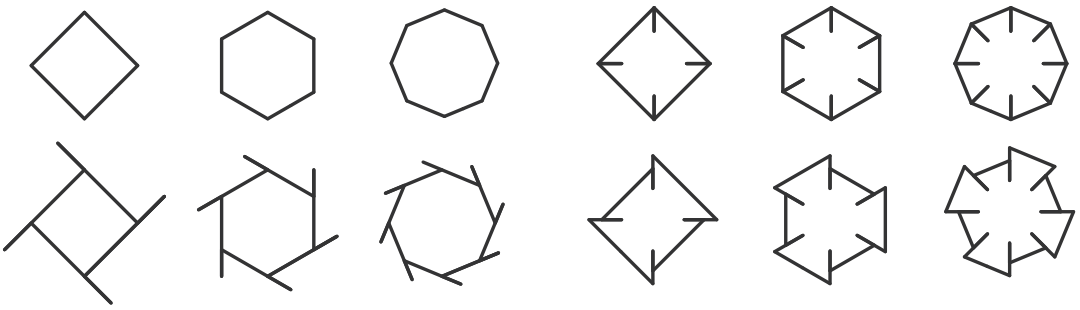
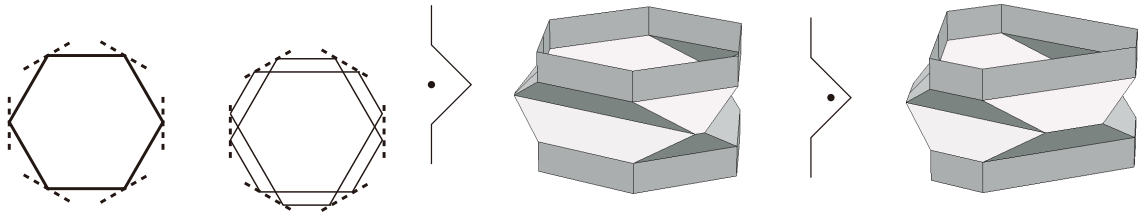


Figure 7: Examples of templates for the trajectory path. The user can modify the shape by controlling the parameters set to each template.

4.2. Position adjustment of the profile polyline

The position of the line elements of the initial surface relative to the trajectory polyline affects the resulting shape, even though the shapes of both the trajectory polyline and the profile polyline do not change. Figure 8 shows this effect. Illustrated as dashed lines at the left of Fig. 8(a), the location and orientation of the reflection planes are defined by the trajectory polyline (a regular hexagon in this case). As illustrated in the right of Fig. 8(a), the paths of the line elements of the 3D model (solid lines) differ according to the position relative to the trajectory. Therefore, the same profile polyline and trajectory line generate a 3D model of different shapes when their relative position changes.

Note that in Fig. 8(b) the top and the bottom of the profile polyline are located at the same position vertically as the end point of the trajectory. This input makes the regular hexagonal openings of both the top and the bottom. In contrast, the openings of the top and bottom are a non-regular hexagon in Fig. 8(c). This figure shows the importance of position adjustment of the profile polyline relative to the trajectory.



(a) (b) (c)

Figure 8: The trajectory polyline and reflection planes (left of (a)). The paths of the line elements of the 3D model defined by the reflection planes (right of (b)). The same profile polyline and trajectory line generates different shapes of the 3D model ((b), (c)). A dot shows the point view of the trajectory polyline.

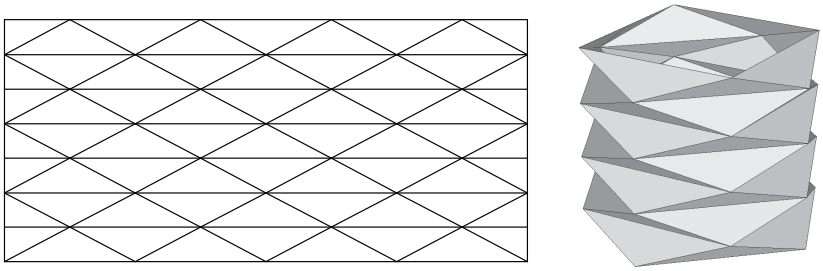
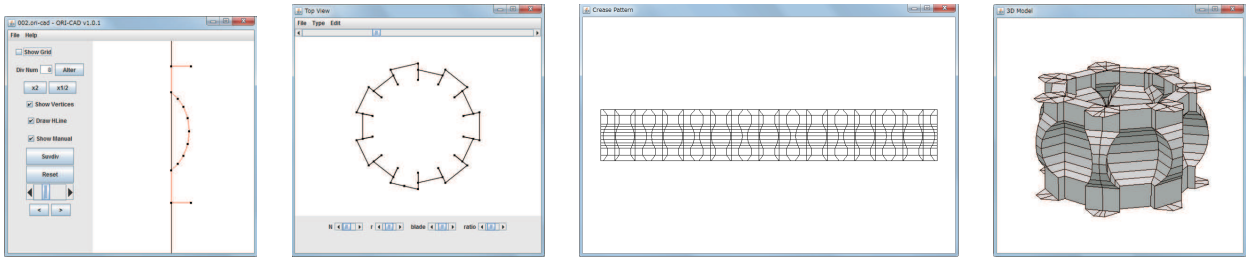


Figure 9: The diamond pattern, created with some degenerated line elements.

Hence, we added a user interface to our system to control the position of the profile polyline relative to the trajectory after the user defines both shapes. By controlling the position so that some line elements degenerate (having zero length), we can design the well-known diamond pattern shown in Fig. 9.

5. Result

We implemented the algorithm and the user interfaces described in the previous sections with a Java application. The application is composed of four windows (Fig. 10).



(a) (b) (c) (d)

Figure 10: Application windows of our system. These windows show (a) the profile polyline, (b) the trajectory polyline, (c) the crease pattern, and (d) the 3D model.

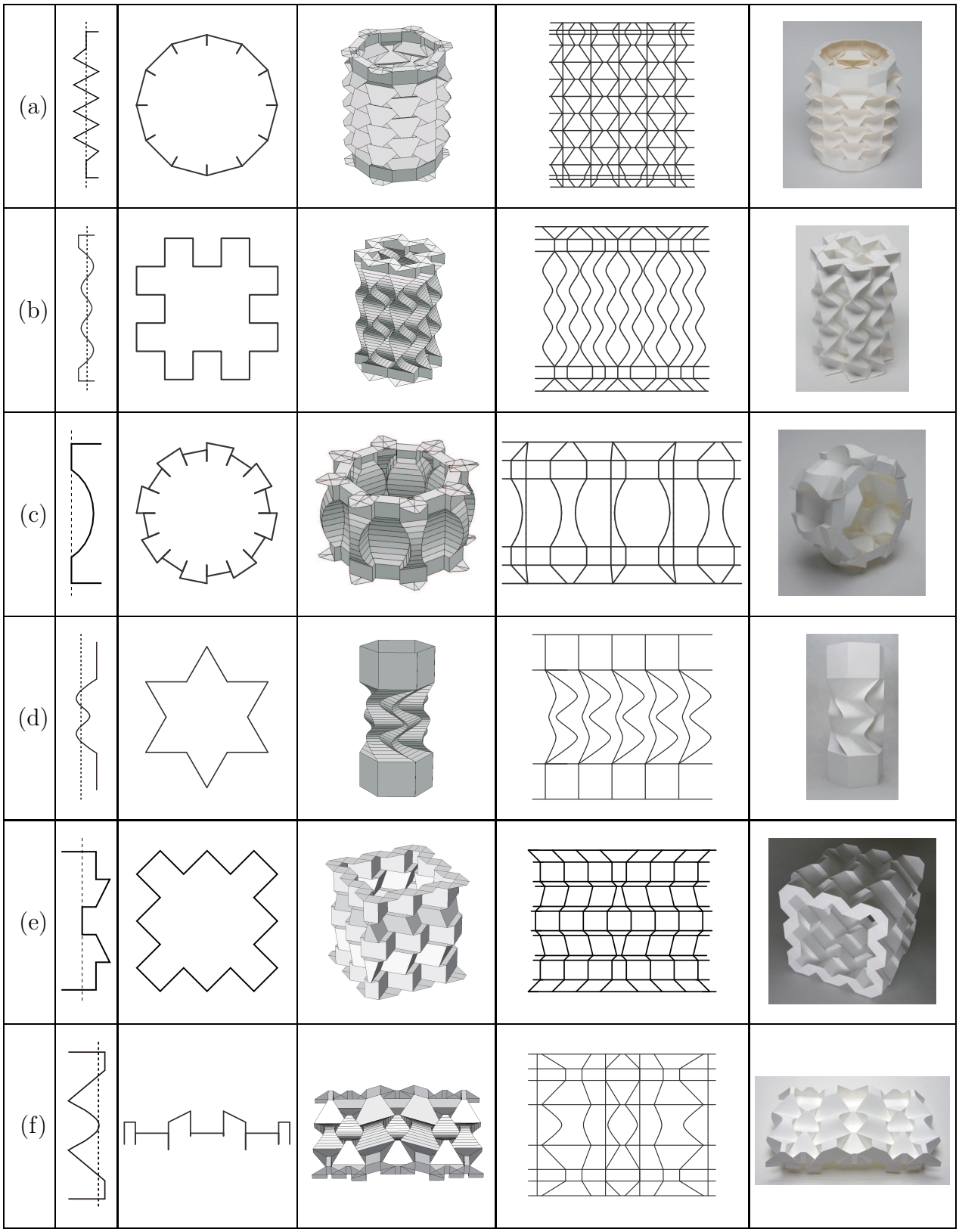


Figure 11: Origami objects designed with our system. From left to right, each line shows the profile polyline, trajectory polyline, 3D model, a part of crease pattern, and photo of the fabricated model.

(a) A main window on which a user inputs a profile polyline.

(b) A top view window on which the user inputs a trajectory. The user can select a predefined trajectory pattern from the menu. The user can also input an arbitrary polyline independently without using the patterns.

(c) A window that displays the crease pattern.

(d) A window that shows the 3D model. The user can rotate the model to see it from an arbitrary angle.

Figure 11 shows origami objects designed with our system. Some of the models have curved surfaces approximated by a set of thin rectangles. Since we can omit the folding when a crease is almost flat, the surfaces are reconstructed by moderately bending a sheet of paper. Most of the models are column-like shapes designed with a closed trajectory, and they are stable when fabricated. Figure 11(f) is an object whose trajectory is not closed.

6. Conclusion

We could design a variety of shapes quickly using our system. We believe these origami objects show that providing a well-conceived user interface makes it possible to easily explore geometrically attractive origami.

The trajectory used in our system lies on a horizontal plane. Although this limitation helps to simplify the user interface, the variety of shapes designed with this system is limited. The trajectory does not actually need to lie on a plane to implement the algorithm. A more complicated 3D model can be generated from a non-planar trajectory with the same algorithm described in this paper. The origami object shown in Fig. 12 is an example designed by using a non-planar trajectory. However, the expansion of the degrees of freedom causes difficulty in controlling the 3D shape. Furthermore, we confirmed that a non-planar trajectory frequently causes penetrations among polygons that are not allowed in fabrication. To find a reasonable balance between the degrees of freedom and the ease of use is one of the topics we will address in future work.

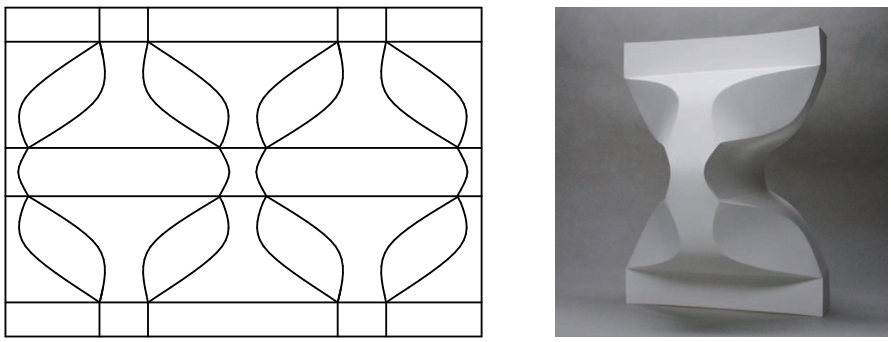


Figure 12: An origami object consisting of five layered components. Two large parts of them were designed by using a non-planar trajectory.

References

[1] A. Bateman: *Paper Mosaic Origami Tessellations*. http://www.papermosaics.co.uk/software.html

[2] R.J. Lang: *TreeMaker*. http://www.langorigami.com/science/computational/treemaker/treemaker.php

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

[4] J. Mitani, T. Igarashi: *Interactive Design of Planar Curved Folding by Reflection*. Proc. of the Pacific Conference on Computer Graphics and Applications 2011, pp. 77–81.

[5] J. Mitani: *PRO-REVO*. http://mitani.cs.tsukuba.ac.jp/ori_revo/

[6] T. Tachi: *Origamizer*. http://www.tsg.ne.jp/TT/software/

Received August 3, 2012; final form November 28, 2012