Rendering method for flat Origami

Jun Mitani

Department of Computer Science, University of Tsukuba, Japan
PRESTO, Japan Science and Technology Agency

Abstract

In flat Origami (Origami which is folded flat), some cases exist that have a closed-loop in the overlap order of faces after they are folded. It is difficult to display this shape correctly on the screen when Origami is expressed by sets of plane polygons of zero thickness as is generally used in CG because all faces are placed on the same plane. In the present paper, we propose a new rendering technique to solve this problem. In the proposed method, we prepare a matrix that represents the overlap relation between two faces and a face ID buffer, the concept of which is similar to a Z buffer in the z-buffer algorithm. With this buffer, the face located in the uppermost is monitored in each pixel at the rendering stage. We render the shape on the face ID buffer using a scanline algorithm and display the folded shape by outputting the result in which the edges are extracted. Moreover, we render the shape in technical illustration style by coloring each vertex according to the number of mountain and valley folds connected to the vertex. In addition, we propose a simple pseudo shading algorithm.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Bitmap and framebuffer operations

1. Introduction

Origami is a form of amusement and art that involves folding paper. By folding a piece of paper, various shapes can be produced from a single sheet of paper. Many studies have investigated the geometry of Origami.There are various traditional pieces that express specific shapes such as “crane” or “frog”, and there are other types of Origami that allow us to enjoy the beauty of geometric shapes such as Unit Origami that combines two or more sheets of paper and Origami Tessellation (Fig.1(b)) that involves Twist Fold (Fig.1(a)) in a repeating pattern.

Generally, Origami Tessellation is a kind of flat Origami. Since the shape of paper is usually expressed as a face of zero thickness when treated by computer, the situation whereby multiple faces overlap on a single plane commonly occurs. The painter’s algorithm (also known as a priority fill) is one method of rendering the appearance of flat Origami, although this algorithm fails when a closed-loop exists in the face overlap order. This failure commonly occurs in Origami Tessellation that has many closed-loops. Although the Z buffer algorithm, which is used for rendering 3D objects, works well for objects that have closed-loops in 3D space, it does not work for Origami Tessellation that is folded flat and all faces have the same depth (z-value). In the present paper, we propose a new algorithm that solves this problem by using a matrix that maintains the relationships of overlapping between every two faces and a face ID buffer that holds the ID of the face that is located at the uppermost of each pixel. As a result, it is possible to correctly render flat Origami that has closed-loops in the overlap order. Moreover, we propose a method for rendering flat Origami in the technical illustration style using the linear interpolation of vertex colors and a method for rendering using pseudo shading.

We explain Twist Fold and Origami Tessellation in Section 2 and introduce related research in Section 3. In Sections 4 and 5, we describe the proposed methods. The results are shown in Section 6, and our conclusions and future research are described in Section 7.

2. Twist Fold and Origami Tessellation

An example of Twist Fold is shown in Fig. 1(a). There is a type of Origami called Origami Tessellation that is composed of multiple Twist Folds as shown in Fig. 1(b).The crease pattern of an Origami Tessellation is generated from a tiled plane. The details are described in [Hul02], and an ap-

plication that generates the crease pattern of Origami Tessellation is available on the Web [Bat]. It is difficult to display the folded shape using common CG rendering algorithms because it has closed-loops in the face overlap order and all faces lie on the same plane (all faces have the same z value).

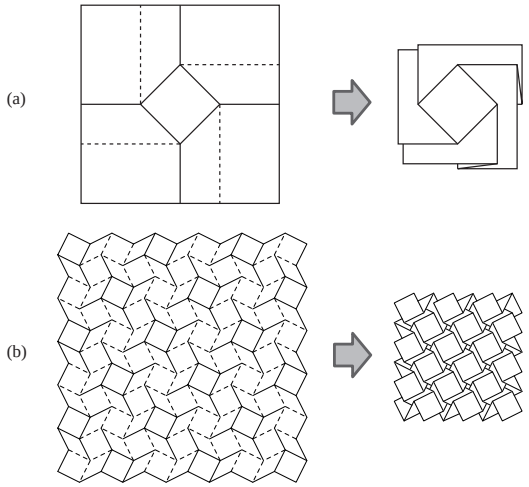

Figure 1: *(a) Twist Fold. (b) Origami Tesselation.*

3. Related Works

Origami has been the subject of numerous studies, most of which have considered the geometry of Origami. Most of the results of recent research in the field of Origami are well surveyed in [DO07]. With the spread of computers, the use of computers in the study of Origami has increased. "Tree Maker" is an application that helps to design Origami by automatically generating the crease pattern from the user's specified skeleton of the target shape [Lan]. Miyazaki et al. [MYYT96] proposed an interface and a data structure that enables users to fold Origami in virtual space interactively. Although the data structure maintains the overlap order, it cannot treat cases that have closed-loops. Furuta et al. [FMF07] also realized virtual Origami folding by adopting spring-mass simulation. Although it is possible to fold various models, it happens that one face penetrates other faces when rendering because face ordering is not considered. Thus, research on Origami by computer has become common but has not yet solved the problem of how to treat special cases when multiple faces lie on the same plane and the overlap order of faces has closed-loops.

4. Overlap relation of faces

4.1. Matrix expression of overlap relation

Although the order of overlapping of faces can be serially defined when no closed-loops exist, it cannot when they exist. Hence, we use a matrix that represents the overlap relation between every two faces. When the number of polygonal faces included in the Origami piece is N, an $N \times N$ matrix can describe all overlap relations. We refer to this matrix as the OR matrix hereinafter. Each element m_{ij} is set to one of the following three states:

- U (Upper) F_i is located above F_j.
- L (Lower) F_i is located below F_j.
- - (Undefined) F_i and F_j do not overlap.

For example, the OR matrix for the simple folding shown in Fig. 2(a),(b) is defined as (c). (In this case, the OR matrix is uniquely defined from the crease pattern. There are cases in which multiple different OR matrixes can be defined from a single crease pattern.)

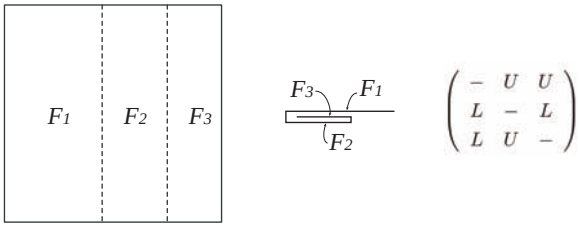

Figure 2: *A simple crease pattern (a), side view of the folded one (b) and the OR matrix (c).*

4.2. Defining the overlap relation

Defining the overlap relation between every two faces from the crease pattern is a difficult problem. Bern and Hayes proved the problem of determining the overlap relation from an arbitrary crease pattern to be NP complete [BH96]. Although determining every element of the OR matrix from a crease pattern is not easy, it can be obtained by a brute-force approach because the number of possible cases is finite. In the present paper, we do not discuss the approach of how to obtain the correct OR matrix and instead assume that a valid OR matrix can be obtained.

5. Rendering

5.1. Render to face ID buffer

Here, we describe how to render folded Origami based on the OR matrix. The basic concept of the proposed approach is similar to the z-buffer method. We prepare a buffer that holds the ID (unsigned integer) of faces with the same size as the rendering area (referred to hereinafter as the "face ID buffer"). The ID of the face that is placed uppermost at each pixel is stored in the corresponding position of this buffer. The ID is stored using the scanline algorithm used in the z-buffer method. Here, the scanlined face ID is overwritten on the buffer only when the position of the buffer is empty or the element m_{ij} is "U" (Upper). (The i is scanlined face ID, and the j is ID already stored in the position.)

5.2. Line style rendering

We then use the Sobel filter that is used in image processing to extract edges to the face ID buffer. The mask of the filter is shown in equation (1). With this filter, we can obtain contours of faces. We export the obtained contours to the frame buffer.

$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, h_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (1)$$

5.3. Technical illustration style rendering

Every face in the flat Origami is placed on the same plane and has the same normal direction. Therefore, all of the faces become the same color when using the common rendering method, which does not use global illumination. This provides poor comprehension of the structure of the Origami piece. Therefore, we add colors to faces to make the structure of the Origami piece easy to understand using a new approach based on the heuristics.

As a generally experienced rule, the vicinity of the valley folded lines becomes dark because little light reaches the area. On the other hand, the vicinity of the peak folded lines becomes bright. Then, we set the brightness B of each vertex of polygonal faces using the following equation:

$$B = (M - V + 2)/4 \quad (2)$$

where M and V are the number of mountains and valleys, respectively, of folded lines connected to the vertex on the contour of a face. Since the values M and V can take 0, 1, or 2, the value of B becomes $0 \leq B \leq 1$. The color of each pixel in a face is calculated by linearly interpolating the colors of vertices on the contour.

When the color of the vertices is defined using only the B, most of the faces are filled with a single color. This yields an unnatural image visually. Hence, we add changes to the brightness of vertices according to the position (coordinate value), so that the inside of faces is colored with gradation. Here, we fixed the color of each vertex V_{color} as follows:

$$V_{color} = RGB(B'r, B'g, B'b) \quad (3)$$

where r, g and b are RGB value of the original color. B' is the blightness calculated as follows:

$$B' = min(1, w_a B + w_b R + w_c) \quad (4)$$

where R is a value in which the distance between the position of the vertex and the upper left corner of the bounding box of the face is divided by the diagonal length of the bounding box which is used to generate gradation. The values of w_a, w_b, and w_c are the weights of the parameters and we can change them so that we can obtain the expected result.

5.4. Pseudo shading

To facilitate the recognition of the structure of overlapping faces, it is desirable that reasonable shading is applied. Again, it is impossible to use standard CG rendering to answer this request because the target shape is flat and all faces are placed on the same plane. Here, we propose a new method that adds pseudo shading using a concept similar to Ambient Occlusion [LB99]. In the method of Ambient occlusion, the intensity of ambient light is adjusted according to the ratio of the existence of a shield object on a sphere that is centered at the target position. For flat Origami, we need only consider blocking by upper faces. Therefore, we use the model in which the intensity of ambient light at a point in the shaded area (Fig.3(a)) is linearly reduced with the ratio of the existence of the blocking object in a circle centered at the position, as shown in the following equation:

$$I = 1 - \frac{s}{S} \quad (5)$$

where I is the pseudo intensity of ambient light, S is the area of the circle centered at the position, and s is the sum of the area covered by the upper faces as shown in Fig.3(b). The value I is calculated for each pixcel and the blightness of the pixcel is multiplied by this.

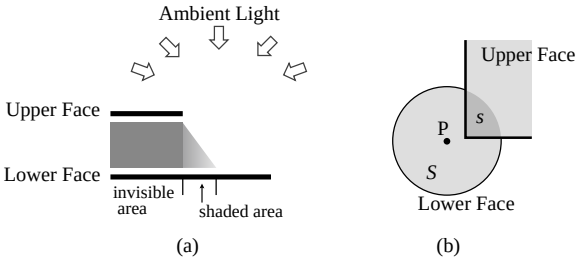

Figure 3: *(a) Lower face has the shaded area. (b) The intensity of ambient light at P is estimated by using s and S.*

6. Result

The rendering results of the proposed methods are shown in Fig.4 and Fig.5. The application was implemented using Java on a standard PC (CPU:Intel Core2 Duo 2.6GHz, RAM 3.5GB). The size of rendering area is 512x512 pixels. The weights w_a, w_b, and w_c, and RGB values r, g, and b are set to 0.675, 0.225, 0.15, 1.0, 1.0, and 0.95, respectively. These values are decided by a number of trials. In Fig.4, (1) and (2) are examples of simple Twist Fold and Origami Tessellation, respectively. Fig. 5(a) is an example of one of the best known Origami pieces, namely, the “crane”, and Fig. 5(b) shows an example in which the texture image is adopted. It can be seen that our method works well for rendering flat Origami regardless whether it has or does not have closed-loops in the overlap order of faces. The computational time for rendering Fig.5(2-c), (2-d), (2-e) were 63ms, 93ms and

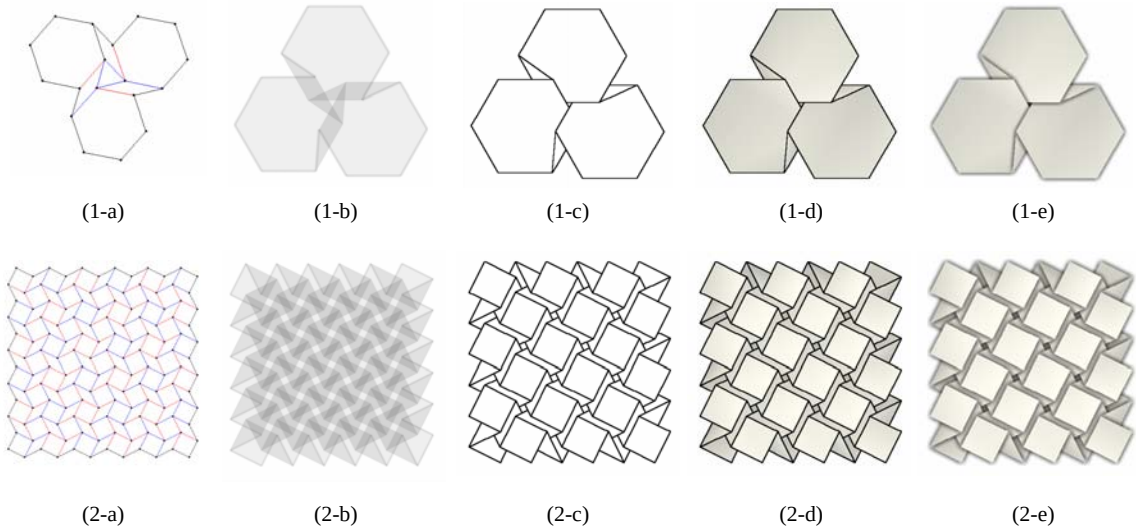

Figure 4: *Results. (a) Crease pattern. (b) Rendered translucently using Java2D API. (c) Line representation. (d) Illustration representation. (e) Shaded representation.*

1515ms respectively. The pseudo shading consumed much time because it requires the calculation of intensity of ambient light for all pixels.

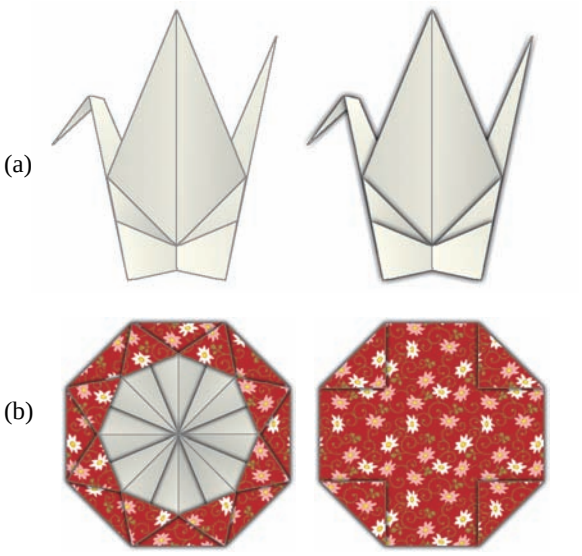

Figure 5: *(a) Example of the "crane". Left: Illustration representation. Right: Shaded representation. (b) Example of the "medal" rendered with texture.*

7. Conclusion and future research

We proposed new methods to appropriately render flat Origami pieces that have closed-loops in the overlap order of faces using a overlap relation matrix and a face ID buffer. It may be necessary to extend our method to make it possible to treat 3D Origami. Although we applied colors to vertices based on heuristics, it is possible to apply more appropriate colors based on physical simulation.

References

[Bat] BATEMAN A.: Paper mosaics. http://www.papermosaics.co.uk/.

[BH96] BERN M., HAYES B.: The complexity of flat origami. In *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms* (1996), pp. 175–183.

[DO07] DEMAINE E. D., O'ROURKE J.: *Geometric Folding Algorithms*. Cambridge University Press, 2007.

[FMF07] FURUTA Y., MITANI J., FUKUI Y.: Modeling and mouse interface for interactive virtual origami operation (in japanese). In *INTERACTION 2007* (2007), vol. 2007, pp. 137–144.

[Hul02] HULL T.: *Origami 3: Third International Meeting of Orgami Science, Mathematics, and Education Sponsored by Origami USA*. A K Peters Ltd, 2002.

[Lan] LANG R. J.: Tree maker. http://www.langorigami.com/science/treemaker/.

[LB99] LANGER M. S., BÜLTHOFF H. H.: *Perception of shape from shading on a cloudy day.* Tech. Rep. 17, Max-Planck-Institut für biologische Kybernetik, 1999.

[MYYT96] MIYAZAKI S., YASUDA T., YOKOI S., TORIWAKI J.: An origami playing simulator in the virtual space. *The Journal of Visualization and Computer Animation 7*, 1 (1996), 25–42.