PROCEEDINGS
13th INTERNATIONAL CONFERENCE ON GEOMETRY AND GRAPHICS
August 4-8, 2008, Dresden (Germany)
ISBN: 978-3-86780-042-6

THE FOLDED SHAPE RESTORATION AND THE RENDERING METHOD OF ORIGAMI FROM THE CREASE PATTERN

Jun Mitani
University of Tsukuba, JAPAN / PRESTO, JST, JAPAN

ABSTRACT: Recently, it is not uncommon to use a computer to research on Origami. We developed a dedicated editor, ORIPA, for designing the crease pattern of Origami intending to establish the basis of the future research on Origami. The editor has unique features that the folded shape of flat Origami (Origami which is folded flat) is restored from the crease pattern and the CG image of that can be generated. The users can input crease patterns by using a simple user interface. We expect that this editor can have beneficial influence on Origami researches. In this paper, we introduce the user interface of this system and the inner algorithms of rendering. Restoring the folded shape of Origami from the crease pattern can be replaced by the problem of finding valid overlap relations between all two faces. This relation can be represented by a matrix. To define the matrix is not a simple problem because it is proven that this problem belongs to a class of NP-complete. Although ORIPA finds the valid overlap relations by using brute-force approach, the answers are found in reasonable time due to the elimination of the invalid overlap relations at the early stage of estimation. After a valid overlap order of folded Origami is found, there is another problem; how to display it. In flat Origami, 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 because Origami is usually represented by a set of plane polygons of zero thickness as is generally used in CG and all faces are placed on the same plane. In the proposed method, we use the matrix that represents the overlap relation and a face ID buffer and 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. By using this buffer, we propose three rendering styles. Lastly, we introduce applications that use the exported data from ORIPA.

Keywords: Origami, Computer Graphics, Computational Geometry, Rendering

1. INTRODUCTION

Origami is known as a Japanese art and play that creates a variety of shapes by folding a square sheet of paper. The process of how to fold the sheet of paper to make the objective figure is challenging and compatible with geometrical methods. Therefore, many studies about Origami had been done in the field of mathematics. In 2006, an international conference 4OSME (The Fourth International Conference on Origami in Science, Mathematics, and Education) was held in Pasadena. As the name of the conference shows, Origami is considered as a topic of Science, Mathematics, and Education today. In 2007, a book that covers a huge number of studies about Origami was published by Demaine and O'Rourke[2]. A lot of theorems and their proofs related with Origami are introduced in the book. The fruits of the studies have been applied to recent engineering, such as designing foldable structures and education of geometry.

Recently, it is not unusual to use a computer for studying Origami. Some softwares have been designed that simulate folding paper or assist designing new works. But there are still not efficient Origami modeling tools that

can build Origami models in digital data which have information for both of structure and geometry, because directly inputting these data into a computer needs difficult process. So, we focused to input crease pattern first and then calculate the folded figures in a computer. With this approach, we can reduce a cost for modeling Origami structure. In this paper, we introduce a dedicated Origami Pattern Editor, named ORIPA. This editor has UI for inputting a crease pattern efficiently and features for estimating folded figures. ORIPA can discriminate whether the pattern can be folded into flat or not. When the pattern can be folded into flat, ORIPA outputs the rendered image. The ORIPA is released on the web[6]. So anyone can download and try to use. We hope this application could contribute toward future study of Origami.

We explain the crease patterns of Origami and the user interface of ORIPA in Section 2. In Section 3, we describe the matrix expression of overlap relation of faces. In Section 4, we describe a method for rendering. The results are shown in Section 5 and applications that use the exported data from ORIPA are introduced in Section 6. Our conclusions and future research are described in Section 7.

2. INPUT CREASE PATTERNS

In this section, we describe the crease pattern of Origami and the features of ORIPA for inputting crease patterns.

2.1 Crease patterns of Origami

The crease pattern of Origami is generated by folding a sheet of paper and consists of Mountain-fold lines and Valley-fold lines. Generally, folding is done by referring symbols (points or lines on the crease pattern). The ways of the fold is clarified as 7 classes by Fujita and Hatori as follows (Huzita's axioms and Hatori's addition [2][4]).

(1) Given two points p_1 and p_2, we can fold a line connecting them.
(2) Given two points p_1 and p_2, we can fold p_1 onto p_2.
(3) Given two lines l_1 and l_2, we can fold line l_1 onto l_2.
(4) Given as point p_1 and a line l_1, we can make a fold perpendicular to l_1 passing through the point p_1.
(5) Given two points p_1 and p_2 and a line l_1, we can make a fold that places p_1 onto l_1 and passes through the point p_2.
(6) Given two points p_1 and p_2 and two lines l1 and l2, we can make a fold that places p_1 onto line l_1 and places p_2 onto line l_2.
(7) Given a points p_1 and two lines l1 and l_2, we can make a fold perpendicular to l_2 that places p_1 onto line l_1.

Figure 1 shows diagrams of corresponding operations.

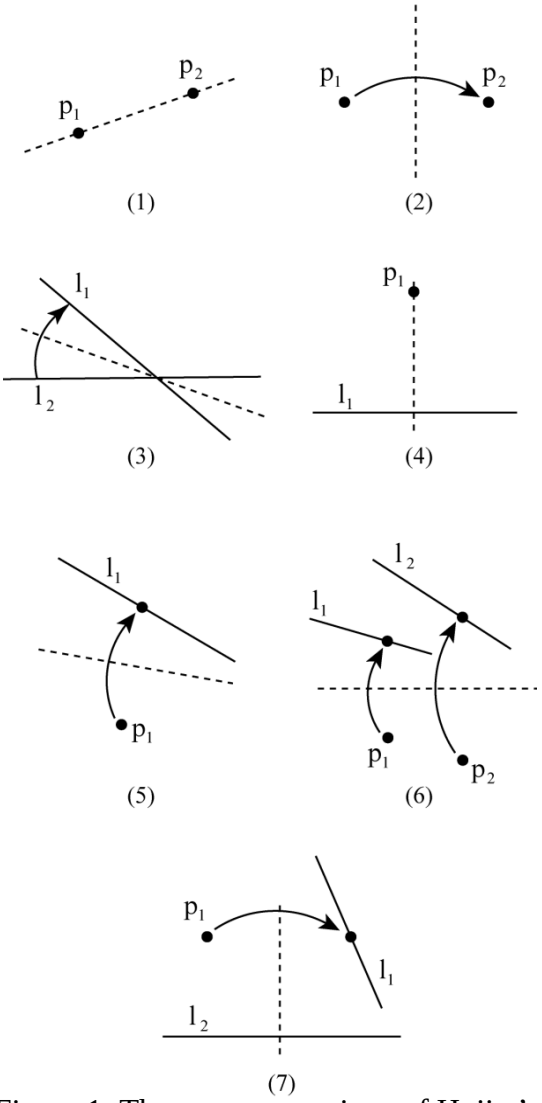


Figure 1: The seven operations of Hujita's axioms and Hatori's addition.

2.2 User interface

In the seven operations of folding, from (1) to (4) are commonly used for folding general Origami. Instead, (5), (6), and (7) are rare. We designed the user interface of ORIPA so that the user can easily input the general patterns appeared by using the folding from (1) to (4). And considering the patterns well appeared in the common Origami, we implemented the following 9 commands to ORIPA.

1. A bisector of a specified angle.
2. A perpendicular bisector of two specified points.
3. A line that connects two specified points.
4. A line that passes two specified points.
5. A line that passes a specified point and perpendicular of a specified line.
6. Three lines that connect specified three points and the incenter of the triangle.
7. Lines that locate symmetric position against specified lines.
8. A line that connects two specified points.
9. A line that has specified length and angle.

With these commands, user can input crease patterns more efficiently than using standard 2D drawing tools. Figure 2 is the main Window of ORIPA and the crease pattern of *Crane* inputted by using the above commands.

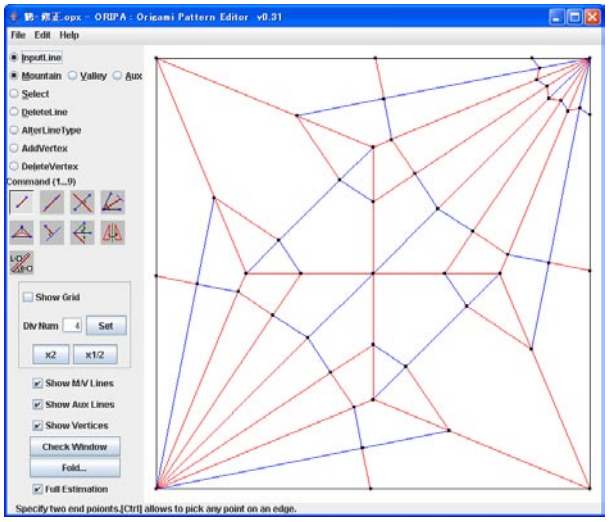


Figure 2: Application window of ORIPA with the crease pattern of *Crane*.

2.3 Validity check for flat Origami

A crease pattern inputted by a user sometimes may invalid for flat Origami. This means that Origami may not be folded into flat even though that is folded following the pattern. The conditions of a crease pattern around a single vertex in flat Origami should satisfy the following conditions [2].

- The number of creases is even.
- The number of mountains and the number of valleys differ by ±2.
- The sum of the alternate angles about the vertex is π.

All inner vertices have to satisfy the above conditions. If one or more vertices exist that do not satisfy them, the pattern cannot be folded into flat. ORIPA checks the conditions for each vertex and highlights the invalid vertices to notify the users.

3. OVERLAP RELATION OF FACES

After a valid crease pattern is inputted, ORIPA estimates the folded figure by calculating the locations of all polygonal faces in the crease pattern and the overlap relation of them. In this section we describe a matrix that represents the overlap relations of faces.

3.1 Matrix expression of overlap relation

The order of overlapping of faces can be serially defined when no closed loops exist. But when closed-loops exist, it cannot. For example, as a “Twist Fold” (See Figure 3) has a closed loop of overlapping order of faces, we cannot say which face is located lowest or uppermost. The face (A) is on (B), (B) is on (C), (C) is on (D), and (D) is on (A) in the folded shape in Figure 3. When the number of polygonal faces included in the Origami piece is N, an N x N matrix can describe all overlap relations. We refer to this matrix as the *OR matrix* hereinafter. Each element m_{ij} of this matrix 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 Figure 4 (a) and (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 matrices can be defined from a single crease pattern.)

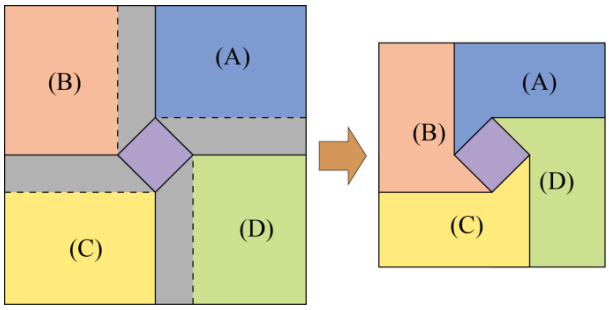


Figure 3: The crease pattern and the folded shape of a “Twist fold” (Dash lines and solid lines in the crease pattern are valley and mountain fold respectively).

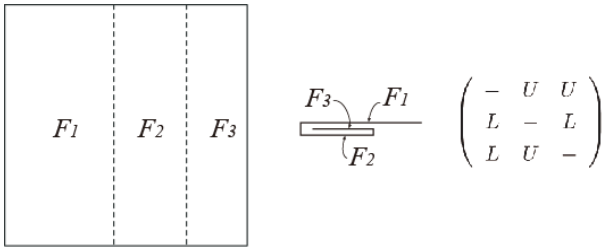


Figure 4: A simple crease pattern (a), side view of the folded one (b) and the OR matrix (c).

3.2 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[1]. Although determining every element of the OR matrix from a crease pattern is not easy, it can be obtained from a brute-force approach because the number of possible cases is finite. In the present paper, we do not discuss the detail of finding valid overlap relations. Although ORIPA finds the valid overlap relations by using brute-force approach, the answers can be found in reasonable time due to the elimination of the invalid overlap relations at the early stage of estimation. ORIPA divides each polygonal face in the folded shape into sub-faces, then the sub-faces are grouped so that the overlapping order of them can be defined serially. After valid overlap orders in each group are extracted, valid order matrixes (no contradictions arise for all groups) are searched.

4. RENDERING

It is sometimes difficult to display the shape of flat Origami correctly on the screen when the 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. The painter's algorithm (also known as 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. 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 because all faces have the same depth (z-value) in flat Origami. In this section, we propose a new rendering technique to solve this problem.

4.1 Render to face ID buffer

Here, we describe how to render folded Origami based on the OR matrix estimated from the crease pattern. 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 hereafter 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.)

4.2 Line style rendering

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

4.3 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 the common rendering method is used, 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 easily to be understood using a new approach based on the heuristics.

As a generally experienced rule, folded lines in the vicinity of the valley become dark because little light reaches this area. On the other hand, folded lines in the vicinity of the peak become bright. Then, we set the brightness B to each vertex of polygonal faces according to the value $M - V$, 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.

The color of each pixel in a face is calculated by linearly interpolating the colors of vertices on the contour. After that, we add gradation to make the results natural.

4.4 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[5]. In the method of Ambient occlusion, the intensity of ambient light is adjusted according to the ratio of the size of a shield object on a sphere that is centered at the target position. For flat Origami, we only need to consider blocking by upper faces. Therefore, we use the model in which the intensity of ambient light at a point in the shaded area (Figure 5(a)) is linearly reduced with the ratio of the size of the blocking object in a circle centered at the position, as shown in the following equation:

$$I = 1 - \frac{s}{S}$$

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 Figure 5(b).

The value I is calculated for each pixcel and the blightness of the pixcel is multiplied by this.

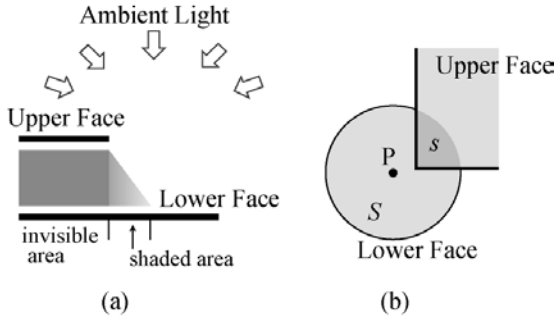


Figure 5: (a) Lower face has the shaded area. (b) The intensity of ambient light at P is estimated by using s and S.

5. RESULT

The features mentioned in previous sections were implemented onto ORIPA. ORIPA has 4 windows as shown in Figure 6. They are the designing window with which users input a crease pattern (a), the validity check window which notices invalid creases when they cannot be folded into flat (b), the folded shape display window which displays the folded shape with transparency (c), and the rendering window which shows the rendered result using our method mentioned in the previous section (d). The application was developed by using Java on a standard PC. Figure 7 is an example of simple Twist Fold which has a

closed-loop in overlap order. (a) is the crease pattern, (b) is the folded shapes rendered with line style, (c) is illustration representation, and (d) is shaded representation. Figure 8(a) is an example of one of the best known Origami pieces, namely, the crane, of which crease pattern is shown in Figure 2 and Figure 8(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 of whether it has closed-loops or not in the overlap order of faces.

6. APPLICATION

ORIPA can export not only the data of inputted crease pattern but also the folded shapes. Here we introduce some applications developed by others that use the data. Figure 9(a) shows an application that simulates cutting of the folded Origami and shows the cross lines on the pattern. Figure 9(b) shows an application[7] that simulates rigid folding from the crease pattern. Figure 9(c) shows an application[3] in that user can open the folded model in virtual 3D.

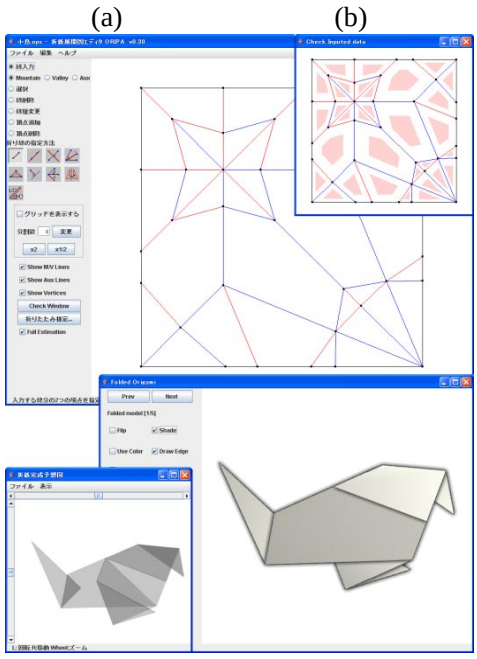


Figure 6: ORIPA's windows. (a) Designing window, (b) validity check window, (c) folded shape display window, and (d) rendering window.

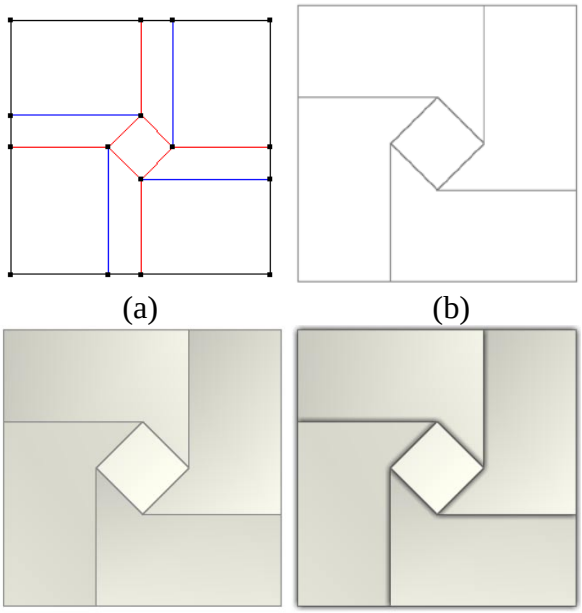


Figure 7: Results. (a) Crease pattern. (b) Rendered translucently using Java2D API. (c) Line representation. (d) Illustration representation. (e) Shaded representation.

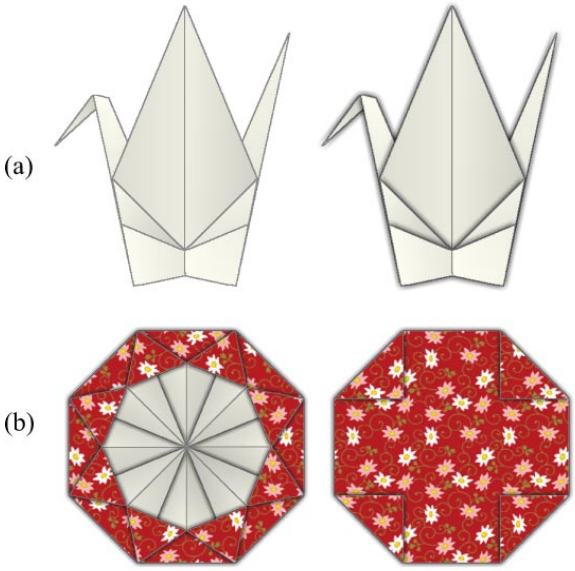
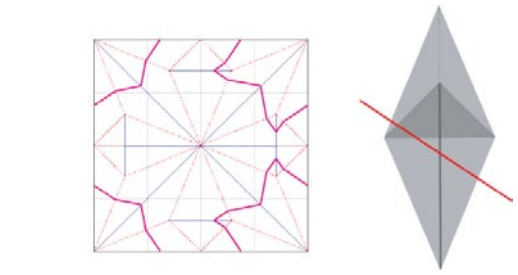
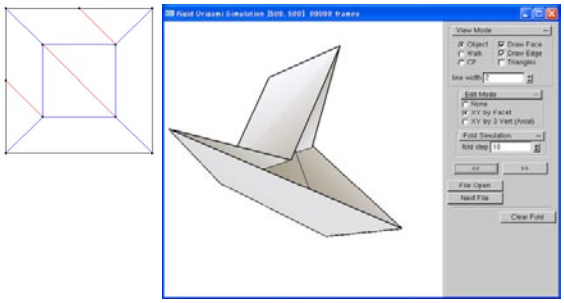


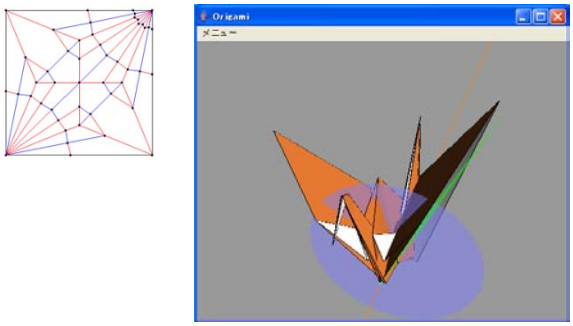
Figure 8: (a) Example of the *crane*. Left: Illustration representation. Right: Shaded representation. (b) Example of the *medal* rendered with texture.



(a)



(b)



(c)

Figure 9: Applications. (a) Simulation of cutting a folded Origami. (b) Simulation of Rigid Origami[7]. (c) Interactive Origami folding on a PC[3].

7. CONCLUSION AND FUTURE WORK

We proposed methods for inputting the crease patterns, representing the overlap relations of faces using a matrix, and rendering the folded shape. We implemented these features and developed a dedicated editor, ORIPA. ORIPA can export not only the crease pattern inputted by the user but also the folded shape and overlap relations. It means the data contains full information about the piece of Origami. Anyone who makes the importer of the data can rebuild the model of Origami in his/her application with this ORIPA. We hope our application could contribute to future study of Origami. In future, we aim to add features that enable users to edit crease pattern and to see the folded shape at the same time so that ORIPA could assist designing process of a new work.

REFERENCES

[1] Bern, M., and Hayes, B.: The complexity of flat origami. *In Proc. of the seventh annual ACM-SIAM symposium on Discrete algorithms*, pp. 175–183, 1996.

[2] Demaine, E., and O'Rourke, J., Geometric folding algorithms. Cambridge university press, 2007.

[3] Furuta, Y., Kimoto, H., Mitani, J., and Fukui, Y. Computer Model and Mouse Interface for Interactive Virtual Origami Operation. *IPSJ Journal*, Vol.48, No.12, pp.3658-3669, 2007.

[4] Lnag, J. R., Huzita Axioms. http://www.langorigami.com/science/hha/hha.php4

[5] Langer, M. S., and Bülthoff, H. H. Perception of shape from shading on a cloudy day. Tech. Rep. 17, Max-Planck-Institut für biologische Kybernetik, 1999.

[6] Mitani, J. ORIPA; Origami Pattern Editor. http://mitani.cs.tsukuba.ac.jp/pukiwiki-oripa/.

[7] Tachi, T. Simulation of Rigid Origami. In Proc. of 4OSME, 2006.

ABOUT THE AUTHORS

1. Jun Mitani, Ph.D. is a lecturer at the Department of Computer Science at the University of Tsukuba, Japan, and a researcher of the PRESTO, Japan Science and Technology Agency. His research interests are Computer Graphics and Geometric Modeling. He can be reached by e-mail: mitani@cs.tsukuba.ac.jp or through postal address: 1-1-1 Tennohdai, Tsukuba 305-8573, Japan.