A Computer-Aided Design System for Origamic Architecture

Jyun-Ming Chen, Yu-Zhi Zhang
Department of Computer Science and Engineering, Tatung University, Taiwan
jmchen@ttu.edu.tw, caa1211@hotmail.com

ABSTRACT

An Origamic Architecture (OA) is a paper craft form that creates a pop-up three-dimensional structure as it is opened. It is similar to a "pop-up story book", but it is made by cutting a single piece of paper. Because of this limitation, designing an OA requires considerable experience. This research proposes a computerized method to assists the design of OAs.

An OA is modeled by a set of planar polygons. , and our method is the application of polygons clipping operations on the unfolded pattern to guarantee that the model can be made from a single sheet of paper. We also present methods for checking the model's validity and displaying folding animation.

Keywords: Origamic Architecture, Pop-Up Card, Computer-Aided Design, Paper Crafts.

1: INTRODUCTIONS

Origamic Architecture (OA) is an amazing paper art. By cutting and folding, a form jumps out of a piece of paper in an incredible 3D form. The concept for OA was devised by M. Chatani in the early 1980s. He has produced many books of patterns and pictures of his work [1,2].

OA comes in three types, differentiated by the angle of opening: 90, 180 and 360 degrees. The 90-degree models are the most common type of OA. No paper gluing and pasting is required. This paper focuses on the computer based design methods of such OAs and study the corresponding pop-up theory.

Designing OAs not only needs creative minds but also requires engineering skills. Required skills include the method to accomplish flat foldability of paper mechanisms when pages are closed. Specialists who design and create such mechanisms are known as paper engineers. The design in pop-up crafting has so far been primarily manual and based on traditional crafting methods.

Due to the extensive manual process, the method of trials and errors was particularly evident in the design stage. Trials and errors are common in the sizing of the sketches, and folding the paper pieces. A misfit of the pop-ups on the pages would result in repeated work. As such, OA making can be labor intensive, time consuming and burdened with repetitive tasks, especially for the less experienced pop-up enthusiasts.

Books [1,2] had been published to teach novices pop-up constructions by following step-by-step instructions with templates and illustrations, but the methods of handling OA in computers have not been well studied, and we could find few related works. Chatani first demonstrated CG animation of OA [2], but the data are pre-calculated and hard-coded in the program. Some works study the design of pop-up cards with computer [3,4,8]. These are all for the 180° types, and these techniques cannot be adapted to the design of the OA we aim for.

J. Mitani and H. Suzuki proposed a method for designing OA using a voxel data structure [5]. This enables interactive design of OA and easy generation of the unfolded pattern by using the characteristics of voxel representation (Fig. 1). However, the characteristics of voxel representation limits the OA design to be rectilinear.

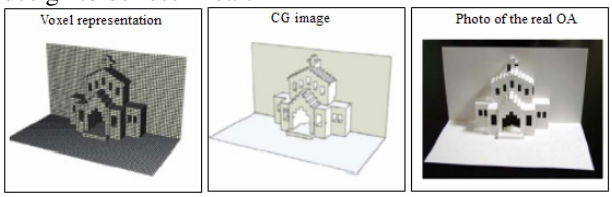

Figure 1 OA designed using voxel representation [5].

J. Mitani and H. Suzuki also proposed a new method [6]. It uses a set of planar polygons to facilitate the design a wider variety of OA. The designer is required to input the outline of all the new component faces, both the vertical faces and horizontal faces. However, the size and position of each horizontal face in an OA model have some restrictions. The complexity of the system discourages a novice design from learning and using the system. Also, the issue of incorporating smooth-contour faces is unresolved.

In this paper, we also use a set of planar polygons to build the OA models, but the method we propose only requires the designer to input the contours of vertical faces. The corresponding horizontal faces are created by the system automatically. Furthermore, we use the Chaikin's curve to incorporate smooth-contoured faces and openings.

2: BASIC THEORY OF OA

2.1: Vertical and Horizontal Faces

In a 90° OA, all faces are parallel to either the bottom face, or the back face. These two types of faces are hence called horizontal faces (short for HFace) and vertical faces (VFace) respectively, as in Fig. 2.

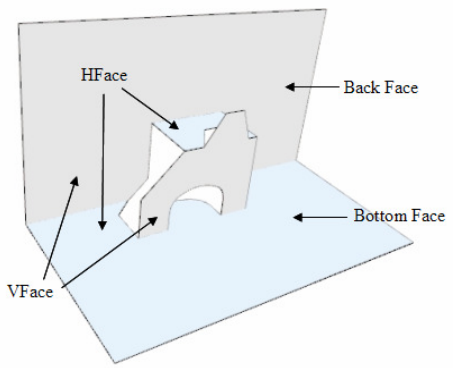

Figure 2 VFace and HFace.

2.2: Model Coordinate and Pattern Coordinate

The coordinate systems of the three-dimensional model and two-dimensional pattern are bijectively related. In the following, we will use the term "model coordinate system" and "pattern coordinate system" to address the two models respectively.

Refer to Fig. 3 for the following discussion. Given a distance t_v between any VFace and the Back Face and a distance t_h between HFace and the Bottom Face, we can convert between the two coordinate system as follows.

Model Coordinate(3D → Pattern Coordinate(2D) conversion

$$\begin{cases} X_p = X_m \\ Y_p = Z_m + Y_m \end{cases} \tag{1}$$

Pattern Coordinate(2D) → Model Coordinate(3D) conversion

VFace:

$$\begin{cases} X_m = X_p \\ Y_m = -t_v \\ Z_m = Y_p + t_v \end{cases} \tag{2}$$

HFace:

$$\begin{cases} X_m = X_p \\ Y_m = Y_p - t_h \\ Z_m = t_h \end{cases} \tag{3}$$

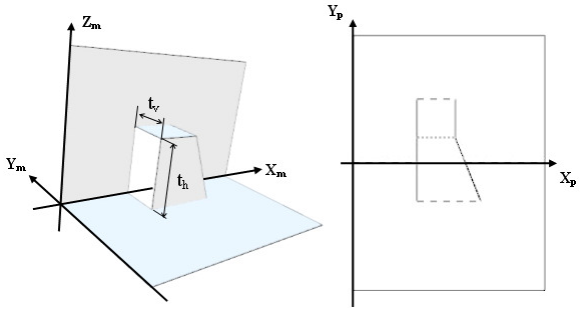

Figure 3 Model coordinate and pattern coordinate.

.

2.4: Integrity Constraint

In the OA design process, the HFaces and VFaces are generated from Back Face and Bottom Face by cutting and folding. As a result, the following two constraints must be satisfied for a valid pattern (Fig. 4):

$$\begin{aligned} & F_i \cap^* F_j = \phi, \text{if } i \neq j \\ & F_1 \cup^* F_2 \cup^* \ldots \cup^* F_n = S \end{aligned} \tag{4}$$

where S is the original sheet of paper and F_i's are the polygons comprising the OA design. The Fi's are termed OAFaces hereafter.

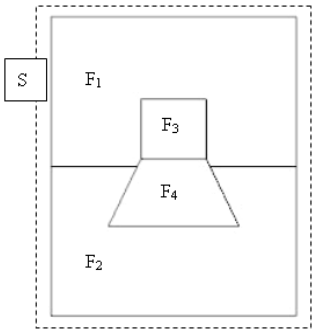

Figure 4 Polygons that construct OA pattern.

During the OA design, many polygons will be constructed and this constraint must be maintained at all times. A two-dimensional polygonal Boolean engine, providing the operations of regularized union ($\cup^*$), intersection ($\cap^*$) and difference ($-^*$), is use to ensure the model integrity. In our prototype system, the General Polygon Clipping Library (GPC) [7] by A. Murta is used.

3: COMPUTER AIDED DESIGN FOR OA

In the previous section, we have discussed the basic theory of OA. Section 3.1 gives an overview of the interactive computerized OA design interface. Sections 3.2 and 3.3 introduce the face building functions in the interface. Finally, a method to generate the unfolded pattern is discussed in Sec. 3.4. Using this pattern, the designer can realize the design in a physical paper model.

3.1: User Interface

Traditionally, an OA is designed by trial and error. The designer had to sketch the unfolded pattern by hand, and guessed how the 3D structure would appear when they are opened. To address this problem, we propose an interactive interface to enable users to design 3D figures intuitively by viewing 3D graphics on a computer display.

The designer sketches a contour and chooses a Face Function to decide what kind of the face will be added to the OA model. The steps of the interface we propose are described as follows：

1. Initialize an OA containing only a Back Face and a Bottom Face (Fig. 5a).
2. Determine the depth of the edit plane and sketch the contour (Fig. 5b).

3. A VFace is generated according to the input contour. The system also generates a supporting HFace automatically (Fig. 5c) It should be noted that as new faces are generated, the previously constructed OAFaces are modified to maintain the integrity constraint.
4. Repeat steps 2 and 3 until the desired OA is achieved.

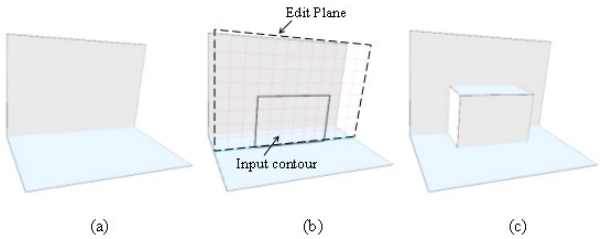

Figure 5 Interface for OA design.

3.2: Add a Face

When the user completes a contour in 3D model coordinate, the system will convert it to 2D pattern coordinates by applying Eq. (1) in Sec. 2.4. The system then adds new faces to the OA model. It should be noted that the system automatically maintains the conditions for pattern generation (Sec. 2.4) by modifying the OA patterns via the underlying Boolean engine. This enables to the designer to focus on the main features of the design, without worrying the HFaces.

To achieve this, the system performs the following operations in step 3 of the interface proposed previously.

3a. Generate a new OAFace according to the input contour, and update all the existing OAFaces in the model by applying the following equation, where *i* is the index of OAFaces.

$$OAFace(i) = OAFace(i) -^{*} NewFace \quad (4)$$

3b. Add the NewFace to the model.

3.3: Face Function

When a contour is sketched by user, system will create a VFace and a set of associative HFaces. It will convert coordinate values of the input contour vertices from 3D to 2D by Eq. (1), and then store it to be a new VFace.

Because HFaces must support VFace pop-up, there are some restrictions to the lengths and positions of them. For the sake of design convenience, we propose a method to let the HFace be created automatically.

For an OA model to be flatly folded when it is closed, the fold line between the HFace and the VFace must be parallel to the pattern coordinate x-axis. Therefore, we collect all this kind of edges and they should be the bases for creating HFace.

After adjusting all the contours of the VFaces in clockwise direction, the edges whose end points are from left to right and parallel to the x-axis are named *Upper*, and the edges whose end points are from right to left and parallel the x-axis are named *Lower*, as in Fig. 6.

System will produce an HFace on each Upper edge, and each length of the new HFace is the depth of the input contour, such as in Fig. 7. Then they will be added to the model by using the method in Sec. 3.2, as in Fig. 8.

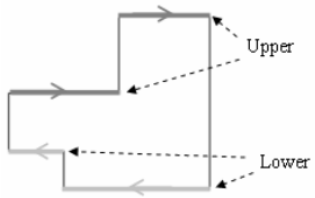

Figure 6 Classify Upper and Lower.

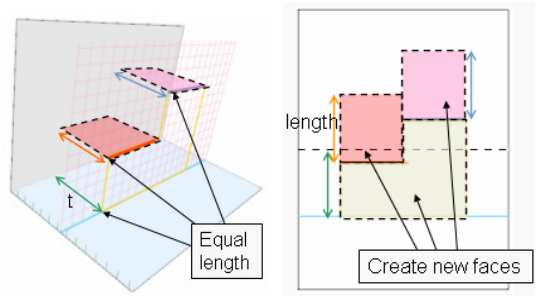

Figure 7 Create the new VFace and HFaces.

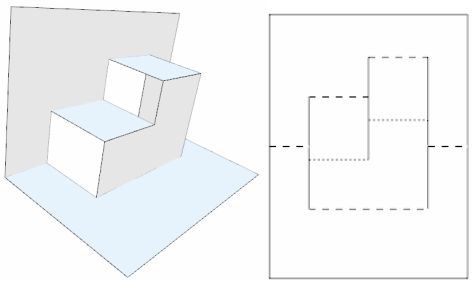

Figure 8 Result of the face function.

However, if the new VFace and HFaces cover one another, the sequence of adding faces will produce different results, such as in Fig. 9. Some of them are not desirable according to common design intention (Fig. 9.c). This issue has been discussed this in detail in [9] and will be published in another article.

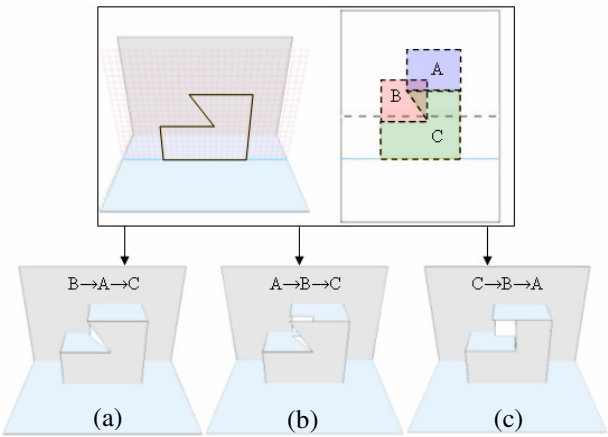

Figure 9 The different results of creating HFace.

3.4: Unfolded Pattern Generation

The unfolded pattern can be easily generated by outputting the contours of each face, using pattern coordinate values, on a sheet of paper. The contouring edges are further classified into three types as follows :

- A line is a mountain line if it is parallel to the x-axis, shared by the same VFace and HFace both in pattern coordinate and model coordinate.
- A line is a valley line if it is parallel to the x-axis and shared by the same VFace and HFace both in pattern coordinate and model coordinate [9].
- All other lines are cut lines.

Figure 10 shows the different drawing styles used to aid the designers in realizing the paper model.

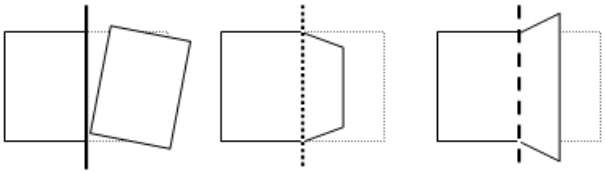

Figure 10 Cut (solid), mountain (dotted) and valley (dashed) lines.

4: CHECK FOR POP-UP CONDITION

In Sec. 3, we propose a method for OA design using 3D computer graphic. Although this method is intuitive and convenient, we cannot guarantee the OA model designed can satisfy the pop-up condition. In this section, we will discuss the basic theory of pop-up and propose a method for testing the pop-up condition.

4.1: The Basic Theory of Pop-Up Condition

The pop-up condition is due to the driving force act upon the Back Face, we classify the force sent from the driving force and act upon the each VFace into two types：

1. Direct communication (Fig. 11) ：When the Back Face is pulled to unfold, each VFace is due to the pull to unfold equally, and when the Back Face is pushed to fold, each VFace is also pushed to fold equally.
2. Indirect communication (Fig. 12) ：When the Back Face is pulled to unfold, each VFace is due to the push to unfold contrary, and when the Back Face is pushed to fold, each VFace is pulled to fold contrary.

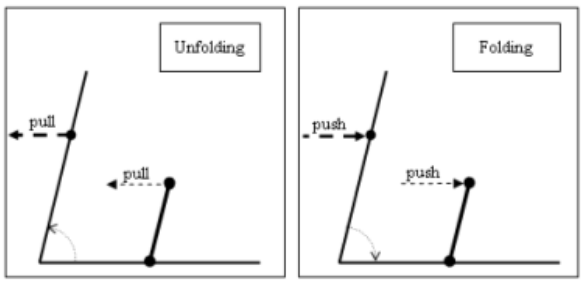

Figure 11 Direct communication.

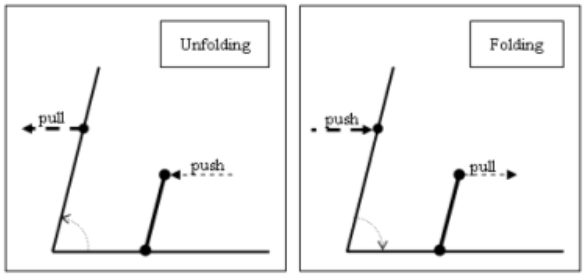

Figure 12 Indirect communication.

We can infer from basic pop-up structures that if a face satisfies following two conditions, it can pop up by direct communication:

1. **Right position:** A face can find one connective face, and can form a parallelogram in the side view by stretching them to Bottom Face and Back Face, such as Fig. 13a.
2. **Connectivity:** A face can find at least two ways connection to the two faces which already can pop up through by other faces in the right position, one of way directions is up or back, and another is down or front, such as Fig. 13b.

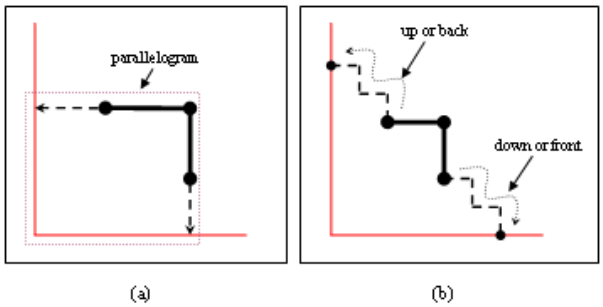

Figure 13 The pop-up theory for direct communication: Right position and connectivity.

We infer the other basic structures which can pop-up by indirect communication that if a face satisfies following condition, it also can pop up:

Surround: A face can find one connective face and they can form a parallelogram by connecting the two faces already can pop up, as Fig. 14.

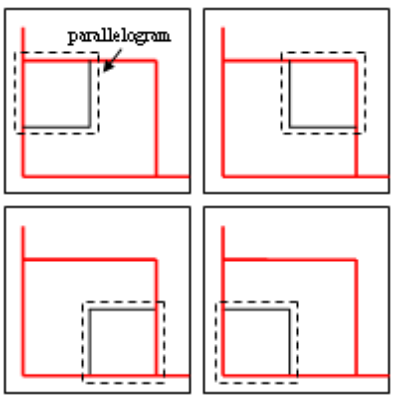

Figure 14 The pop-up theory for indirect communication: Surround.

4.2: Criteria for Pop-Up

First, initialize a set of faces to contain only the Back Face, we name it Back_Set. Collect connective faces forth or down to Back_Set, as in Fig. 15a, so that each the face in Back_Set at least has a fold line can connect to Back Face by passing up or back connective faces.

Then, initialize a set of faces to contain only the Bottom Face, we name it Bottom_Set. Collect connective faces up or back to Bottom_Set, as in Fig. 15b, so that each the face in Bottom_Set at least has a fold line can connect to Bottom Face by passing down or forth connective faces.

Further, we collect the faces both in Back_Set and Bottom_Set to Share_Set, as in Fig. 16. So that all the faces in Share_Set should satisfy the connectivity condition, and they can exactly satisfy the pop-up condition too. So far the faces in Share_Set will satisfy the right position and connectivity conditions.

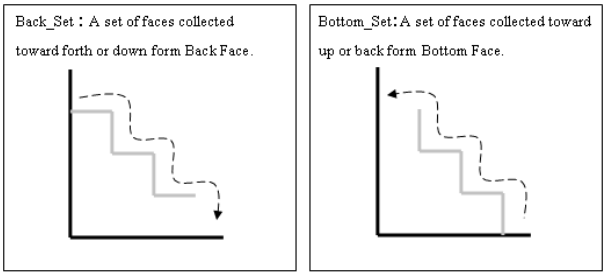

Figure 15 Back_Set and Bottom_Set.

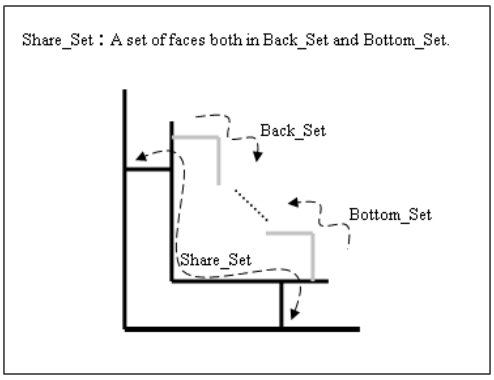

Figure 16 Recursively collect Share_Set.

In addition, we find the indirect face, which is a face has at least two fold lines, one of which connects to Share_Set and another connects to Share_Set by passing through a connective face, as shown in Fig. 17. Although it may not be collected to previous Share_Set, it exactly can pop up by pop-up theory proposed in 4.1. After we collect Share_Set every times, the system will check existence of indirect faces automatically, and then collect them in Share_Set. So that the faces in Share_Set will also satisfy the surround condition.

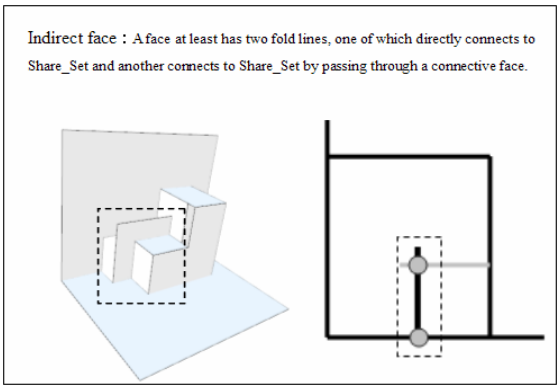

Figure 17 Indirect face.

Afterward we use the faces in Share_Set to collect Back_Set and Bottom_Set again. The above process is the recursive phase until there are no more faces to be collected.

After recursive phase, we name the face which only have one fold line and can connect to Share_Set ornamental face, as in Fig. 18, and add it in Share_Set. Even through ornamental faces can not help other faces pop up, they usually are used to be ornamental with real OA, such as openings or pulls in Fig. 19. So we define the ornamental faces are a kind of legal faces, and add it in Share_Set. Finally, the faces in the model but not in Share_Set are illegal, and they will be high light in our system. The flow chart of this judgment algorithm is illustrated in Fig. 20.

Fig. 21 shows four examples of applying this criteria, where the left pattern shows the illegal faces, the middle pattern is the Back_Set and the right pattern is the Bottom_Set.

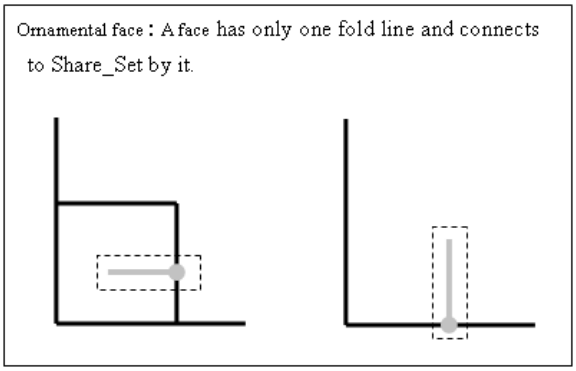

Figure 18 Ornamental face.

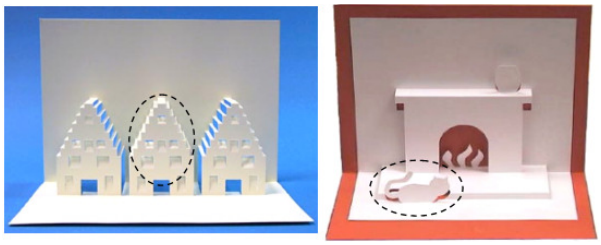

Figure 19 Samples of ornamental faces (designed by M. Chatani).

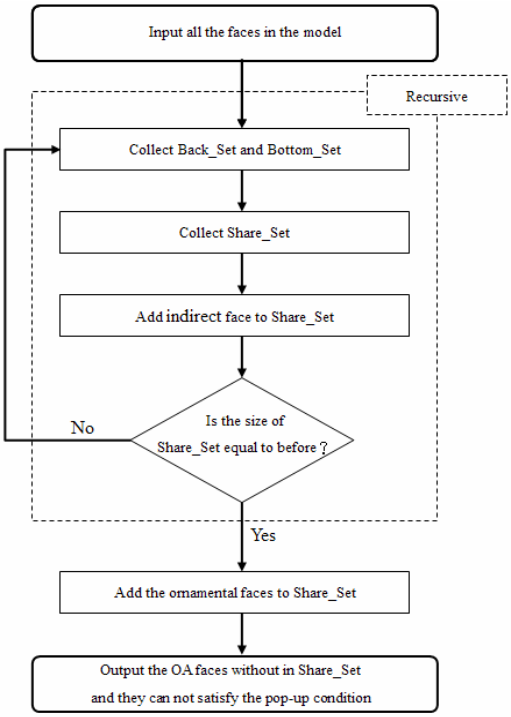

Figure 20 The flow char of judgment for the pop-up condition.

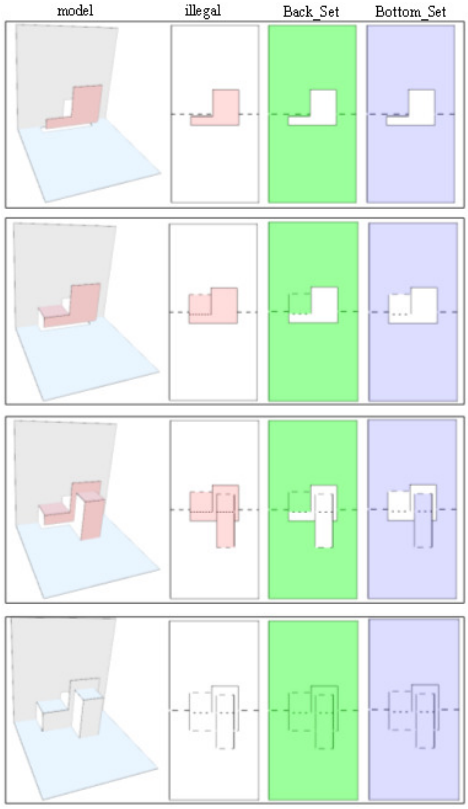

Figure 21 Judgment for the pop-up condition.

5: RESULT

A prototypical OA design system has been implemented and several interesting models have been constructed. We have implemented our method on a PC and used this implementation to design some OA models. There is a example Fig. 22 (left up is the CG image, left down is the pattern and right is the photograph).

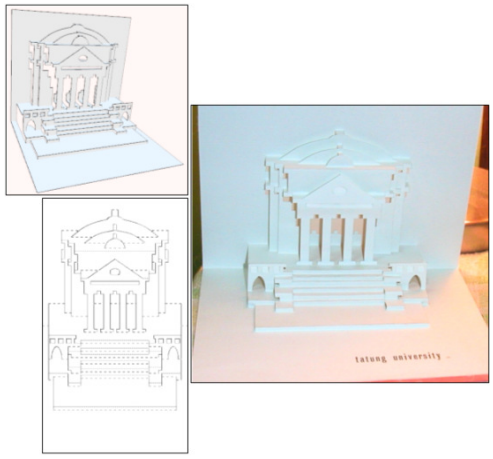

Figure 22 Shan-Chi Hall of Tatung University.

6: CONCLUSION AND FUTURE WORK

In this research, the interface allows users design 3D model intuitively, it can aid users do not need to imagine the open structure of OA. And the judgment for pop-up condition can reduce many minute survey processes, trials and errors. By using our method and a little imagination, everybody can create many wonder productions.

Although we have proposed a useful method for determining the pop-up condition of 90-degree OA, it is only a kind of conjecture. In other words, we ensure if this method judges the face legal, it exactly can pop up but we are not sure this method can find all the pop-up faces. Because rotary motions of faces in an OA may be produced by the elasticity of paper, a powerful judgment should consider more mechanical reasons.

REFERENCES

[1] M. Chatani, "Origamic Architecture of Masahiro Chatani", Shokokusya, Tokyo, 1984, *in Japanese*.

[2] M. Chatani, S. Nakamura, and N. Ando, "Practice of Origamic Architecture and Origami with Personal Computer", Kodansya, Tokyo, 1987, *in Japanese*.

[3] A. Glassner, "Interactive Pop-up Card Design, Part 1", IEEE Computer Graphics and Applications, 22(1):79-86, 2002.

[4] A. Glassner, "Interactive Pop-up Card Design, Part 2", IEEE Computer Graphics and Applications, 22(2):74-85, 2002.

[5] J. Mitani and H. Suzuki, "Computer aided design for origamic architecture models with voxel data structure", Intellectual Property and Social Justice (IPSJ), 44(5) 1372-1379, 2003.

[6] J. Mitani and H. Suzuki, "Computer aided design for origamic architecture models with polygonal representation data structure", Computer Graphics International (CGI'04), 1530-1052, 2004.

[7] A. Murta, "General Polygon Clipper Homepage", http://www.cs.man.ac.uk/~toby/alan/software/#download, 2005.

[8] Y. T. Lee, S. B. Tor, and E. L. Soo, "Mathematical modeling and simulation of Pop-up books", Computers & Graphics, 20(1):21-31, 1996.

[9] Y. Z. Zhang, J. M. Chen, "A Computer-Aided Design System for Origamic Architecture"; Master's Thesis, Tatung University, 22-28, 2006.