A CAD System for Diagramming Origami with Prediction of Folding Processes

Naoya Tsuruta, Jun Mitani, Yoshihiro Kanamori,
and Yukio Fukui

1 Introduction

In recent years, many methods for designing origami pieces based on mathematical theories have been developed. One of the most powerful approaches has been implemented in the software system TreeMaker [Lang 06]. Although origami pieces designed using this software are restricted to a particular structure referred to as a uniaxial origami base, many complex and realistic works of art have been created. Describing the sequence of folds used to create a complex work is difficult. However, traditional origami diagrams, namely, a sequence of step-by-step illustrations, are still used, even today.

One problem in using such diagrams is that drawing diagrams manually is time consuming; as an origami piece becomes more complex, the cost for drawing the diagrams increases. Traditional origami pieces, such as a crane, are folded by approximately a dozen steps. But, many recent complex pieces need more than 100 steps, thus requiring more than 100 diagrams each to explain how to fold them. To avoid this cost, a crease pattern can sometimes be used instead of a diagram. A *crease pattern* consists of line segments that show the crease lines that appear on the paper when an origami piece is unfolded. Although drawing crease patterns is much easier than drawing diagrams, folding from crease patterns is more difficult for nonexperts because crease patterns do not contain any proce-

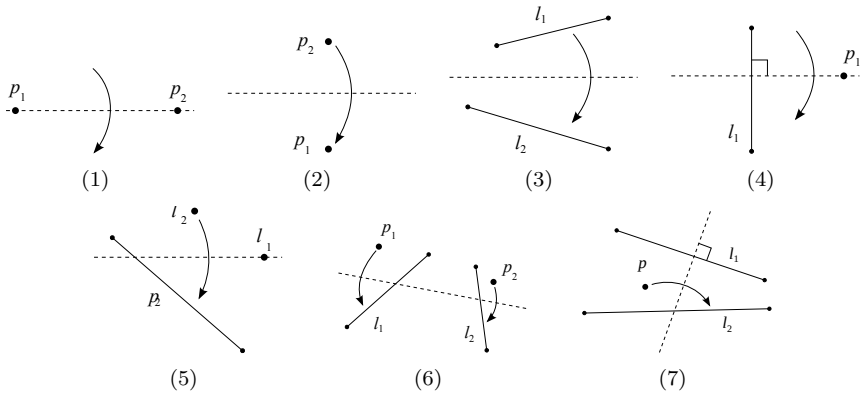


Figure 1. Huzita-Hatori (Huzita-Justin) axioms.

dural information. Therefore, for nonexperts, origami diagrams are still generally required.

In this paper, we propose a computer-aided design (CAD) system that alleviates the problem of drawing diagrams of flat-foldable origami pieces. Our system predicts possible candidate folds that can occur in the current state of an origami piece and lists these candidates. This prediction is made based on the axioms of the Huzita-Hatori (Huzita-Justin) axioms [Hatori 10], a series of seven basic methods for folding a sheet of paper by referring to points and lines (Figure 1). All possible folding lines are generated by applying these axioms in our system, and then folded shapes are calculated and listed as candidates for the next step. When the user selects one of these candidates, the shape is automatically appended to the origami diagram as the latest step. To moderate the explosion of the number of candidates and to simplify the implementation of the system, we limit the axioms to the first four and limit the types of folding to valley folds only. Since the axioms we use for the prediction fold one line at a time, the list of candidates does not include some major folds that require multiple noncollinear lines to be folded simultaneously, such as rabbit-ear-folds and petal-folds. Although this might be a limitation of our system, we provide a user interface for specifying folding manually when the intended folds are not among the candidates. Details of the prediction feature are discussed in Section 3.

We also show the result of enumerating origami pieces folded by applying Huzita-Hatori (Huzita-Justin) axioms multiple times using the prediction feature recursively. The result of this enumeration suggests the possibility of discovering new interesting origami pieces designed by a computer.

2 Related Work

Miyazaki developed an origami simulator with which a user could fold a simple origami piece on a PC by using standard mouse and keyboard devices [Miyazaki et al. 96]. Although this simulator was groundbreaking, it did not facilitate the production of origami diagrams. The Foldinator [Szinger 02] and the eGami [Fastag 09] systems are dedicated origami simulators that were developed to reduce the cost of drawing by hand. A series of figures are generated by simply specifying lines where a piece of origami is folded step by step. Moreover, these programs automatically add symbols, such as arrows, that are needed to explain how to form the fold. We believe the prediction function proposed in this paper is also applicable to these systems.

As mentioned in the previous section, because drawing a diagram for a complex origami piece is difficult, sometimes just a crease pattern is used. Mitani focused on drawing crease patterns and developed an origami pattern editor (ORIPA) [Mitani 08] that allows users to design crease patterns quickly. This editor can also calculate the shape of a folded origami piece from the crease pattern.

Our system is not the first CAD system to implement predictions. Although not related to origami, Igarashi and Hughes developed a 3D modeling system [Igarashi and Hughes 01] that shows a list of 3D shapes generated automatically as candidates for the next operation. The user can simply select one of these shapes if the user's intention is included in the list. If not, the user continues manually. Our approach of using prediction for drawing diagrams was motivated by this CAD system.

3 Our Proposed System

Figure 2 shows the prototype of our system interface. The upper (main) window contains three panels. The left panel has a toggle button to switch between types of folding (mountain, valley, inside-reverse, and outside-reverse) and rotation/flip buttons. The center panel displays the state of the current origami piece. The user can input a folding line by clicking two positions. The right panel shows the history of steps carried out from the initial to the current state. The lower window is a suggestion window, which shows a list of candidates for the next step. The user proceeds to the next step by clicking one of the candidates or manually inputting a folding line. The flow of the prediction process we propose is as follows:

1. Enumerate all possible folding lines by applying axioms to the current state.

2. Enumerate all possible ways to fold along each line.

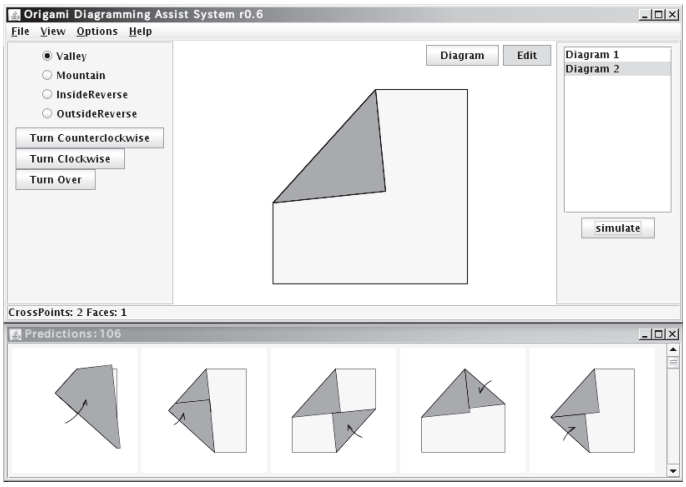


Figure 2. Our proposed system interface.

3. List shapes of origami pieces generated by applying all possible foldings as candidates.

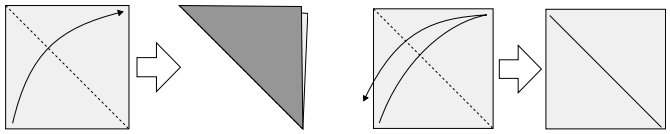
4. Remove duplicate candidates.

5. Assign a score to each candidate.

6. Display a list of candidates, sorted by score.

The prediction function requires considerable computational time; it contains the following processes: arranging folding lines, assigning mountain/valley status to a folding line, and calculating the folded configuration. Therefore, we use only the first four axioms of Huzita-Hatori (Huzita-Justin). This simplification drastically reduces computational time by limiting the number of possibilities.

We believe the effect of this simplification on the accuracy of the prediction is small because the last three axioms are rarely used in practical origami construction. Furthermore, we limit the types of folding to just valley folding and making a crease line to simplify the system (Figure 3). The latter operation, making a crease line, is the fold-and-unfold operation used to make a mark for subsequent folds. By flipping the origami piece, candidates that are equivalent to mountain folding can be obtained. Details of the prediction process are discussed in the following sections.



(a) Valley folding (b) Making a crease line

Figure 3. Folding operations.

3.1 Listing Candidates by Applying Possible Foldings

To list all possible foldings, the system first generates folding lines by applying the four axioms, using all possible points and lines in the current state. We use a simple brute-force approach that tries all possible combinations of points and lines. If multiple folding lines are created at the same position by different axioms (as can occur), then we retain one and discard the others to make each folding line unique.

Next, the system enumerates the possible ways of folding along each folding line. When a folding line passes over multiple layers, there are multiple possibilities. This is because one must choose how many layers are folded at the same time. To simplify the problem, we stipulate that only the topmost n layers may be folded at the same time along the folding line, where n can take on the value from 1 to the total number of layers. Since we apply two types of folding, valley folding and making a crease line, for the folding line, there are at most $2n$ candidates. For example, there are four possibilities when a folding line is applied to the triangle-shaped origami piece illustrated in Figure 4.

The algorithm used to enumerate candidates by the approach described is shown in Listing 1. Impossible configurations may be generated when an ith layer is connected to a lower jth layer by edge e, and e intersects the folding line. As long as only this simple algorithm is used, other impossible

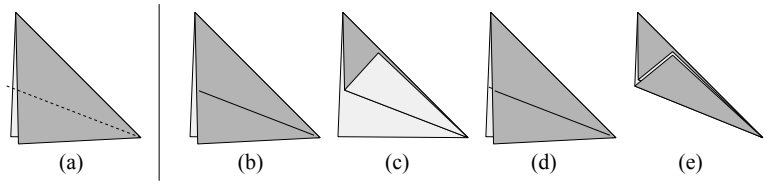


Figure 4. The four folding possibilities for an origami piece with one folding line: (a) the origami piece and a folding line (a dotted line) is applied to the piece; (b) "making a crease line" is applied only to the topmost layer; (c) "valley folding" is applied only to the topmost layer; (d) "making a crease line" is applied to all (two) layers; (e) "valley folding" is applied to all (two) layers.

```
foreach (i = 1 to n) {
  fold topmost i layers at the same time along the folding line
  if(an impossible configuration results){
    discard the result;
    continue;
  } else {
    add the result of applying valley folding
     to topmost i layers to candidates
    add the result of applying making a crease line
     to topmost i layers to candidates
  }
};
```

Listing 1. The algorithm for enumerating candidates.

cases, such as those that arise because one part of the origami piece would penetrate another part, are eliminated. Furthermore, the cases in which flaps are tucked inside other layers are eliminated.

3.2 Removing Duplicate Candidates

The list of candidates obtained by applying the process described in the previous section may contain duplicate elements once rotating and mirroring are taken into account. These duplicates are removed before the list of candidates is displayed. When considering two pieces for which the following three conditions are all satisfied, we recognize those pieces as having the same shape and configuration:

1. The numbers of polygonal parts forming the pieces are the same.
2. The sums of distances from the barycenters to each vertex are the same.
3. The lists of pairs of physically connected polygonal parts in the shapes are the same.

For the third check, we assign IDs to polygons according to the stacking order of the folded shape. Then we can check whether a connected pair listed for one candidate is also included in the list for the other candidate.

3.3 Ranking of Candidates

To make it easy to detect the intended shape from a list of candidates, appropriate sorting of the list is important. The number of candidates increases dramatically as the number of foldings increases, so it is desirable that candidates that have a high probability of being selected by the user be near the top of the list.

To assign scores to each candidate, we consider the angles between a horizontal line (x-coordinate of the screen) and the folding lines applied to the current state. The angles range from 0° to 180°. Angles of 90°, 45°, and 22.5° are often observed in common origami pieces, so we assign higher scores to candidates that have folding lines at these angles. Furthermore, as most origami pieces are symmetric and repetition is often observed, we assign higher scores to candidates that are folded along a folding line based on an angle related to those of the preceding folding step. Two angles are related if one is obtained from the other by reversing an angle through horizontal, vertical, or diagonal lines. Specifically, we assign scores according to the angles of the folding line to the following, in descending order:

- an angle related to those of the preceding folding;
- 0°, 90°, or 180°;
- 45° or 135°;
- 22.5°, 67.5°, 112.5° or 157.5°;
- anything else.

4 Results and Discussion

In this section, we consider the results, focusing upon the efficiency of the prediction function and the enumeration of candidates. We implemented our system in Java and ran it on a PC with a Core 2 Duo 2.66 GHz processor.

4.1 Efficiency of the Prediction Function

We evaluated our system by drawing a diagram for a *kabuto*, a Japanese traditional origami helmet. Figure 5 shows the diagram created by our system. The numerical values are shown in Table 1. The second column shows the ranking of selected candidates. A smaller number indicates a better prediction.

Of the eight steps needed to fold the kabuto, the desired figures were found in the list predicted by the system in five of the steps. We found that our prediction was effective, especially at the early stages of folding. One problem is that it can become difficult to find the desired figure in the generated list, especially in later steps, because the number of candidates increases in these steps. Although we proposed a method to assign scores to each origami piece, it is still difficult to assign appropriate scores to make

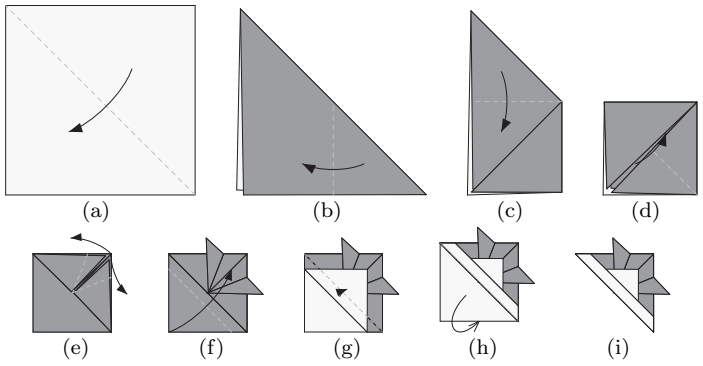


Figure 5. Diagrams for kabuto (helmet).

precise predictions. Furthermore, as the number of candidates increases, more candidates tend to be assigned the same score. To solve this problem, the strategy for assigning scores to candidates will need to be improved. For example, it would be appropriate to assign a higher score to a fold along a line located close to the previous folding line because continuous folds are usually made near each other, such as in steps 6 and 7 and in steps 2 to 5 (formation of the flaps). Adding weights to scores according to the particular axiom used would be another solution. The first and second axioms are those most often used in common origami pieces. For example, all candidates selected for drawing the diagram of the kabuto use only the first two axioms.

The computational time increases in later steps because the number of candidates can increase dramatically, depending upon the complexity of the shape. If it takes several seconds to display candidates, the system lacks interactivity. Therefore, the system would become more usable by both improving the score and reducing the number of candidates.

Number of step	Position of selected figure	Number of candidates	Processing) time (sec)
1	2	4	0.016
2	2	8	0.015
3	1	44	0.031
4	6	23	0.023
5	manual input	44	0.016
6	manual input	595	0.546
7	1	868	0.735
8	manual input	1048	0.859

Table 1. Result for kabuto (helmet).

Number of foldings	Number of candidates	Processing time
1	8	0.017 (sec)
2	1,149	0.273 (sec)
3	1,476,913	5.8 (min)
4	more than 6 billion	terminated

Table 2. The number of foldings and the number of candidates.

4.2 Enumeration of Simple Origami Shapes

By using the prediction function, it is possible to enumerate all variations of the origami pieces that can be made using the axioms. Although the system described above predicts the next folded state from the current state, it can predict multiple steps at one time by applying the prediction procedure recursively. We examined the number of pieces that recursive prediction can generate. The relation between the number of foldings and the number of variations is shown in Table 2. For this analysis, we used all seven axioms to increase the number of variations. For four folds, though, due to the large number of candidates, we could not calculate the exact value, so we estimated the number of candidates.

As expected, the number of variations increases exponentially. There are eight pieces that can be generated from a single fold, as shown in Figure 6. The number of origami pieces generated by folding twice is 1,149. Some are shown in Figure 7. When three or four foldings are made, many more variations are obtained. We were able to find some pieces that resemble recognizable shapes, such as animals, objects, and symbols, among the list of candidates. We show some examples, which we have named, in Figures 8 and 9.

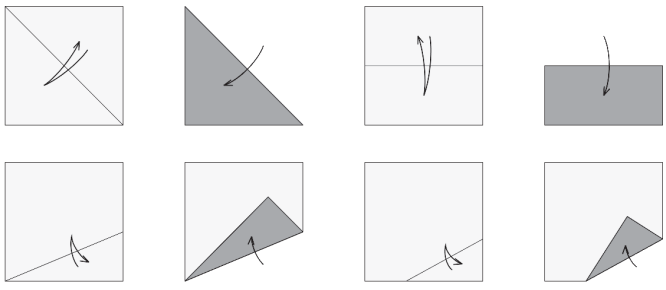


Figure 6. Eight shapes constructed using a single fold. The lower four apply axioms 5 and 6, allowing a point to be on a line.

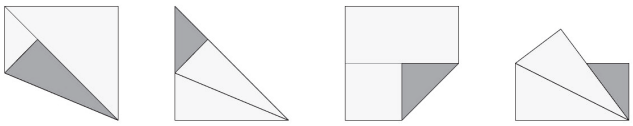


Figure 7. Example of a two-fold shape (number 4 out of 1,149).

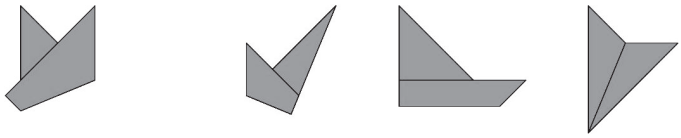


Figure 8. Examples of three-fold shapes. From left to right, fox, tick (check mark), yacht, and arrowhead (number 4 out of 13,957,372).

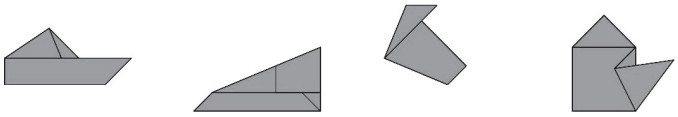


Figure 9. Examples of four-fold shapes. From left to right, boat, iron (appliance), dog, and teapot.

5 Conclusion and Future Work

We have proposed a new system for drawing diagrams using predictions for folding. The system generates folded shapes from a particular state based on the axioms and displays them as candidates. We found that the prediction is especially effective for simple origami figures that are folded in a few steps. It became clear, however, that it is difficult to apply our system to complex origami.

As noted in the Section 1, there are limitations with the prediction algorithm. We use only single-fold axioms, so the system drops many possible foldings that are used in real-world manipulations. However, reducing the number of candidates is important both for computational cost and to simplify user selection. Thus, there is a trade-off between the explosion of the number of candidates and the loss of correct ones. One solution would be to limit the area on which the fold lines are placed. Alternatively, if there was an origami database, we could use the experimental data in this database to improve the accuracy of prediction.

Another area of future work could be the automatic generation of origami pieces. Origami pieces can become complex, as we noted in Sec-

tion 1, but many pieces emphasize simplicity, such as the "2-fold Santa" [Versnick 10]. This piece expresses Santa Claus by folding a square sheet of paper only twice. Our system has the potential to generate many such simple origami pieces.

Bibliography

[Fastag 09] Jack Fastag. "eGami: Virtual Paperfolding and Diagramming." In *Origami^4: Fourth International Meeting of Origami Science, Mathematics, and Education*, edited by Robert J. Lang, pp. 273–284. Wellesley, MA: A K Peters, Ltd., 2009.

[Hatori 10] Koshiro Hatori. "K's Origami: Origami Construction." Available at http://origami.ousaan.com/library/conste.html, accessed May 17, 2010.

[Igarashi and Hughes 01] Takeo Igarashi and John F. Hughes. "A Suggestive Interface for 3D Drawing." In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, pp. 173–181. New York: ACM Press, 2001.

[Lang 06] Robert J. Lang. "TreeMaker." Available at http://www.langorigami.com/science/treemaker/treemaker5.php4, 2006.

[Mitani 08] Jun Mitani. "ORIPA: Origami Pattern Editor." Available at http://mitani.cs.tsukuba.ac.jp/pukiwiki-oripa/index.php, 2008.

[Miyazaki et al. 96] Shinya Miyazaki, Takami Yasuda, Shigeki Yokoi, and Jun ichiro Toriwaki. "An Origami Playing Simulator in the Virtual Space." *Journal of Visualization and Computer Animation* 7:1 (1996), 25–42.

[Szinger 02] John Szinger. "The Foldinator Origami Modeler and Document Generator." In *Origami^3: Proceedings of the Third International Meeting of Origami Science, Mathematics, and Education*, edited by Thomas Hull, pp. 129–136. Natick, MA: A K Peters, Ltd., 2002.

[Versnick 10] Paula Versnick. "Orihouse." Available at http://home.tiscali.nl/gerard.paula/origami/orihouse.html, 2010.