

STEP-BY-STEP FOLDING SEQUENCES FROM ORIGAMI CREASE PATTERNS USING GRAPH REWRITING

Graduate School of Systems and Information Engineering Department of Computer Science

1st-year Master Course Student: Hugo Alves AKITAYA 201220711

Advisors: Jun MITANI, Yoshihiro KANAMORI, Yukio FUKUI

2012 年 11 月 6 日

1. Introduction

Origami is the art of folding paper and traditional in the Japanese culture. Despite being an ancient art, in the past 60 years, the origami world passed through a renaissance^[1], due to the great expansion of origami worldwide, which was made possible mainly through information sharing. One of the progenitors of modern origami, Akira Yoshizawa devised a simple system to document origami, based on its step-by-step sequence that we now call customarily as *origami diagrams*.

This expansion led several origami enthusiasts into looking origami with objective and mathematical perspectives, resulting in a group of modern design techniques called *origami sekkei* (折り紙設計). Within this context, another form of origami codification became very popular among designers and, posteriorly, folders: the *crease pattern* (CP).

The crease pattern is the pattern of creases left on the square of paper after folding model. An example of crease pattern and its folded model is shown in Figure 1. The creases left by a mountain fold are represented here as red dashed lines and the ones left by a valley fold as blue lines.

This paper describes a method to find all possible folding sequences of a given CP using a registered set of origami maneuvers. Such a system can be used by inexperienced folders that cannot fold a CP or by designers that want to diagram their creations. Finding the folding sequence to an origami pattern might also be important to determine a feasible order to perform the folds in an origami-based manufacture process. A 3D animation is also generated, allowing users to easily visualize the chosen folding sequence.

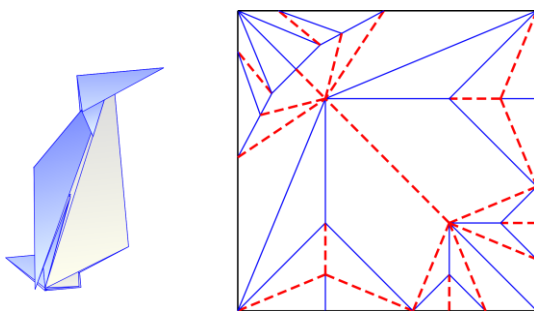


Figure 1 Example of CP (right) and its folded model (left)^[2].

2. Related research

Akitaya et al.^[4] proposes an algorithm to fold simple crease patterns, constructed with TreeMaker^[5], that have only rabbit-ear molecules. The steps are achieved using a heuristic based on the author's folding experience to apply 10 kinds of origami maneuvers to an empty CP until it becomes equal to the input. Following this method, a single sequence of intermediary CPs is found, their folded form can be calculated using ORIPA^[6] and the output images can be used to construct the model's diagrams. This algorithm's range of applicable CPs is very small and even simple origami models like the traditional samurai helmet cannot be handled.

A method for simulating rigid origami was proposed by Tachi^[7], using affine transformations described by Hull^[8]. To determine the dihedral angles between the faces, constraints are created by solving a system differential equations for each vertex using Euler integration to solve local self-intersection.

2. Characteristics of an origami step

In origami diagrams, the complexity of all the model is divided into steps in which the folder has to perform a small number of changes in the creases of the CP, manipulating only a portion of the model.

There is a basic set of origami steps that is recurrently used in diagrams, so that they have been given names. Due to space restrictions we will analyze only the inside reverse fold (or just reverse fold) as an example. Figure 2 shows a diagram representing a reverse fold and the modifications caused on the CP.

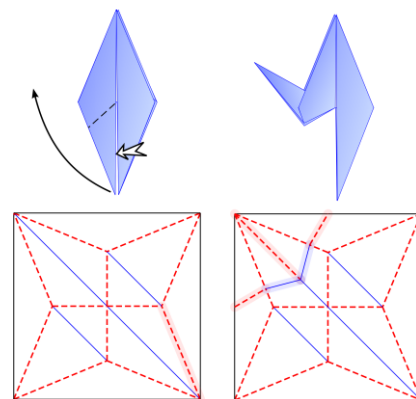


Figure 2 Example of diagrams with the changes each step causes in the CP.

2.1. Simple mountain and valley folds

When just one layer of paper is folded, there will be the simple addition of such crease to the crease pattern (as shown in Figure 3.a). However, as multiple layers are folded by a single valley/mountain step, more than one crease is added to the CP (like shown in Figure 3.b).

Let A be the set of creases added to the CP after a simple valley/mountain step. For any crease $c_i \in A$, if another crease $c_j \in A$ shares a vertex in common v_{ij} with c_i , we say that c_j is a *reflection crease* (since one is the mirrored image of the other) of c_i . The reflection creases have two properties that allow us to identify them in a CP: a) there is always a crease bisecting the angle formed by a reflection pair; b) the crease orientation (mountain/valley) of a crease is always the opposite of its reflection.

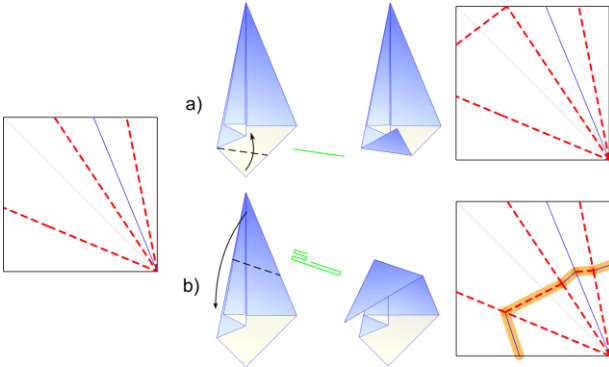


Figure 3 Example of simple valley fold: a) folding through just one layer; b) folding through multiple layers. The green lines symbolize a cut through the fold line seen from bottom view to show the configuration of the layers being folded.

If the pair c_i and c_j is removed the condition of local flat foldability^[9] will not be affected for the vertex v_{ij} . The path formed by a chain of reflection pairs is called a *reflection path*. If a reflection path i) begins and ends on the edge of the paper or ii) forms a cycle, it is called a *complete reflection path*. (like the path highlighted in Figure 3.b).

We can conclude that the removal of a complete reflection path does not affecting local flat foldability. A reflection path that has just one of its ends on the edge of the paper is called a *semi-complete reflection path*. In this work we will call *origami maneuver* an origami step that, when applied, increases the CP complexity (number of reflection paths).

3. Proposed Method

The proposed method tries to recognize the characteristics that an origami maneuver leaves on a CP after applied. By doing this, we can then unfold such

maneuver, leading into a simpler CP (decreasing its degree of complexity). Repeating those processes should result in a CP with zero complexity, i.e. an unfolded square.

To do this, a graph model for the CP is made, and the unfolding process is modeled into a graph rewriting step.

3.1. CP graph representation

A crease pattern is an embedded graph whose edges are creases inside the paper. Each crease has an orientation (valley/mountain) and a position (linking two vertices). Each vertex v_i of the CP can be represented by a node with two attributes for position and a circular linked list L_i to order the edges according to the angle between the edge and the x-axis (preserving topological information). Each crease e_i can be represented by a pair of nodes and a type to assign its orientation.

$$\begin{aligned}
 CP &= (V, E); \\
 V &= \{v_i\}_{i \in [0, n]}; \quad v_i = (x, y, L_i); \\
 x, y &\in \mathbb{R}; \\
 E &= \{e_i\}_{i \in [0, m]}; \quad e_i = (\{v_j, v_k\}, t); \\
 j &\neq k; \quad j, k \in [0, n]; \\
 t &\in \{\text{mountain}, \text{valley}\};
 \end{aligned} \tag{1}$$

3.2. Origami maneuver as a rewriting step

Let CP_0 be the initial CP (that corresponds to the model at the last step of the diagrams) and CP_{-1} the CP representing the state of the model in the penultimate step of the diagrams. We can model a reverse fold as a graph rewriting step in which we transform a subgraph matching the pattern $S_{reverse}$ into the graph $S'_{reverse}$ (this rewriting rule can be written as the morphism $r: S_{reverse} \rightarrow S'_{reverse}$).

We denote $CP_{-1} \xRightarrow{r} CP_0$ to express that CP_0 is the result of the rewriting step that has CP_{-1} as the host graph and r as the rewriting rule. By finding a matching for $S'_{reverse}$ in CP_0 , we can, then, find a possible occurrence of a reverse fold in the diagrams ($CP_0 \xRightarrow{r^{-1}} CP_{-1}$, thus finding CP_{-1} by application of the reverse rule $r^{-1}: S_{reverse} \rightarrow S'_{reverse}$).

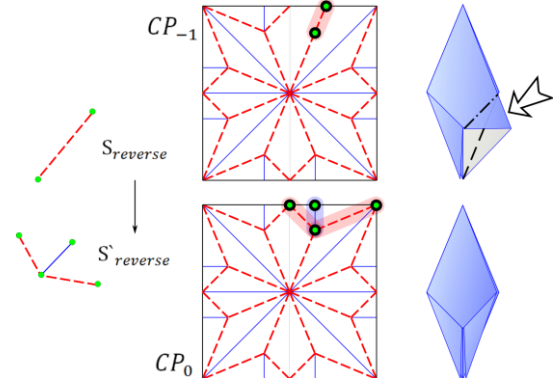


Figure 4 Example of reverse fold as graph rewriting.

3.3. Matching validation process

In simple examples in which each fold of a maneuver pattern modifies just one layer of paper (like the one shown in Figure 4), the simple rewriting step is enough to generate the previous step. But when the folds affect more layers of paper, they cause the addition of reflection creases. Those reflection creases also have to be removed in order to unfold the step properly.

If the rewriting step is not enough to assure local flat foldability on a node, this node must be on the edge of the paper or connected to a semi-complete reflection path. If all nodes of a matching obey the above mentioned rule, this matching is *valid*. Figure 5a) exemplify an invalid matching; 5.b), 5.c) 5.d) exemplifies a case of a valid matching and its unfolding result.

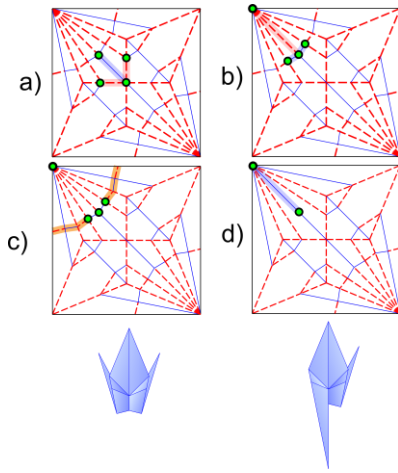


Figure 5 a), b) and c) represents the same CP. a) Example of invalid matching of reverse fold; b) Example of a valid reverse fold matching; c) Two semi-complete reflection path are highlighted; d) CP after applying the unfolding technique and its folded form.

3.4. The step sequence graph

Let the registered set of origami maneuvers rewriting rules $\mathbf{M} = \{(S_i \rightarrow S'_i)\}_{i \in [0, l]}$. For a general state of the origami model represented by CP_n , there might be more than one pattern matching, considering both multiple occurrences of the same rule as occurrences of different maneuver rules. This fact shows that the previous step CP_{n-1} is not unique. We represent as CP_{n-1}^k the k-th previous step generated by the unfolding algorithm applied to CP_n .

Because the first step of the diagram must always be the unfolded square, we can also expect that CPs of the sequence may also have more than one next step. For example, $CP_{-1}^2 \Rightarrow CP_{-3}^1$ and $CP_{-2}^1 \Rightarrow CP_{-3}^1$ might occur. We can organize all the steps into a directed graph as shown in Figure 6 that is called step-graph. By filling the step-graph, we have all possibilities to unfold the input model with the maneuvers within \mathbf{M} . If we get a node

representing the unfolded paper (i.e. $\exists i, j | CP_{-i}^j = (\emptyset, \emptyset)$), CP_0 is foldable under \mathbf{M} . If we invert the edges of the *step sequence graph* every path beginning with the unfolded paper and ending with CP_0 is a possible step-by-step sequence.

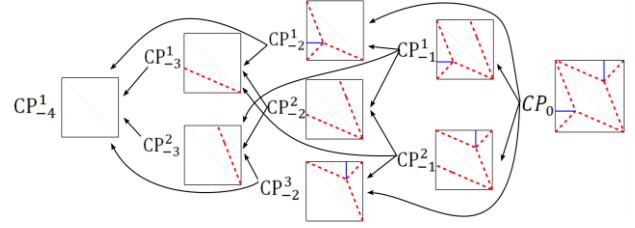


Figure 6 Example of step sequence graph for the traditional fish base. \mathbf{M} contains reverse fold, rabbit-ear fold and simple valley/mountain folds.

4. Simulation

After one possible folding sequence is chosen from the step sequence graph, a 3D animation of the folding process can be created using affine transformations as described in [8]. As the number of moving faces is usually small when animating a single maneuver, an easier and straightforward approach was used as constraints to determine the dihedral angles.

Considering each vertex as the center of a sphere with radius equal to one, the interception of the faces and sphere is a spherical polygon. By dividing this polygon into spherical triangles, it is possible to calculate each dihedral angle applying the spherical law of cosines. For the non-rigid-foldable maneuvers, the angular velocity of each face is maintained constant and distortion is added to join adjacent faces.

5. Results

The implemented software uses brute force to solve the subgraph isomorphism in the rewriting steps. The GUI is shown in Figure 7. The input is a ORIPA file containing CP_0 .

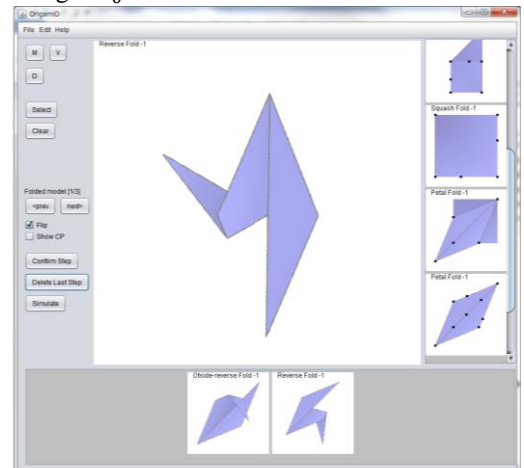


Figure 7 Software implementation GUI.

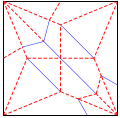
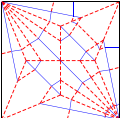
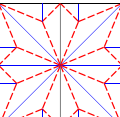
5.1. Sequence generation results

The GUI allows the user to navigate through the step sequence graph from the square towards the folded form in an intuitive way.

The foldability of a model is related with M. With M containing 4 basic folds (inside reverse fold, outside reverse fold, squash fold and petal fold), it is possible to fold most of traditional models and some simple modern designed models like the one illustrated in Figure 1. More complex models should require more specific maneuvers.

Table 1 shows some statistics about the computation time (computation time was measured using the following environment: Intel® Core™ i7-2600 CPU 3.4GHz, RAM: 4GB) and complexity for some examples handled by the software. We can conclude by the results that the cost explodes as the complexity of the model increases. This is due to the increase of folding possibilities.

Table 1 Computation cost for three examples
(* referring to the step sequence graph)

	Comp. time (ms)	Number of nodes*	Number of arcs*
	295	23	51
Figure 2 example			
	451	41	75
Traditional crane			
	1,780,582	22,665	73,204
Traditional frog base			

The simulation result shows that for simple maneuvers, collisions were avoided and the faces movement could be determined smoothly using spherical geometry.

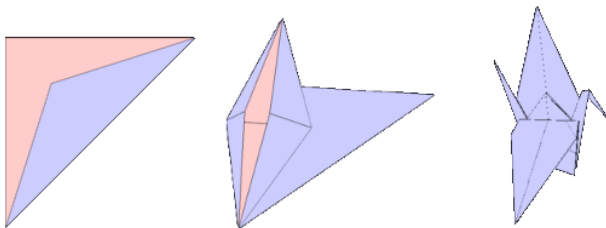


Figure 8 Simulating the folding sequence of the diagrams of a traditional crane.

6. Conclusion and future work

In this work, a method for finding folding sequences from origami crease patterns using graph pattern matching have been presented along with its partial software implementation. The software can handle CPs that can be folded only with simple valley and mountain folds and simple maneuvers, but the software gives the possibility to expand its scope, by inputting new maneuvers into the maneuver registration file. It can accelerate substantially the task of diagramming.

Considerations about maneuver priorities and model symmetry can solve the problem of explosion of possibilities shown in section 5.1, discarding possibilities of candidates that should rarely be chosen. In the simulation, global self-intersection avoidance is still not solved. There are also maneuvers that are not rigidly foldable, meaning that there should be distortions in the faces of paper. We intend also to investigate a smooth way to perform this in 3D animation.

References

- [1] Lang R. , “Origami Design Secrets: Mathematical Methods for an Ancient Art“, Second Edition. CRC Press (2012), pp.3-4, 24, 38, 680-681.
- [2] Komatsu H. , “Penguin“, accessed 2012/11/04, available at: <http://origami.gr.jp/~komatsu/gallery/penguin.html/>
- [3] Mroz K. and Pipkorn B., “Mathematical Modeling of the Early Phase Deployment of a Passenger Airbag – Folding using Origami Theory and Inflation Using LS-DYNA Particle Method”, *6th European LS-DYNA Users’ Conference*. Gothenburg, Sweden (2007).
- [4] Akitaya H. et al., “Development of an Intuitive Algorithm for Diagramming and Animated Tutorial for Folding Crease Patterns”, *Origami 5: Fifth International Meeting of Origami Science, Mathematics, and Education*. CRC Press, (2011), pp. 347.
- [5] Lang R., “TreeMaker”, accessed 2012/10/20, available at: <http://www.langorigami.com/science/computational/treemaker/treemaker.php/>
- [6] Mitani J., “ORIPA: Origami Pattern Editor”, accessed 2012/10/20, available at: <http://mitani.cs.tsukuba.ac.jp/oripa/>
- [7] Tachi T., “Simulation of Rigid Origami”, *Origami4: Proceedings of The Fourth International Conference on Origami in Science, Mathematics, and Education*. CRC Press, (2009), pp. 175–187.
- [8] Belcastro S. and Hull T., “Modelling the folding of paper into three dimensions using affine transformations”, *Linear Algebra and its Applications*, Vol. 348 (2002), pp. 273-282.
- [9] Hull T. , “On the Mathematics of Flat Origamis”, *Congressus Numerantium*, Vol. 100 (1994) , pp. 215-224.