An e-origami artwork of a big wing crane

Tetsuo Ida[†]

University of Tsukuba, 1-1-1 Tennoudai, Tsukuba, 305-8573, Japan

Abstract

We present the construction of a big wing crane as an application of folding with virtual cutting and gluing edges of origami faces in the e-origami environment. With the new methodology, we can reason about the construction algorithm for classical origami in finer steps rather than relying on the skill of the human hand. We employ a new origami model, as described in our earlier work, and we have developed software that implements classical origami folds.

Keywords

e-origami, classical fold, paper folding rule, cut and glue edges

1. Introduction

In our previous paper [1], we introduced a new technique of *cut-and-glue of a shared face edge* to e-origami[1]. We observed that an origami artwork is a complex arrangement of bounded two-sided flat planes, or faces, intricately connected and superposed by repeated folds of a single (virtual) sheet of paper. This collection of faces culminates in an object with a remarkable shape.

Our research demonstrates that cutting an edge shared by two faces unveils a class of classical folds. By gluing the faces divided by the cut, we restore the connection of the separated faces. This cut-and-glue technique opens up vast possibilities, enabling the discovery of new folds that were previously deemed impossible by Huzita-Justin folds, which, when applied to practical constructions, have certain limitations that our approach overcomes [2] and [3]. The inside reverse fold, one of the most straightforward classical folds, is not included in the Huzita-Justin folds. When we apply the cut-and-glue technique, we can realize the inside reverse fold by combining Huzita-Justin folds. We demonstrate the practical application of our method by constructing a big wing crane, a well-versed sophisticated origami structure [4] that demands a deep research investigation.

2. Modeling for e-origami

Origami is a term meaning "folding paper." It also refers to a sheet of paper used for origami. Folding an origami along a fold line and unfolding the fold to the previous shape leaves a line segment, called a *crease*, on the origami. We can construct various interesting geometric objects when we freely choose fold lines and allow overlaps of faces without breaking the original sheet. When we impose mathematically plausible fold line construction rules, we can define an origami geometry that deserves deep mathematical investigation.

Euclidean (plane) geometry constructs geometrical objects using only a straightedge and a compass. Similarly, origami geometry, a tool-less approach, defines its rules. Huzita-Justin's rules are the commonly agreed rule set on which origami geometry is based. In Table 1, we list some (4 out of 7) of the Huzita-Justin rules implemented in the Eos system [5] that we use to construct a big wing crane. These rules, together with newly implemented classical folds (see next section) and software tools of Eos, allow us to manipulate origami flexibly.

SCSS 2024: The 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan

ida@cs.tsukuba.ac.jp (T. Ida)

https://www.i-eos.org/ida (T. Ida)

0000-0002-5683-216X (T. Ida)

[1]We use the term *e-origami* to refer to origami innovated by information technology.

Table 1
Huzita-Justin rules

Rule	Command	Operation
(O1)	HO$[PQ]$	Fold along line PQ
(O2)	HO$[P, G]$	Fold to superpose point P and point Q
(O3)	HO$[PQ, RS]$	Fold to superpose line PQ and line RS
(O4)	HO$[PQ, X]$	Fold along the line perpendicular to line PQ that passes through point X

3. Modeling for e-origami

We give a mathematical notation for the definition of origami and reason about its properties. In origami, we have faces and two kinds of neighborhood relations between the faces, i.e., superposition and adjacency. The superposition is a vertical neighborhood relation, and the adjacency is a horizontal one. A face is a polygon having an attribute of sides. When we denote polygon $P_1 \dots P_n$, where we arrange points $P_1 \dots P_n$ counterclockwise, we call the plane *front* side, and the polygon $P_n \dots P_1$ is identified the same as with polygon $P_1 \dots P_n$ with the attribute of the *back* side. For any faces, $f = P_1 \dots P_n$ and $g = Q_1 \dots Q_n$, f is equal to g iff g is a cyclic permutation of f or a cyclic permutation of a reverse of f.

We let Π be a finite set of faces, $\backsim$ be a binary relation on Π, called *adjacency relation*, and $\succ$ be a binary relation on Π, called *superposition relation*. An abstract origami is a structure $(\Pi, \backsim, \succ)$. We abbreviate abstract origami to *AO*. We denote the set of AOs by $\mathbf{O}$. An abstract origami system is an abstract rewriting system $(\mathbf{O}, \looparrowright)$ [6] , where $\looparrowright$ is a rewrite relation on $\mathbf{O}$, called *abstract fold*.

For $\mathcal{O}, \mathcal{O}'(\in \mathbf{O})$, we write $\mathcal{O} \looparrowright \mathcal{O}'$ when $\mathcal{O}$ is abstractly folded to $\mathcal{O}'$. We begin an origami construction with an initial AO and perform an abstract fold repeatedly until we obtain the desired AO. Usually, we start an origami construction with a square sheet of paper. This initial sheet of paper is abstracted as a structure having a single distinguished face denoted by the numeral 1. Then, the initial AO $\mathcal{O}_1$ is represented by $(\{1\}, \emptyset, \emptyset)$. Furthermore, when we fold face n, the face is divided into two faces $2n$ and $2n + 1$. We use this convention in this paper and the realization of the data structure of the e-origami system Eos [7] and the origami language Orikoto of Eos. Suppose that we are at the beginning of step i of the construction, having AO $\mathcal{O}_{i-1} = (\Pi_{i-1}, \backsim_{i-1}, \succ_{i-1})$. We perform an abstract fold and obtain a next AO $\mathcal{O}_i = (\Pi_i, \backsim_i, \succ_i)$. Thus, we have the following $\looparrowright$-sequence.

$$\mathcal{O}_1 \looparrowright \mathcal{O}_2 \looparrowright \cdots \looparrowright \mathcal{O}_n$$

An abstract origami construction is a finite $\looparrowright$-sequence of AOs. In concrete terms, the operation $\looparrowright$ can be a fold by one of Huzita-Justin's rules, a mountain fold, a valley fold, etc., each requiring arguments of different kinds. We abuse $\mathcal{O}_i$ to be a name of the origami constructed at step i.

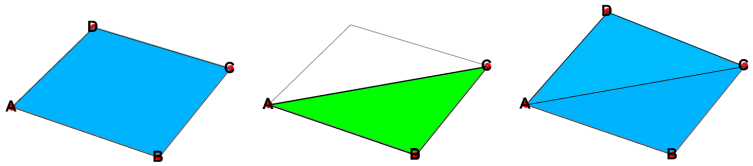
We now move on to concrete origami. Origami is a term meaning "folding paper." It also refers to a sheet of paper used for origami. Folding an origami along a fold line and unfolding the fold to the previous shape leaves a line segment, called a *crease*, on the origami. We can construct various interesting geometric objects when we freely choose fold lines and allow overlaps of faces without breaking the original sheet. When we impose mathematically plausible rules for choosing fold lines, we can define an origami geometry that deserves deep mathematical investigation.

Euclidean (plane) geometry constructs geometrical objects using only a straightedge and a compass. Similarly, origami geometry, a tool-less approach, defines its rules. Huzita-Justin's rules are the commonly agreed rule set on which origami geometry is based. In Table 1, we list some (4 out of 7) Huzita-Justin rules that we use to construct a big wing crane. These rules, newly implemented classical folds (to be discussed in the next section), and our software tools allow us to manipulate origami flexibly.

4. Classical Folds

4.1. Mountain fold and Valley fold

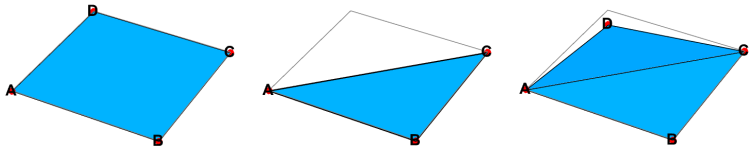
Understanding the fundamental fold operations of the mountain and valley fold is crucial in origami. These folds are named after the resemblance of the crease to a mountain ridge and a valley lap. The crease is formed by the unfold operation that follows the mountain (or valley) fold. The result of the valley fold is shown in Fig.1(b). The origami is now two-layered, but the back layer is not visible since the upper triangle face completely overlaps the lower triangle face. Next, we unfold the origami shown in Fig. 1(c). Unfold does not mean "undoing," although we recover the shape of the origami to the one in Fig.1(a) except for the dotted line segment CA. We call the line segment valley crease. If we have a rich imagination, the crease looks like a lap in the valley.



(a) before valley fold (b) after valley fold (c) after unfold

Figure 1: Valley fold

Similarly, we have a mountain fold below.



(a) before mountain fold (b) after mountain fold (c) after unfold

Figure 2: Mountain fold

The mountain and valley folds are similar to the Huzita-Justin rule (O1), with the important difference that rule (O1) operates on all the stacked faces. In contrast, the mountain and valley folds apply to automatically selected faces.

The terms “valley” and “mountain” are sometimes misleading to beginner origami hobbyists since those folds are not necessarily immediately followed by an unfold. Hence, valley and mountain creases may not appear unless the origami is entirely unfolded. More geometrically clear terminology is desired for origami geometers.

4.2. FO

The above observation led us to define command FO. FO, standing for Fold Origami, is a command FO[θ][*faces*, *ray*] for rotating target faces along *ray* by angle θ. It is a generalization of the mountain and valley folds. The implementation of FO contains an algorithm to select the target faces to be rotated based on *faces*.

Note that FO is defined as a Curried function. Using FO, we could define ValleyFold and MountainFold as follows:

```
ValleyFold = FO[-π]; MountainFold = FO[π];
```

With θ other than $\pm\pi$, FO constructs a 3D origami in general. Usually, we use FO[θ] towards the ending steps of the construction.

4.3. Inside reverse and outside reverse folds

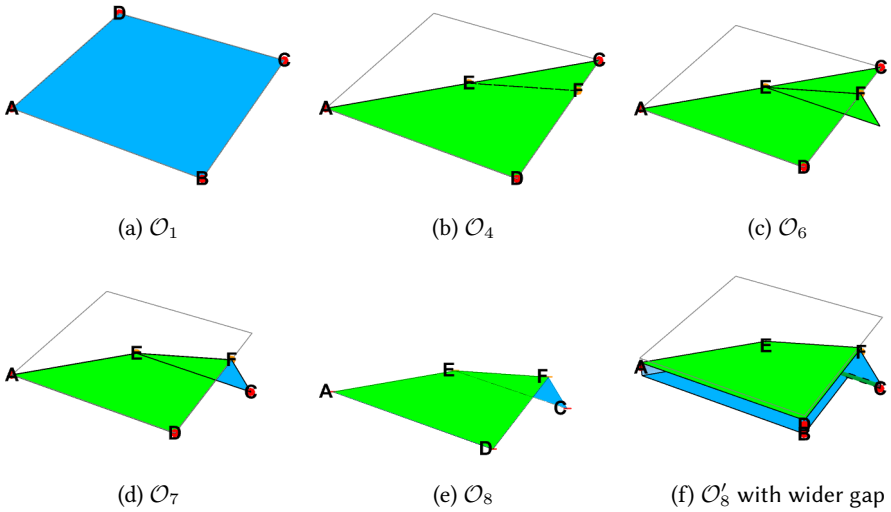
The inside reverse and outside reverse folds are often used in paper folding. Both work on a pair of superposed faces that share an edge. When the cut-and-glue technique is introduced, both folds are realized by combinations of the valley and mountain folds. However, each uses the valley and mountain folds on opposite faces.

4.3.1. Inside reverse fold

Below we show a simple example of an inside reverse fold. Let

$$\mathcal{O}_1 \looparrowright^* \mathcal{O}_4 \looparrowright^* \mathcal{O}_6 \looparrowright \mathcal{O}_7 \looparrowright \mathcal{O}_8$$

be a construction sequence of the example. Each origami $\mathcal{O}_i, i \in \{1, 4, 6, 7, 8\}$ is visualized in Fig. 3.



(a) $\mathcal{O}_1$ (b) $\mathcal{O}_4$ (c) $\mathcal{O}_6$

(d) $\mathcal{O}_7$ (e) $\mathcal{O}_8$ (f) $\mathcal{O}'_8$ with wider gap

Figure 3: Inside reverse fold

Origami $\mathcal{O}_4$ is consructed from $\mathcal{O}_1$ by a sequence of commands; VallyFold, ValleyFold, and Unfold. It is a double-layered stack of faces. On the top layer are two faces, i.e., face CFE and face AEFD. On the second one, i.e., the bottom layer, are two faces of the same shapes as the ones above but in opposite orientations. Each face is identified by a unique face ID, automatically assigned by the system. In this elementary example, we do not have to be concerned with face IDs since it is unnecessary to specify to which faces we apply the inside reverse fold. To the example $\mathcal{O}_4$ of Fig. 3(b) we apply the following command:

```
InsideReverseFold["CE", "FE"]
```

The first argument, CE, specifies the edge to be cut. It is specified as the type ray. The second argument, FE, is the ray along which mountain and valley folds are performed. The above inside reverse fold command performs the following operations:

1. Check if the inside reverse fold is feasible. Namely, check if $\angle\mathrm{CFE} \leq \pi/2$.
2. Cut the edge CE. As a result, point C is split into C and C_1(not shown). The result is $\mathcal{O}_5$. Origami $\mathcal{O}_5$ is not shown in Fig. 3.
3. Valley fold along ray FE. The face C_1EG is moved. *We impose a rule that faces to the right of a fold line (interpreted as a ray) are moved by a fold.* The result is $\mathcal{O}_6$,
4. Mountain fold along ray FE. The face CFE is moved. The result is $\mathcal{O}_7$.
5. Glue the moved edges CE and C_1E to form a new edge CE. The result is $\mathcal{O}_8$.

Note in passing that performing Steps 2 and 3 above in sequence would only be possible when we cut and separated the shared edge EC. As we construct origami in a virtual space, we can cut CE and glue the moved CEs without complication. After following the above steps, the obtained origami is shown in Fig. 3(e) and (f); the latter illustrating the ins and outs of the origami object. Our specially designed viewer generates this graphics image, providing a comprehensive view of the origami.

On the other hand, in the case of the outside reverse fold, we make a polygonal cover on the faces.

4.3.2. Outside reverse fold

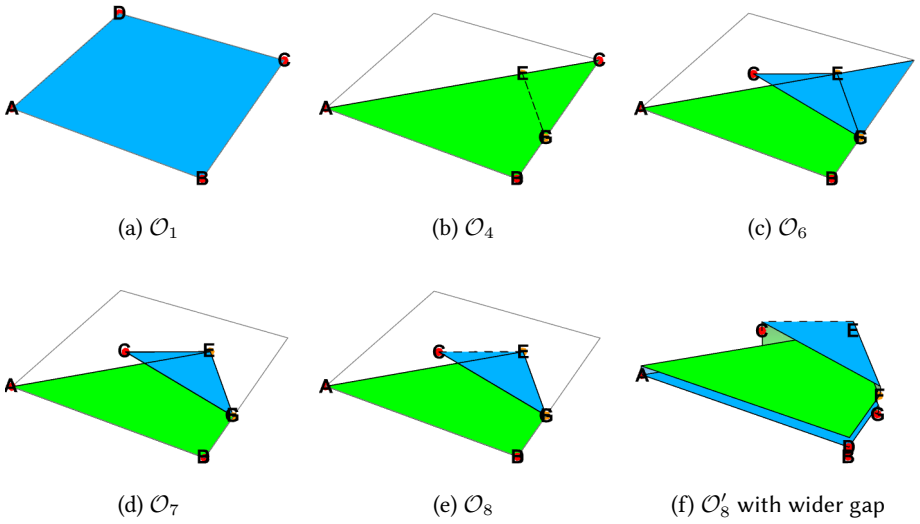
An outside reverse fold is similar to an inside reverse fold. The difference is that the outside reverse fold applies mountain and valley folds to different faces in opposite layers. We refer to the construction sequence

$$\mathcal{O}_1 \looparrowright \mathcal{O}_2 \looparrowright \cdots \looparrowright \mathcal{O}_8,$$

where $\mathcal{O}_i, i = 1, \ldots, 8$ are visualized in Fig. 4(a)∼(e). We apply the following command:

```
OutsideReverseFold["CE","FE"]
```
(1)

to $\mathcal{O}_4$, and obtain $\mathcal{O}_8$. Note that constraint $\pi/2 \leq \angle CEF \leq \pi$ should be satisfied; otherwise the execution of (1) fails. The origami construction sequence in Fig. 4 is now self-explanatory. Figure 4 (f) shows the origami structure of $\mathcal{O}_8$ more clearly.



(a) $\mathcal{O}_1$ (b) $\mathcal{O}_4$ (c) $\mathcal{O}_6$

(d) $\mathcal{O}_7$ (e) $\mathcal{O}_8$ (f) $\mathcal{O}'_8$ with wider gap

Figure 4: Outside reverse fold

4.3.3. Other classical folds

In addition to the classical folds we have discussed so far, we analyzed the folding algoriths of the following classical folds well-known in the origami community and implemented them: SquashFold, RabbitEarFold, SwivelFold, InsideCrimpPleatFold, and OutsideCrimpPleatFold, using the cut-and-glue technique. They are given in the Appendix.

5. Construction of a big wing crane

5.1. Overview

A crane origami is an exciting example. It is one of the best-known artworks, requiring the classical folds we discussed. Making a delicate crane by hand is challenging for beginner origami hobbyists. In

the context of e-origami, this example poses another challenge in modeling a new class of folds. We will present the construction of a big wing crane origami. However, instead of describing in natural language with guiding icons and annotation, we present an algorithm for constructing a crane origami. This algorithm provides a step-by-step prescription for folding the crane in a programming language. We present a big wing crane, a well-known flying crane origami variation [4]. Algorithmically, it is simpler than the ordinary, well-known ones.

We start with a rhombus-shaped piece of paper. We have included entire program codes in a separate webpage (www.i-eos.org/orikoto-program-of-a-big-wing-crane) to help the reader better understand the internal origami structure. These codes are not just for show-they allow us to visualize the intricate folds and their arrangement, making the construction process more accessible. In this construction, we specify the origami faces of textured patterns on the completed work, adding an artistic dimension to the origami. This approach, requiring a special texture mapping algorithm, opens up new creative possibilities in origami modeling. To complete the artwork shown below, we need 71 steps. In one step, an origami object will make one structural change. For example, we count one structural change of the origami object in one application of a valley fold, a mountain fold, and one of the Huzita-Justin folds. One inside reverse fold requires four substeps, i.e., cut, valley fold, mountain fold, and glue. The number of steps for the entire construction is surprisingly large at first sight, but we should observe that the origami objects have several symmetries. Therefore, a similar code sequence is repeated more than once. The number of crucial operations is limited. While the space is limited in this paper, we assure the readers that they grasp the essence of the construction by showing carefully selected crucial steps and output graphics of the origami construction process.

The construction consists of the following four stages.

(1) bird base construction,
(2) leg construction,
(3) bill construction,
(4) wing construction.

We will briefly explore the steps in each stage.

5.2. Bird base construction

LLet τ be a construction $\mathcal{O}_1 \looparrowright^* \mathcal{O}_k$, where $k > 1$ and $\mathcal{O}_k$ has a certain distinguidh feature $\mathcal{A}$, we call τ an $\mathcal{A}$-base construction. We want to construct a bird-like origami $\mathcal{O}_k$. In this stage, we will construct a bird base, i.e., $\mathcal{A}$ is a bird and $k = 49$. We also call $\mathcal{O}_k$ a bird base.

Let τ be a construction $\mathcal{O}_1 \looparrowright^* \mathcal{O}_k$, where $k > 1$ and $\mathcal{O}_k$ has a certain distinguidh feature $\mathcal{A}$, we call τ an $\mathcal{A}$-base construction. We want to construct a bird-like origami $\mathcal{O}_k$. In this stage, we will construct a bird base, i.e., $\mathcal{A}$ is a bird and $k = 49$. We also call $\mathcal{O}_k$ a bird base.

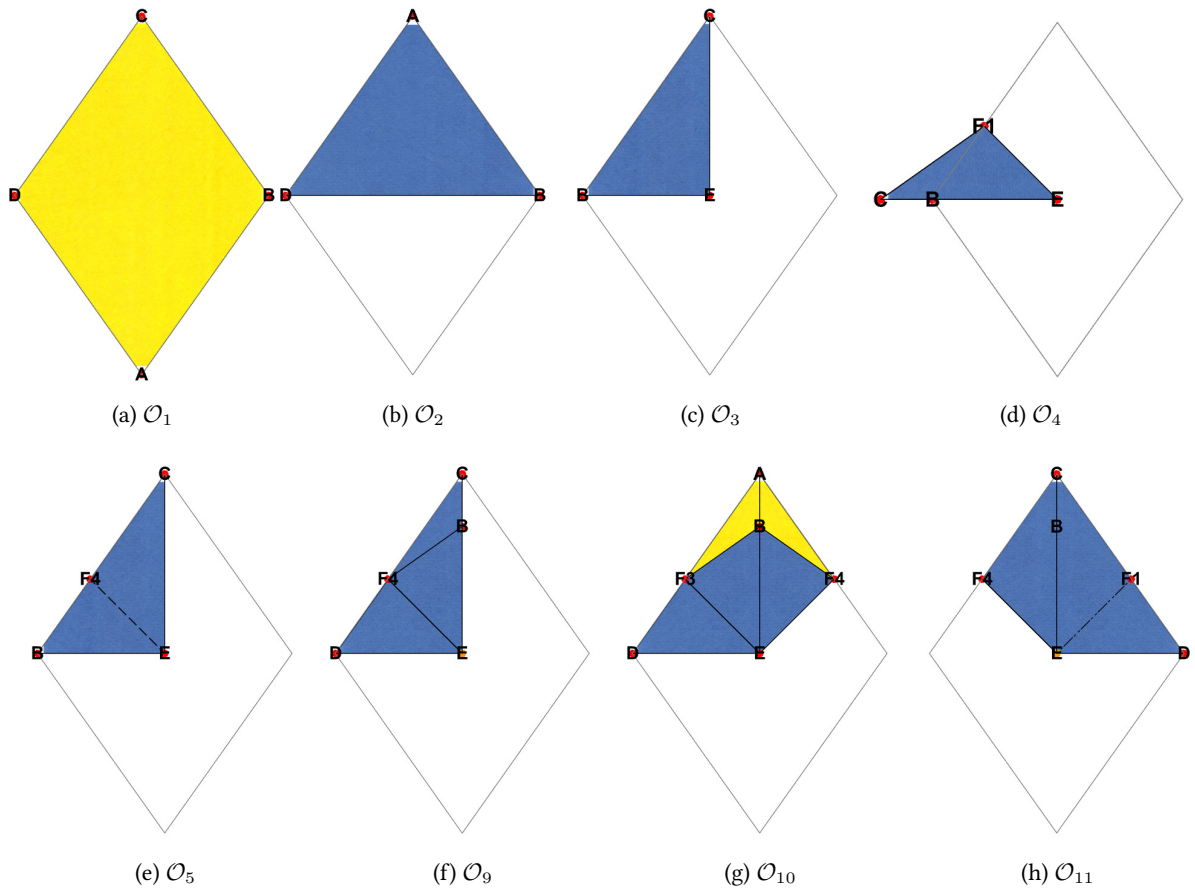
The first 11 steps in τ produce $O_1, \ldots, O_{11}$ shown in Fig. 5. We start with an initial origami of a rhombus shape to make the crane's wings bigger than the crane made from the square initial origami. Applying rule (O2) to $\mathcal{O}_1$ and $\mathcal{O}_2$, we obtain $\mathcal{O}_3$. $\mathcal{O}_3$ is quad layered. We apply rule (O3) to fold $\mathcal{O}_3$ along the bisector of $\angle$BCE, and then unfold $\mathcal{O}_4$ to obtain $\mathcal{O}5$. Steps 4 and 5 aim to construct a valley crease F4E that will be used in the inside reverse fold on $\mathcal{O}_5$. $\mathcal{O}_5$ has three other creases, F3E, F2E, and F1E, at the intersection of the bisector and the hypotenuses of the right triangular faces on four layers.

The subsequence of the construction $\mathcal{O}_5 \looparrowright^* \mathcal{O}_9 \looparrowright \mathcal{O}_{10}$ shows the first application of the inside reverse fold on $\mathcal{O}_5$:

```
InsideReverseFold}["BE", "F4E"].
```

The application of InsideReverseFold requires four substeps, i.e., cut, valley fold, mountain fold, and glue, and returns $\mathcal{O}_9$. The top view of $\mathcal{O}_5$ and $\mathcal{O}_9$ appears the same, but two triangular faces BF3E and BF4E have been moved below face CEF4 in $\mathcal{O}_9$ by the application. When we move face CEF4 at step 10, we observe the difference. By turning over $\mathcal{O}_{10}$, we have origami $\mathcal{O}_{11}$ seen from the backside[2].

[2]We have a command TurnOver[ray], which is a variation of FO[π][ray].

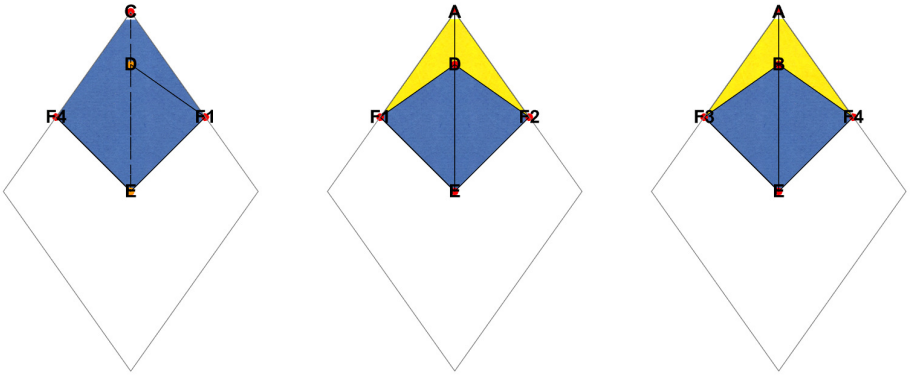


(a) $\mathcal{O}_1$ (b) $\mathcal{O}_2$ (c) $\mathcal{O}_3$ (d) $\mathcal{O}_4$

(e) $\mathcal{O}_5$ (f) $\mathcal{O}_9$ (g) $\mathcal{O}_{10}$ (h) $\mathcal{O}_{11}$

Figure 5: Subsequence 1 of the bitd base

Similarly, we apply another inside reverse fold to the right half of $\mathcal{O}_{11}$, and by following the construction sequence $\mathcal{O}_{11} \not\mapsto^* \mathcal{O}_{15} \not\mapsto \mathcal{O}_{16} \not\mapsto \mathcal{O}_{17}$.



(a) $\mathcal{O}_{15}$: by insert reverse fold (b) $\mathcal{O}_{16}$: by (O1) (c) $\mathcal{O}_{17}$: by turn over

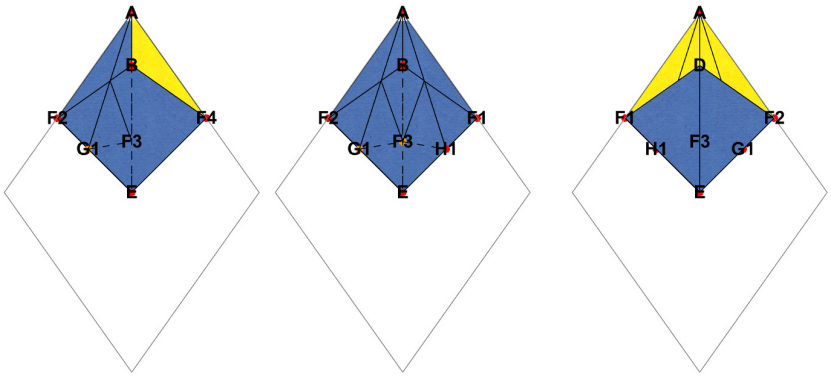
Figure 6: Subsequence 2 of the bird base

The rest of the subsequence in the bird base is given in Figs. 7 and 8 with annotation in the title of each sub-figure. Each sequence of figures are the visualization of

$$\looparrowright^* \mathcal{O}_{23} \looparrowright \mathcal{O}_{29} \looparrowright \mathcal{O}_{30},$$

and

$$\looparrowright^* \mathcal{O}_{36} \looparrowright \mathcal{O}_{42} \looparrowright \mathcal{O}_{43} \looparrowright^* \mathcal{O}_{46} \looparrowright \mathcal{O}_{48} \looparrowright \mathcal{O}_{49}.$$



(a) $\mathcal{O}_{23}$: by insert reverse fold (b) $\mathcal{O}_{29}$: by insert reverse fold (c) $\mathcal{O}_{30}$: by turn over

Figure 7: Subsequence 3 of the bird base

At the end of the bird base construction, we obtain O_{48} and $\mathcal{O}_{49}$. The latter is the bird base. The bird base is crucial in origami crane construction, whether for the big wing crane or for a commonly known classical crane. This slim diamond-shaped origami piece has a crack in the middle of the upper half shown in $\mathcal{O}_{49}$ in Fig. 8(f). This piece is quad-layered, and each double layer is vertically symmetric as well as horizontallly symmetric with respect fo the axis AB.

5.3. Leg construction

The leg construction subsequence is the following:

$$\looparrowright^* \mathcal{O}_{53} \looparrowright^* \mathcal{O}^*_{57} \looparrowright^* \mathcal{O}_{61}$$

We apply the inside reverse fold to the right and left parts of the bird base O_{49}. Note that the inside reverse fold can be applied to multiple layers of faces.

5.4. Bill construction

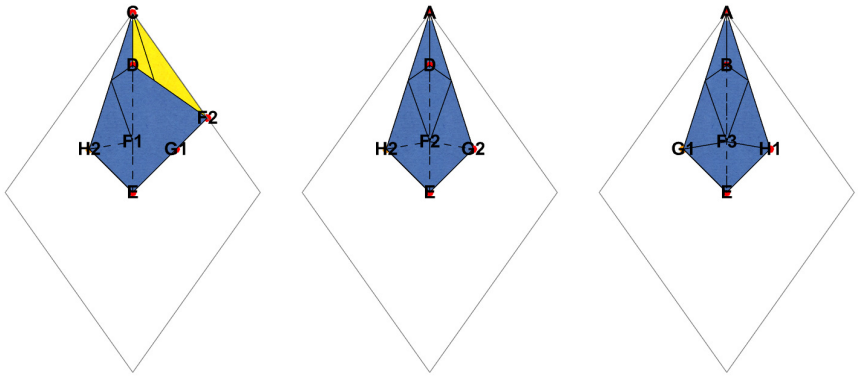
The bill construction subsequence is the following:

$$\looparrowright^* \mathcal{O}_{61} \looparrowright^* \mathcal{O}_{63} \looparrowright^* \mathcal{O}_{67} \looparrowright \mathcal{O}_{68}$$

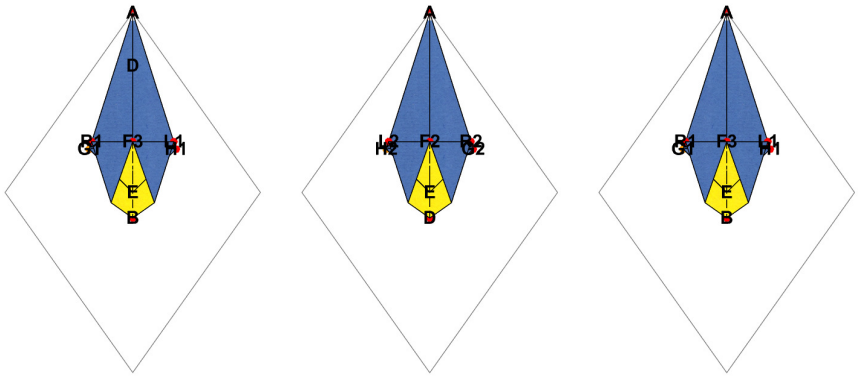
Here, we make crease Z1Z2, where we choose by the designer's preference arbitraly positions of points Z1 and Z2 on the edges. The crease determines the twist of the bill. We apply the inside reverse fold on O_{62}. Then, we rotate O_{66} along R1L1 by π and obtain O_{68}.

5.5. Wing construction

Finally, we apply the command FO twice to make the origami three-dimensional. We apply FO[-(3/8) π] to those faces that constitute the wings. We complete the construction to bring the bill partly to the right, as this is our preferred posture. Our viewer can manipulate the origami object. The viewer and



(a) $\mathcal{O}_{36}$: by insert reverse fold (b) $\mathcal{O}_{42}$: by insert reverse fold (c) $\mathcal{O}_{43}$: by turn over



(d) $\mathcal{O}_{46}$: bring inner face to outside (e) $\mathcal{O}_{48}$: bring inner face to outside (f) $\mathcal{O}_{49}$: by turn over

Figure 8: Subsequence 4 of the bird base

Mathematica's graphics functions [8], for example, remove the point names and change the size of the image and lighting.

$$\looparrowright^* \mathcal{O}_{69} \looparrowright \mathcal{O}_{70} \looparrowright \mathcal{O}_{71}$$

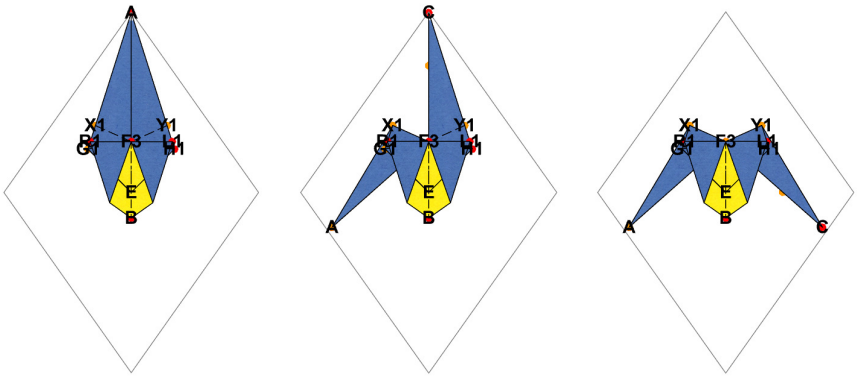
5.6. Texture mapping and adjustment of posture

The following are the results of our artwork. Figure 11(a) is the polished version of $\mathcal{O}_{71}$. We removed the point names, enlarged the image, and readjusted the posture of the crane. Figures 11(b) and (c) are obtained by adding textures to the image of $\mathcal{O}_{71}$, using the functionality of texture mapping of Mathematica.

6. Concluding remarks

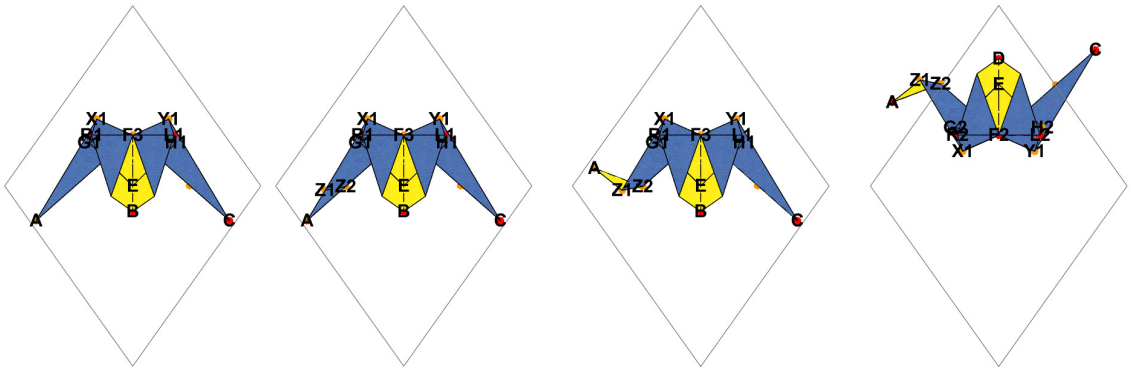
We have shown that the cut-and-glue technique simplifies modeling the classical folds and, hence, the implementation in the e-origami environment. Using a cut operation, we have shown that the seemingly complex inside (and outside) reverse fold is reduced to a sequence of FO folds. We illustrated other popular classical folds, such as a rabbit ear fold, are realized similarly.

We constructed a big wing crane origami as a nontrivial application of the newly defined inside reverse fold. The design is coordinate-free yet allows for the freedom of choice of wing angles and



(a) $\mathcal{O}_{53}$: creae X1F3 (b) $\mathcal{O}_{57}$: by insert reverse fold (c) $\mathcal{O}_{61}$: right leg construction

Figure 9: Leg construction



(a) $\mathcal{O}_{61}$ (b) $\mathcal{O}_{63}$ (c) $\mathcal{O}_{67}$ (d) $\mathcal{O}_{68}$

two positions of bending points of a bill. The construction proceeds as step-by-step fold operations described in the Orikoto origami programming language. Thus, creating a big wing crane is purely an algorithmic process. Furthermore, the added feature of texture mapping of our system, Eos, makes the final product more original artwork.

The algorithm's implementation in the earlier paper [1] was extensively tested and extended for more capabilities. The newer Eos version and the construction program are published at the website [9].

References

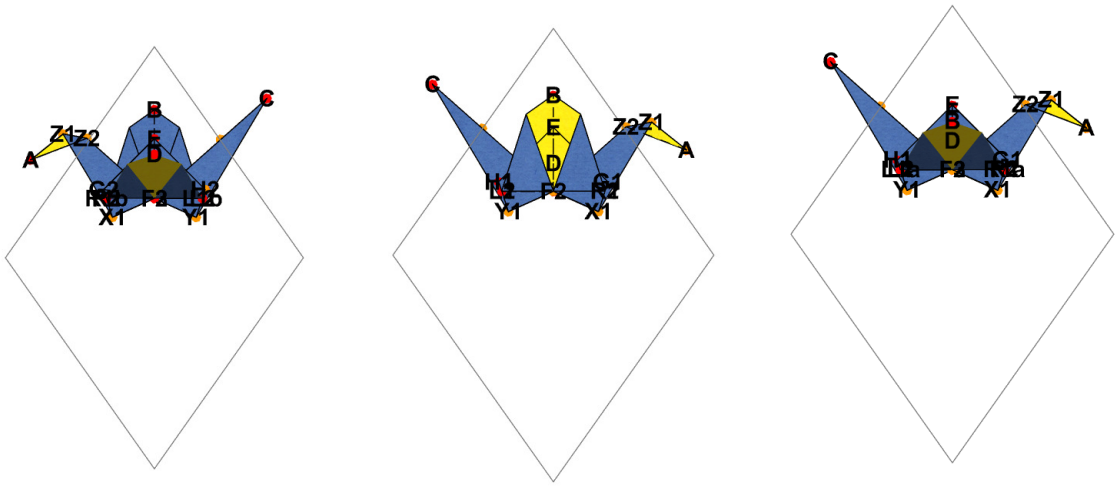
[1] T. Ida, H. Takahashi, A new modeling of classical folds in computational origami, in: P. Janičić, Z. Kovács (Eds.), Proceedings of the 13 th International Conference on Automated Deduction in Geometry, volume 352 of *EPTCS*, Elsevier Inc., 2021, pp. 41–53.

[2] H. Huzita (Ed.), Proceedings of the First International Meeting of Origami Science and Technology, Ferrara, Italy, 1989.

[3] J. Justin, Résolution par le pliage de l'équation du 3e degré et applications géométriques, L'Ouvert (1986) 9 – 19.

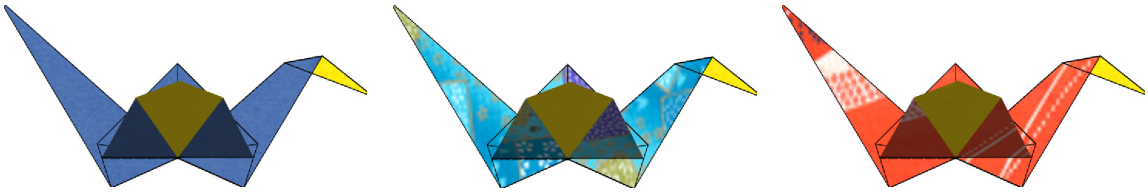
[4] K. Fushimi, M. Fushimi, Geometry of Origami, Nippon Hyoron sha Co. Ltd., 1979. (in Japanese).

[5] T. Ida, An introduction to Computational Origami, Texts and Monographs in Symbolic Computation, Springer International Publishing Switzerland, 2020.



(a) $\mathcal{O}_{69}$: by FO (b) $\mathcal{O}_{70}$: by turn over (c) $\mathcal{O}_{71}$: by FO

Figure 10: Opening wings



(a) final (b) textured crane (c) another textured crane

Figure 11: Monotone and textured cranes

[6] M. Bezem, J. W. Klop, Abstract reduction systems, volume Term Rewriting Systems, Cambridge University Press, 2003, pp. 7 – 23.

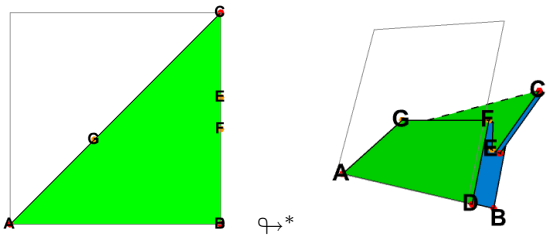
[7] T. Ida, D. Tepeneu, B. Buchberger, J. Robu, Proving and Constraint Solving in Computational Origami, in: Proceedings of the 7th International Symposium on Artificial Intelligence and Symbolic Computation (AISC 2004), volume 3249 of *Lecture Notes in Artificial Intelligence*, 2004, pp. 132–142.

[8] Wolfram Research, Inc., Mathematica, 2023.

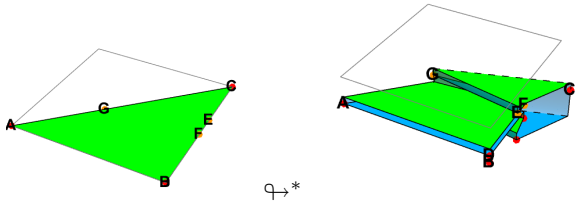
[9] Eos project, 2024. URL: https://www.i-eos.org.

A. Classical folds realizable by cut-and-glue technique

- Squash fold
  SquashFold = InserReverseFold; (O1)
- Inside crimp pleat fold
- Outside crimp pleat fold

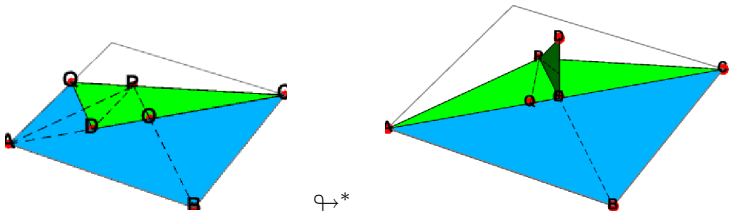


(a) before $\rightsquigarrow^*$ (b) after



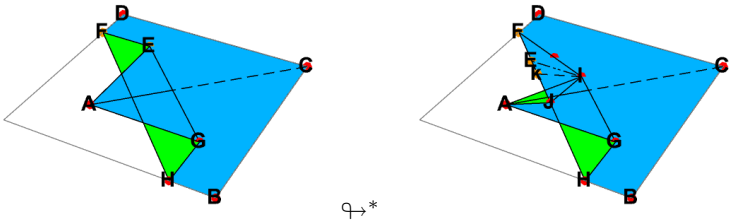
(a) before $\rightsquigarrow^*$ (b) after

- Rabbit ear fold



(a) before $\rightsquigarrow^*$ (b) after

- Swivel fold



(a) before $\rightsquigarrow^*$ (b) after