**Proceedings of the ASME 2014 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference**
**IDETC/CIE 2014**
**August 17-20, 2014, Buffalo, New York, USA**

# DETC2014/MESA-35556

# FOLDING RIGID ORIGAMI WITH CLOSURE CONSTRAINTS

**Zhonghua Xi**
MASC Group
Department of Computer Science
George Mason University
Fairfax, Virginia 22030
Email: zxi@gmu.edu

**Jyh-Ming Lien**
MASC Group
Department of Computer Science
George Mason University
Fairfax, Virginia 22030
Email: jmlien@cs.gmu.edu

## ABSTRACT

Rigid origami is a class of origami whose entire surface remains rigid during folding except at crease lines. Rigid origami finds applications in manufacturing and packaging, such as map folding and solar panel packing. Advances in material science and robotics engineering also enable the realization of self-folding rigid origami and have fueled the interests in computational origami, in particular the issues of *foldability*, i.e., finding folding steps from a flat sheet of crease patterns to desired folded state. For example, recent computational methods allow rapid simulation of folding process of certain rigid origamis. However, these methods can fail even when the input crease pattern is extremely simple. This paper attempts to address this problem by modeling rigid origami as a *kinematic system with closure constraints* and solve the foldability problem through a randomized method. Our experimental results show that the proposed method successfully fold several types of rigid origamis that the existing methods fail to fold.

## 1 Introduction

Paper folding, also known as origami, has many practical applications and rich mathematical properties beyond its artistic forms. In robotics, origami is used by researchers as a mean to gain deeper understanding of motion control and planning algorithms. Examples include robot manipulator performing fine-motor tasks of folding origami [1], carton [2] or cloth [3]. However, it is not until recently, the ideas of active materials and self-folding sheets promote the passive role of origami in these robotics systems to a more active one. Advances in material science and robotics engineering accelerate the development of *self-folding origami* that fold either by reacting to various stimuli such as light [4], heat and magnetic fields [5] or via the micro-thick folding actuators [6]. Their applications include surgical instruments for minimally invasive surgery, where there is a need for very small devices ($\approx$1 mm) that can be deployed inside the body to manipulate tissue [7]. An example of a self-folding origami is illustrated in Fig. 1.

Designing self-folding origami that can resume or approximate a single or multiple target shapes requires careful *foldability* analysis of the target shapes. That is, given a crease pattern, can this pattern be folded into a desired 2d or 3d shape? The foldability problem can usually be addressed in two steps. First, does there exist an angle assignment of the crease pattern that maps an unfolded sheet to the target shape? Second, if such a mapping does exist, what is the folding process, i.e., a sequence of angle assignments, that brings the crease pattern from the unfolded state continuously to the folded state. In this paper, we will focus on the second part of the foldability problem.

Due to limitations in the current design, most self-folding origamis are rigid. Rigid origami, a subclass of origami, can be considered as a type of mechanical linkage that uses flat rigid sheets joined by hinges. Rigid origami has many practical uses, ranging from folding maps and airbags to packing large solar panel arrays for space satellites and folding space telescope. To address the foldability issues of rigid origami, researchers have attempted to simulate or plan the folding motion [8–12]. These existing methods, however, are known to be restricted. For example, the work by Miyazaki et al. [10] only allows bending,

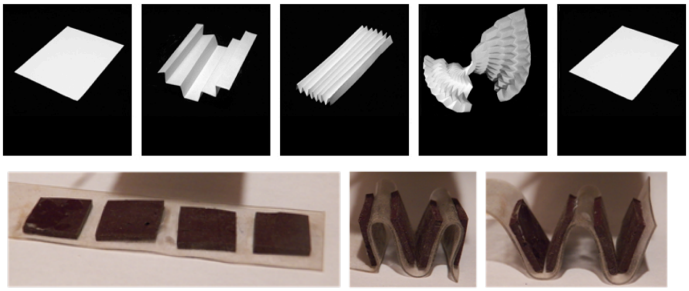

**FIGURE 1**. A multi-field responsive origami structure [5] actively folds from an initially flat sheet to complex three-dimensional shapes in response to different applied fields.

folding-up, and tucking-in motions. Balkcom's method [1] cannot guarantee the correct mountain-valley assignment for each crease. The well-known Rigid Origami Simulator by Tachi [12] may produce motion with self-intersection and can be trapped in a local minimum.

This paper attempts to address the drawbacks in existing methods by modeling rigid origami as a *kinematic system with closure constraints* and solve the foldability problem through a randomized method. We observe that many rigid origamis display noticeable differences in their folding motions in the early stage and in the rest of the folding process, based on this observation, our method iteratively expands the search by adaptively adjusting the randomness and the desire of moving closer to the goal. Our experimental results show that the proposed method successfully fold several types of rigid origamis that the existing methods fail to fold.

## 2 Rigid Origami Model

### 2.1 Crease Pattern

In this paper, we use crease pattern to represent the rigid origami model. A crease pattern is a straight-edged graph embedded in the plane as shown in Fig. 2. An edge of this graph correspond to the location of a crease line in an unfolded sheet of paper. A crease can be either *mountain folded* or *valley folded*. A mountain fold forms a convex crease at top with the paper beside the crease folded down. On the other hand, a valley fold forms a concave crease with both sides folded up. An example of mountain and valley folds is shown in Fig. 3.

### 2.2 Vertices in Crease Pattern

Vertices in the crease pattern can be categorized into two groups: real vertices and virtual vertices. Vertices on the boundary of the paper are considered as virtual vertices and they will not act

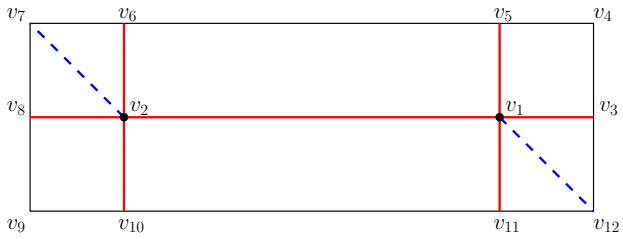


**FIGURE 2**. An example of multi-vertex crease pattern. Mountain creases are shown as solid lines in red, valley creases are show as dashed lines in blue.

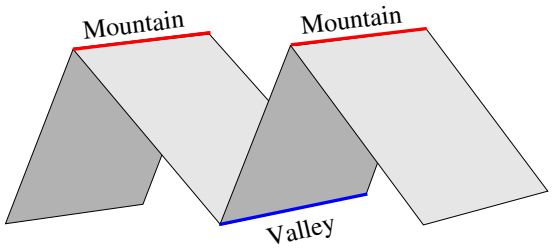


**FIGURE 3**. A crease can be folded as either a mountain fold (in red) or a valley fold (in blue).

as vertices for the purpose of our results. All other vertices on the paper are considered as real vertices. For example, vertices $v_1$ and $v_2$ in Fig 2 are real vertices and all the other vertices are virtual vertices.

### 2.3 Crease Lines and Faces

We use $l_{(i,j)}$ to denote a crease line that connects real vertex $v_i$ and vertex $v_j$ which can be either real or virtual, and we use $\rho_{(i,j)}$ to denote the folding angle of $l_{(i,j)}$. We use $F_{(i,j,\ldots)}$ to refer the face that is on the left of $\overrightarrow{l_{(i,j)}}$ (e.g., $F_{(1,3,..)}$ is refer to $F_{(1,3,4,5)}$ in Fig 2).

For real vertex $v_i$, we sort the crease lines $l_{(i,j_t)}$ in the order of the *plane angle* $\alpha_{(i,j)}$ which is the angle between x-axis and $\overrightarrow{l_{(i,j)}}$. We use $c_i$ to denote the number of crease lines that are connected to $v_i$. For instance, in Fig 2, for real vertex $v_1$, $c_1 = 5$, the sorted crease lines are $\{l_{(1,3)}, l_{(1,5)}, l_{(1,2)}, l_{(1,11)}, l_{(1,12)}\}$.

### 2.4 Configuration

We use the folding angles of all crease lines as variables to represent the configuration of an origami model. More specifically, we define a configuration $\mathscr{C} = \{\rho_{(i_1,j_1)}, \rho_{(i_2,j_2)}, \cdots, \rho_{(i_n,j_n)}\}$ for an origami with $n$ crease lines, where $\rho_{(i_k,j_k)}$ is the dihedral angle of two faces that connected by the crease line $l_{(i_k,j_k)}$ on the folded

shape. In this paper, $\rho$ is bounded in [-$\pi$, $\pi$] for origami, otherwise adjacent faces will penetrate each other during the folding process. In the real world, the range of $\rho$ is further limited by the material (e.g., $\pi/2$ for DE material in [5]). By given the limitation, we are able to simulate the maximum foldable shape when folding are done with real material, an example of using DE material is shown in Fig.7(k), flat-folded shape with ideal paper is shown in Fig.7(j) for comparison. Given a crease pattern, the shape of the folded origami can be determined by the configuration $\mathscr{C}$.

## 3 Related Work

Origami, the art of paper folding, found its root in Japan around the 17th century and became popular worldwide near the mid-1900s. However, not until 1990s, researchers become increasingly interested in its rich mathematical properties. In the rest of this section, we will discuss related work on generating origami folding motion.

**Planning and simulating origami motion.** In 1996, Miyazaki et al. [10] simulates origami folding by a sequence of simple folding steps, including bending, folding up, and tucking in. It is easy to reconstruct an animation from a sheet of paper to the final model. However, the simplicity of folding steps limits the types of origami models that could be represented in the system. Consequently, this method is not suitable for many complex origami models whose folding process cannot be represented as simple folding steps such as the Miura pattern shown in Fig. 6(g). Song et al. [13] presented a probabilistic-roadmap-method (PRM) based framework for studying folding motion. However, their kinematic representation of origami is a *tree-structure model* whose folding angle of each crease line is independent of other crease lines. Although tree-structure model greatly simplifies the folding map that can be easily defined along the path from base to each face, this model is not applicable to represent the majority of the origami, such as the one shown in Fig. 6(g), due to their *closure constraints*. Balkcom [1] proposed a simulation method based on the ideas of virtual cutting and combination of forward and inverse kinematics using a rigid origami model. Although this approach is computational efficient, it cannot guarantee the correct mountain-valley assignment for each crease, i.e., a mountain fold can become a valley fold or vice versa. More recently, Tachi [12] proposed an interactive simulator for rigid origami model (known as Rigid Origami Simulator (ROS)) which generates folding motion of origami by calculating the trajectory by projection to the constrained space based on rigid origami model, global self-intersection avoidance and stacking order problems are not considered in his work. Perhaps the work closest to our approach proposed in this paper is by An et al. [6]. They proposed a new type of self-reconfiguration system called self-folding sheet. They first construct the corresponding folded state for a given crease pattern and angle assignment then continuously unfold the paper using local repulsive energies (via a modification of ROS [12]). By reversing the unfolding sequence, they obtained the path starting from a flat sheet and ending with the desired folded state.

**Planning under closure constraints.** Although there exists little work on origami motion planning, there have been many methods proposed to plan motion for articulated robots under closed-chain constraints [14–17]. Interestingly, we see many similar ideas used in both closed-chain systems and origami folding. For example, gradient decent was used by [12] for rigid origami simulation and by [14] for generating valid configuration of a closed-chain system. Another example is inverse kinematics, which plays the central role both in Balkcom's simulator [1] and in constructing the so-called *kinematic roadmap* [16, 18] for capturing the topology of free configuration space. Tang et al. [19] proposed an efficient sampling-based planner for spatially constrained systems. By sampling in the reachable distance space in which all configurations lie in the set of constraint-satisfying subspaces and using a local planner, they can significantly reduce the computation time for finding a path.

## 4 Randomized Rigid Origami Folding

Before we discuss our planning method in more detail, we would like to point out that, even for simple crease patterns (such as those contains only a single real vertex shown in Fig. 6), the folding trajectory is often nonlinear. Fig. 4(a) shows a plot of the folding angles for each crease from a flat sheet to a completely folded Miura origami (Fig. 6(g)).

Similar to the problem faced in systems with closure constraints, traditional motion planners that perform local planning using linear interpolation usually fail to connect two seemingly nearby configurations. Moreover, we observe that, for rigid origami, whose folding angles are highly constrained by each other, its folding pathway has very distinct characteristics between the early folding stage and the rest of the folding process. That is, there are abundant valid configurations when the origami is still flat, however, once the folding process started, the folding pathway quickly becomes very narrow and highly non-linear due to the closure constraints. This difference can be observed from the smoothness of the trajectories shown in Fig. 4(a) and Fig. 4(b). The former is much smoother than the latter. As we will see later in this section, these observations play important roles in designing our randomized folding algorithm.

In the rest of this section, we will first discuss the necessary conditions for a configuration to be foldable and feasible in Section 4.1. Then, we will briefly talk about how to fold the crease

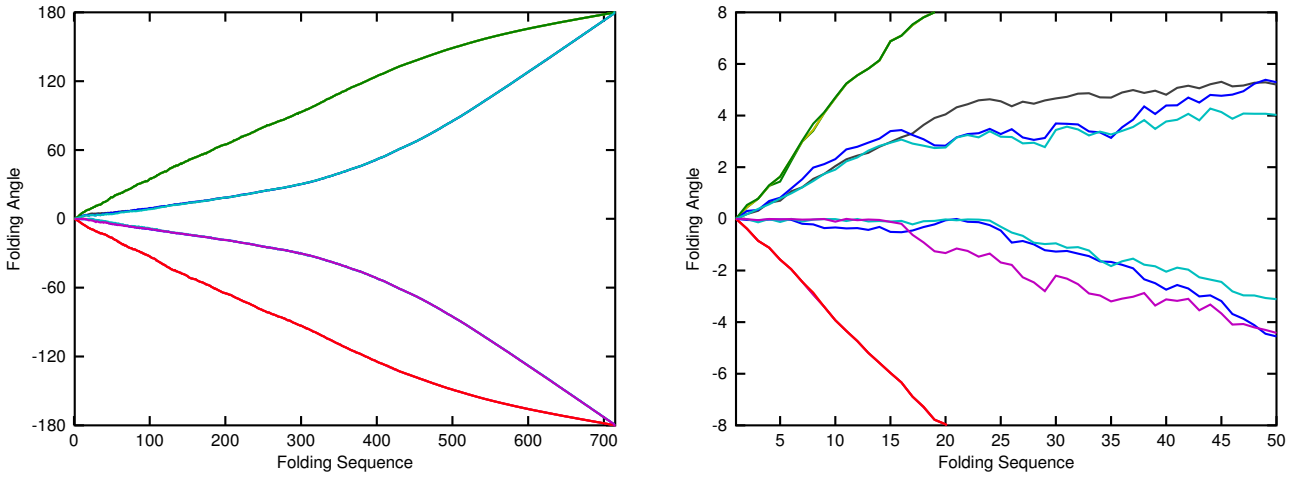


**FIGURE 4**. (a) Trajectories of folding angles of 12 crease lines for Miura crease pattern shown in Fig. 6(g). Path was found by the proposed method. (b) The first 50 steps of the folding trajectories.

pattern from a valid angle assignment using a folding map in Section 4.2. Finally, we describe our randomized rigid origami folding method in Section 4.3.

### 4.1 Foldable and Feasible Configurations

Given a configuration $\mathscr{C} = \{\rho_{(i_1,j_1)}, \rho_{(i_2,j_2)}, \cdots, \rho_{(i_n,j_n)}\}$, we can classify $\mathscr{C}$ according to its *foldability* and *feasibility*. First, let $v_i$ be a real vertex in a foldable multi-vertex crease pattern and let $B_i$ be the $4 \times 4$ matrix which translates a point in $\Re^3$ by $v_i$. For crease line $l_{(i,j)}$, let $A_{(i,j)}$ be the matrix in homogeneous coordinates which rotates the $xy$-plane by plane angle $\alpha_{(i,j)}$ (the angle needed to rotate in CCW that can make positive x-axis overlap with the crease line $\overrightarrow{l_{(i,j)}}$), and let $C_{(i,j)}$ be the matrix in homogeneous coordinates which rotates by folding angle $\rho_{(i,j)}$ in the $yz$-plane. Then the folding matrix for the crease line $l_{(i,j)}$ around $v_i$ will be $\chi_{((i,j),i)} = B_i A_{(i,j)} C_{(i,j)} A_{(i,j)}^{-1} B_i^{-1}$. If we pick $F_{(i,j_{c_i},\ldots)}$ (where $c_i$ is the number of crease lines around $v_i$) as $F_0$ which will be fixed it in the $xy$-plane during folding and multiply the folding matrices though all crease lines that are around $v_i$ in order of their plane angles $\alpha_{(i,j)}$, then

$$\prod_{t=1}^{c_i} \chi_{((i,j_t),i)} = I \tag{1}$$

These necessary conditions of foldability for multi-vertex rigid origami were first discovered by Balcastro and Hull in 2002 [11]. Let us take the multi-vertex crease pattern shown in Fig. 2 as an example to illustrate Eq. (1). For real vertex $v_1$ and $v_2$ the following equations should hold respectively.

$$\chi_{((1,3),1)}\chi_{((1,5),1)}\chi_{((1,2),1)}\chi_{((1,11),1)}\chi_{((1,12),1)} = I$$
$$\chi_{((2,1),2)}\chi_{((2,6),2)}\chi_{((2,7),2)}\chi_{((2,8),2)}\chi_{((2,10),2)} = I$$

There are several properties that the folded paper should have:

1. unstretchable,
2. flat (planar) for all faces,
3. free of self-intersection,

A foldable configuration $\mathscr{C}_{foldable}$ only guarantees the first two properties. In order to have a valid configuration $\mathscr{C}$ that satisfy all three of these properties, we need to check if $\mathscr{C}$ is free of self-intersection. In order to do so, we will need a folding map for each face. A folding map is a function that map a point in $\Re^2$ to the corresponding point of folded state for a given foldable configuration $\mathscr{C}_{foldable}$ in $\Re^3$. With the folding map for all faces, we can fold the paper to the foldable configuration $\mathscr{C}_{foldable}$ and perform collision detection to check the feasibility of $\mathscr{C}_{foldable}$.

### 4.2 Folding Map

Let $F_0$ be an arbitrary face that will be fixed in the $xy$-plane during the entire folding process, and given another face $F_{(i_p,j_p,\ldots)}$, let $\gamma$ be any *vertex-avoiding path* starting from a point in $F_0$ and ending at a point in $F_{(i_p,j_p,\ldots)}$ by cross some crease lines. We say that a path is vertex-avoiding if it does not intersect with

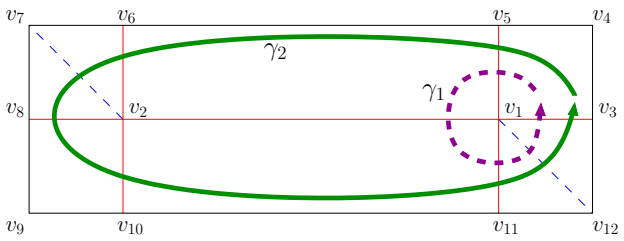


**FIGURE 5**. A multi-vertex crease pattern with two vertex-avoiding paths.

any vertices. Let the crease lines that $\gamma$ crossed be, in order, $l_{(i_1,j_1)},...,l_{(i_p,j_p)}$. Then the folding map for $(x,y) \in F_{(i_p,j_p,...)}$ is:

$$f(x,y) = f(x,y,0,1) = \prod_{k=1}^{p} \chi_{((i_k,j_k),i_k)}(x,y,0,1) \qquad (2)$$

Note that the folding map is independent of the path $\gamma$. This means no matter which path we choose, the folding map remains the same. This property was first proved in [11]. Thus, we can pick an arbitrary path $\gamma$ from $F_0$ to $F_{(i_p,j_p,...)}$ and compute the production of rotation matrices as the folding map for $F_{(i_p,j_p,...)}$.

Let us take Fig.5 as an example to illustrate Eq. (2). We first pick an arbitrary face, say $F_{(1,3,4,5)}$ as $F_0$. Suppose that we find two paths $\gamma_1$ and $\gamma_2$ from $F_0$ to $F_{(1,11,12)}$ shown as dashed curve in magenta and solid curve in green, respectively, in Fig.5. The folding map for $F_{(1,11,12)}$ can be $\chi_{((1,5),1)}\chi_{((1,2),1)}\chi_{((1,11),1)}$ if we follow the path $\gamma_1$ or $\chi_{((1,5),1)}\chi_{((2,6),2)}\chi_{((2,7),2)}\chi_{((2,8),2)}\chi_{((2,10),2)}\chi_{((1,11),1)}$ if $\gamma_2$ was chosen. We can pick either $\gamma_1$ or $\gamma_2$ for computing the folding map for face $F_{(1,11,12)}$ since the results will be the same if the given configuration is foldable.

By using the folding map computed from a given configuration, we can *instantaneously* fold a crease pattern to a desired folded shape (could be either flat or non-flat). The problem of how we can find the intermediate configurations remains.

### 4.3 Folding via Randomized Search

Finding a path in a highly constrained high-dimensional configuration space is always challenging. We propose to repetitively sample a configuration $\mathscr{C}_\tau$ randomly near the best configuration known to us $\mathscr{C}_\triangle$ so far, and use a non-linear optimization approach to find a valid configuration locally around $\mathscr{C}_\tau$. Our approach only expands the closest configuration to the goal and uses an adaptive weight adjustment to balance between randomness and the desire to move towards to the goal. Details of the proposed method are described in Algorithm 1. Note that, in Algorithm 1, $W_0$, $W_1$, $W_2$ and $D$ are user defined parameters, and *weight* is bounded in $[0,1]$. In this paper, unless noted otherwise, we consistently set these parameters to: $D = 0.02$, $W_0 = 0.8$, $W_1 = 0.2$, and $W_2 = 0.01$. See detailed discussion in Section 5 on the values of these parameters.

**Algorithm 1** Randomized Rigid Origami Folding

**Input:** Start configuration $S$, goal configuration $G$
**Output:** Foldable and feasible path from $S$ to $G$

1: $weight \leftarrow W_0$
2: $\mathscr{C}_\triangle \leftarrow S$
3: **while** $G$ not reached **do**
4: $\quad \mathscr{C}_{rand} \leftarrow$ a random configuration
5: $\quad \overrightarrow{dir} \leftarrow (1 - weight) \cdot \mathscr{C}_{rand} + weight \cdot G$
6: $\quad \mathscr{C}_\tau \leftarrow \mathscr{C}_\triangle + D \cdot \overrightarrow{dir}$
7: $\quad \mathscr{C} \leftarrow$ FINDFOLDABLE($\mathscr{C}_\tau$) ▷ Section 4.3.1
8: $\quad$ **if** IsValid($\mathscr{C}$) and $\mathscr{C}$ is closer to $G$ **then** ▷ Section 4.3.2
9: $\quad\quad \mathscr{C}_\triangle \leftarrow \mathscr{C}$
10: $\quad\quad weight \leftarrow weight + W_1$
11: $\quad$ **else**
12: $\quad\quad weight \leftarrow weight - W_2$
13: $\quad$ **end if**
14: **end while**

Algorithm 1 first initializes the planner by setting the *weight* to $W_0$, and set the closest configuration $\mathscr{C}_\triangle$ to $S$. In each step, Algorithm 1 samples a random configuration $\mathscr{C}_{rand}$ and find a direction $\overrightarrow{dir}$ by linearly combining $\mathscr{C}_{rand}$ and $G$ with corresponding weights, $1 - weight$ and $weight$, respectively. Then, a new configuration $\mathscr{C}_\tau$ is created by moving $\mathscr{C}_\triangle$ forward distance $D$ along $\overrightarrow{dir}$. However, even if $D$ is a tiny number, the target configuration $\mathscr{C}_\tau$ is usually unfoldable. Thus, we introduced the function FINDFOLDABLE for finding a foldable configuration $\mathscr{C}$ around $\mathscr{C}_\tau$. If $\mathscr{C}$ is feasibly and it is closer to the goal $G$ than $\mathscr{C}_\triangle$, Algorithm 1 replaces $\mathscr{C}_\triangle$ by $\mathscr{C}$. We use Euclidean distance to measure how far away two configurations are, however, other metric (e.g., 1-norm distance or infinity norm distance) can also be used. Algorithm 1 repeats this process until $G$ is reached. The value of *weight* will be adjusted adaptively during the process.

**4.3.1 Finding Foldable Configuration** Non-linear optimization (NLOPT) is used to find a foldable configuration in function FINDFOLDABLE. Given a configuration $\mathscr{C}_\tau$, FINDFOLDABLE pushes $\mathscr{C}_\tau$ to a foldable configuration $\mathscr{C}$ near $\mathscr{C}_\tau$ by minimize the objective function shown in Eq. (3).

$$F(\mathscr{C}) = \sum_{i=1}^{n_v} \left| \prod_{k=1}^{c_i} \chi((i,j_k),i) - I \right| , \qquad (3)$$

where $n_v$ is the number of real vertices, $c_i$ is the number of crease lines incident to the real vertex $v_i$, and, finally, $(i, j_k)$ is the $k$-th crease lines around $v_i$.

More specifically, for a given real vertex $v_i$, we want the production of rotation matrices of crease lines around $v_i$ to be as close to an identity matrix as possible. Since, in a foldable configuration, each real vertex in the crease pattern should be an identity matrix shown in Eq. (1). Special treatment for the stationary face $F_0$, which is fixed in the $xy$-plane, is required. The folding map of $F_0$ shown in (2) should always be an identity matrix regardless which closed vertex-avoiding loop $\gamma$ is used for computing the folding map, otherwise $F_0$ will no longer stay in the $xy$-plane. Note that, NLOPT might still return an unfoldable configuration due to that maximum iteration has been exceeded or no foldable configuration exists around the configuration $\mathscr{C}_\tau$. These unfoldable configurations can be filtered out by performing the foldability check on the returned configuration.

**4.3.2 Detecting Invalid Configuration** A foldable configuration might still be an invalid configuration due to self-intersection. Thus a collision checking is applied after the configuration $\mathscr{C}$ is returned by FINDFOLDABLE. Local intersection is avoided by bounding the folding angle in $[-\pi, \pi]$ for each crease line. Global intersection is avoided by applying collision detection between faces of the origami. In our implementation, we checking collision on all pairs of faces and we say an origami has self-intersection if penetration is detected while face overlapping is considered as valid.

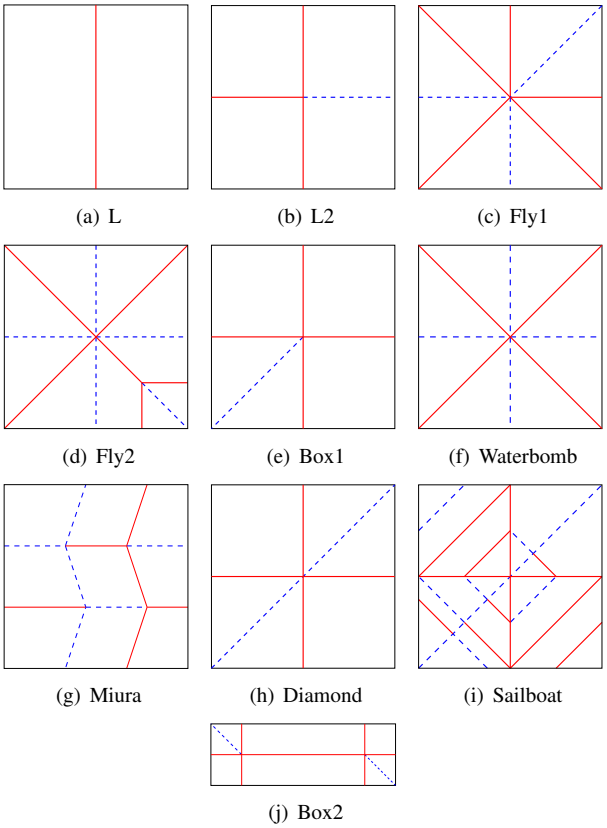


**FIGURE 6**. Crease patterns used in our experiments. Mountain creases are shown as solid lines in red, valley creases are show as dashed lines in blue.

## 5 Experiment Results

### 5.1 Environment Setup

We implemented the proposed method in C++ using GNU NLopt library. Triangulated crease patterns used in our experiments are shown in Fig. 6 and their folded states are shown in Fig. 7. Degree of freedom (DOF) of the origami ranges from 2 (L pattern) to 34 (Sailboat pattern). All the experiment data are collected on a workstation with 2.30GHz Intel Xeon E5-2630 CPU and 32GB memory. Unless stated otherwise, the parameters in Algorithm 1 are set to: $D = 0.02$, $W_0 = 0.8$, $W_1 = 0.2$, $W_2 = 0.01$. The maximum iterations for NLOPT is 1000 and the maximum number of samples is 100000. Every data point reported in the tables and plots in this section is an average over 10 runs.

While the source code will be made available after the current work is published, we provide an interactive web-based tool at: `http://masc.cs.gmu.edu/jsobj/origami.html` We also strongly encourage the reader to review the submitted videos for better visualization of the folding process. Two folding sequences produced by the proposed method are shown in Figs. 8 and 9.

### 5.2 Running Time

The proposed method is able to fold all the crease patterns in Fig. 6. Experimental results are shown in Table 1, which includes running time in seconds, number of configurations sampled by the planner, number of valid configurations, and the number of configurations sampled and tested by NLOPT to find foldable configurations (labeled as "Tested" in the table). Note that the total number of valid configurations is also the number of intermediate states connecting start and goal configurations, and the number of tested configurations is much greater than that of sampled configurations. In general, the running time is strongly correlated to the number of tested configurations by NLOPT and the DOF of the origami.

Because our planner is parameterized by expanding distance $D$ and goal-bias weights $W_i$, we study how the values of $D$ and weights affect the performance of our algorithm. Results ob-

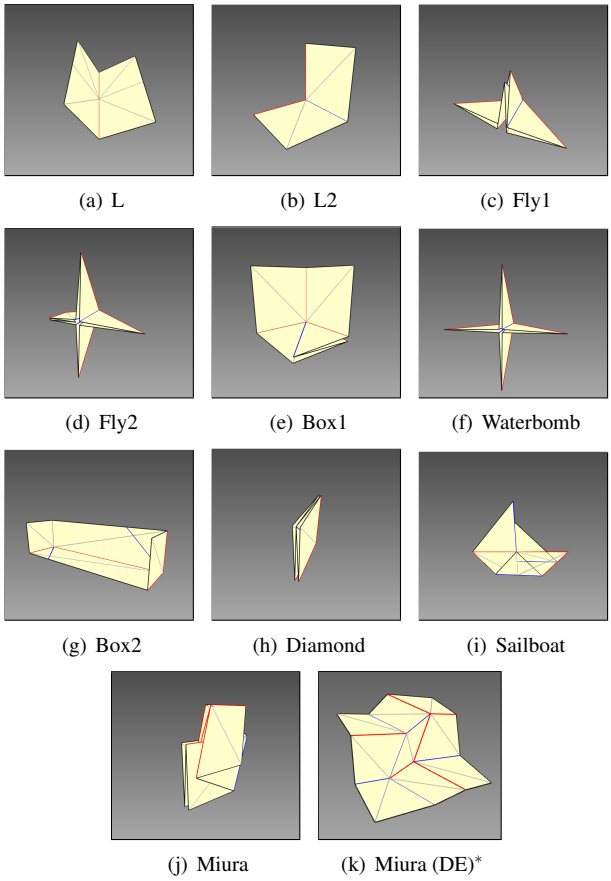


**FIGURE 7**. Folded states of crease patterns shown in Fig. 6. Note that some of the models do not fold completely for the sake of better visualization.*Folding with DE material, which has a maximum folding angle of $\pi/2$.

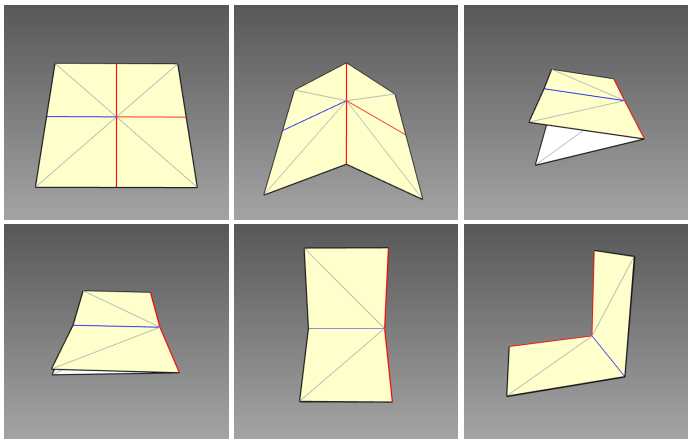

**FIGURE 8**. Folding process of L2 pattern shown in Fig. 6(b)

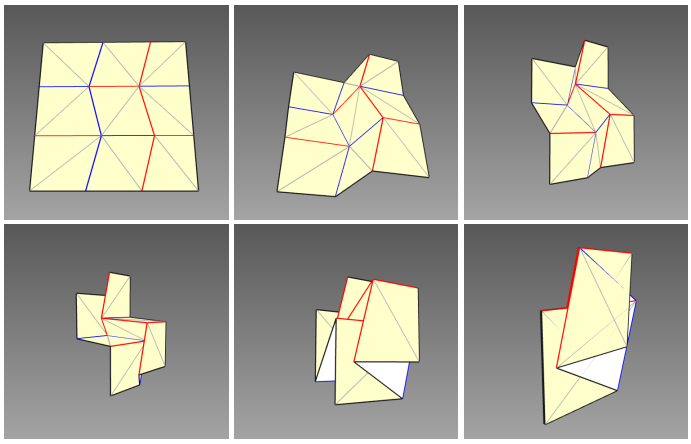

**FIGURE 9**. Folding process of Miura pattern shown in Fig. 6(g)

tained by varying $D$ are shown in Fig. 10(a). Notice that the $y$-axis of Fig. 10(a) is the *relative running time*, which is defined as $\frac{T_D}{T_{base}}$, where $T_D$ is the running time of a given value of $D$ and $T_{base}$ is the baseline running time reported in Table 1 when $D = 0.02$. We observe that larger $D$ helps to find a path more efficiently as expected. However, if $D$ becomes too large, the computational efficiency will drop on models whose folding trajectories are highly non-linear such Box2 and Miura. Note we assume that $D$ is always small enough so the subpath from the best known configuration $\mathscr{C}_\triangle$ to a new valid configuration $\mathscr{C}$ is always valid. The assumption is relaxed by checking if we can find a valid configuration from the midpoint of $\mathscr{C}_\triangle$ and $\mathscr{C}$ (via NLOPT and the self-collision detector discussed earlier).

To show the role played by goal-bias weights in the proposed planner, we conducted experiments over various static weights $W$. We disabled the *adaptive weight adjustment* in lines 10 and 12 of Algorithm 1 in order to to highlight the influence of a given weight, i.e., $W_0 = W$ and $W_1 = W_2 = 0$. Results are shown in the plot in Fig. 10(b). Note that the relative running time on $y$-axis is in logarithmic scale, and the relative running time, which is defined as $\frac{T_W}{T_{base}}$, where $T_W$ is the running time of a static weight $W$ and $T_{base}$ is the baseline running time reported in Table 1 using adaptive weight adjustment. We can see when $W$ is zero, i.e., without given any bias to the goal, our planner has difficulty of finding a path within the given number of samples due to the sparsity of high dimensional space. When $W = 1$, i.e., the planner always expand the search to the goal, it can be easily trapped at a dead-end. Overall, static weights between 0.4 and 0.8 give the best performance. However, if we compare the results with static weights to the baseline results shown in Table 1, we can conclude that adaptively adjusting *weight* (i.e., via lines 10 and 12 of Algorithm 1) does provide much better performance.

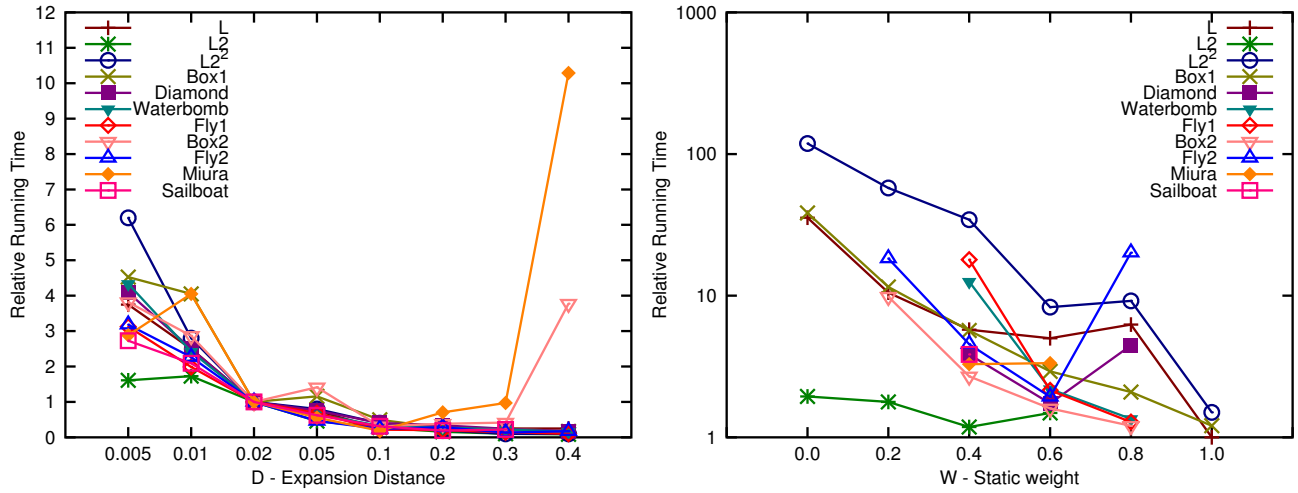


**FIGURE 10**. (a) Relative running time over various expansion distance $D$. The relative running time is defined as $\frac{T_D}{T_{base}}$, where $T_D$ is the running time of a given value of $D$ and $T_{base}$ is the baseline running time reported in Table 1 when $D = 0.02$. (b) Relative running time over various static weight $W$. The relative running time is defined as $\frac{T_W}{T_{base}}$, where $T_W$ is the running time of a static weight $W$ and $T_{base}$ is the baseline running time reported in Table 1 using adaptive weight adjustment.

**TABLE 1**. Running Time

| Model | DOF | Time (s) | Sampled | Valid | Tested |
|---|---|---|---|---|---|
| L | 2 | 0.004 | 110 | 110 | 156 |
| L2 | 4 | 8.317 | 40323 | 787 | 2047635 |
| $L2^2$ | 4 | 0.010 | 331 | 331 | 407 |
| Box1 | 6 | 0.101 | 250 | 250 | 15181 |
| Diamond | 6 | 0.514 | 1074 | 390 | 65033 |
| Waterbomb | 8 | 0.613 | 731 | 359 | 54046 |
| Fly1 | 8 | 0.631 | 605 | 421 | 53613 |
| Box2 | 9 | 1.463 | 864 | 358 | 126885 |
| Fly2 | 11 | 2.425 | 864 | 460 | 127064 |
| Miura | 12 | 13.564 | 1813 | 677 | 511050 |
| Sailboat | 34 | 83.923 | 6103 | 1970 | 1396032 |

(Note: $L2^2$ is L2 with an intermediate state defined by user)

### 5.3 Reverse Search

An et al. [6] find folding path by continuously unfolding the model from the folded state. Since the flatten model has more flexibility which makes path planning hard or even not able to fold the origami to desired state from an unfolded sheet of paper.

Reverse search can be beneficial especially if certain folding order is required, such as the L2 shown in Fig. 6(b). Reverse search provides a strong heuristic towards to the goal (unfolded state) and dramatically reduces the search space which can not only improve the efficiency but also increase the chance to find a path. To test this observation, we conducted experiments using reverse search and the results are shown in Table 2. As we can see, for most of the models, the running time remains almost identical or even slower, however, reverse search does have a huge improvement on models that has implicit folding orders such as L2 and Sailboat.

Let us take another view to exam the idea of reverse search using Fig. 4(a) (Miura origami). As we can see from Fig. 4(a), the trajectories of folding angles are quite smooth especially for the portion between the middle part and the goal. However, if we zoom in to the first 50 steps of the trajectories shown in Fig. 4(b), we can see that there are quite a lot of turbulences at the very beginning. This indicates that the paper has greater flexibility when it is almost flat. It is exactly this difference between the beginning and the end of the folding process that motivates An et al. [6] to obtain the folding path by unfolding because existing folding tools, such as [12], often get lost at the beginning of the folding process. Using randomized search and adaptive weight adjustment, our proposed method is able to find these implicit patterns in Miura origami just after 100 steps without given any hint on the relationships of the creases.

**TABLE 2**. Running Time using Reverse Search

| Model | Time (s) | **Speedup**(x) | Sampled | Valid | Tested |
|---|---|---|---|---|---|
| L | 0.006 | 0.667 | 108 | 108 | 149 |
| L2 | 1.936 | 4.296 | 6123 | 473 | 352462 |
| $L2^2$ | 0.019 | 0.526 | 439 | 439 | 488 |
| Box1 | 0.167 | 0.605 | 291 | 291 | 17790 |
| Diamond | 0.203 | 2.532 | 374 | 371 | 24517 |
| Waterbomb | 0.444 | 1.381 | 346 | 346 | 25725 |
| Fly1 | 0.549 | 1.149 | 464 | 338 | 40694 |
| Box2 | 1.805 | 0.811 | 772 | 408 | 112541 |
| Fly2 | 1.440 | 1.684 | 489 | 452 | 73134 |
| Miura | 11.231 | 1.208 | 1577 | 638 | 457353 |
| Sailboat | 27.099 | 3.097 | 2703 | 1542 | 460593 |

(Note: $L2^2$ is L2 with an intermediate state defined by user)

### 5.4 Taking Symmetry into Consideration

Some of the crease patterns are highly symmetric, for example, the Miura crease pattern. The trajectories of the folding angles found by the proposed method shown in Fig. 4(a) also confirms the symmetry property: 12 trajectories are overlapped into only four, and there is also vertical symmetry (corresponding mountain creases and valley creases have the same folding angle, but with opposite sign), the DOF of the Miura crease pattern can be further reduced to 2. Thus, if we take symmetry into consideration, the DOF of the origami can be reduced significantly. As shown in Table 3, the running time of finding a path gains more than four times speedup when our planner considers symmetry. Although we do not detect symmetry automatically, we believe this can be approximated by checking whether the initial trajectories overlap.

**TABLE 3**. Running Time using Symmetry

| Model | DOF | | **Speedup** (x) |
|---|---|---|---|
| | before | after | |
| Diamond | 6 | 2 | 15.05 |
| Fly1 | 8 | 3 | 4.69 |
| Waterbomb | 8 | 2 | 14.67 |
| Fly2 | 11 | 4 | 11.12 |
| Miura | 12 | 2 | 13.98 |

### 5.5 Compare to Existing Works

Although there have been several existing works on simulating or planning motion of rigid origami [1, 6, 10], most of these works are only applicable to specific type of rigid origami. Tachi's Rigid Origami Simulator (ROS) [12] provides the most general solution so far and is the only publicly available software the we are aware of. Consequently, we have tested ROS extensively using the crease patterns shown in Fig. 6. However, we found that it is difficult to provide a meaningful comparison to our methods due to that both approaches focus on different objectives. The main objective of this paper is to find folding path from one configuration to another while ROS focused on folding a crease patten as much as possible. In addition, even for a very simple crease pattern, such as L2 shown in Fig. 6(b), ROS usually failed to find a valid folding path due to self-intersection. In many situations, the gradient approach causes ROS to be trapped in a local minimum due to its inability to find the implicit folding order induced by the crease pattern. Moreover, ROS does not support user defined criteria of the folding process, such as user defined intermediate states that should be reached in order (as phases for active-materials), maximum acceleration of folding speed for a partially crease (maximum toque that can be applied to that crease), etc.

## 6 Conclusions

In this paper, we proposed a randomized approach for planning the motion of rigid origami. We used a nonlinear optimization method to find a valid configuration around a given sample configuration. An adaptive bias that attracts the search process to the goal was also introduced to speed up the search process. The experimental results shows that our planner could efficiently and effectively find valid path for various types of rigid origami that existing tools fail to fold.

The proposed randomized rigid origami folding method is designed to assist the foldability analysis of self-folding origami. Self-folding origami using active-materials usually have many kinematic and dynamic constraints, such as maximum folding angles, and may often requires *multiple folding phases* in order to fold itself to the desired state, see details in [5]. User defined motion criteria can be easily introduced into the proposed framework. For example, our method supports multi-phase folding by given a sequence of valid intermediate configurations $\{\mathscr{C}_1, \mathscr{C}_2, ..., \mathscr{C}_n\}$.

**Limitations and Future Work** Even through our preliminary results are encouraging, our method still has much room for improvement. For example, results of $L2^2$ in Tables 1 and 2 is obtained by folding a two phase origami model via an intermediate state defined by user. (That is, $L2^2$ and L2 have the same crease

pattern and angle assignment but $L2^2$ folds via a given intermediate state.) We can see that by explicitly specifying the folding order, the search space could be reduced significantly and provides over 800 times speedup comparing to knowing nothing about the folding order (130 times speedup if search in reverse order). The next step of our work will focus on analyzing the folding order of a given crease pattern.

## REFERENCES


[1] Balkcom, D. J., and Mason, M. T., 2008. "Robotic origami folding". *The International Journal of Robotics Research,* ***27***(5), pp. 613–627.

[2] Yao, W., Cannella, F., and Dai, J. S., 2011. "Automatic folding of cartons using a reconfigurable robotic system". *Robotics and Computer-Integrated Manufacturing,* ***27***(3), pp. 604–613.

[3] Miller, S., Van Den Berg, J., Fritz, M., Darrell, T., Goldberg, K., and Abbeel, P., 2012. "A geometric approach to robotic laundry folding". *The International Journal of Robotics Research,* ***31***(2), pp. 249–267.

[4] Ryu, J., D'Amato, M., Cui, X., Long, K. N., Qi, H. J., and Dunn, M. L., 2012. "Photo-origami—bending and folding polymers with light". *Applied Physics Letters,* ***100***, p. 161908.

[5] Ahmed, S., Lauff, C., Crivaro, A., McGough, K., Sheridan, R., Frecker, M., von Lockette, P., Ounaies, Z., Simpson, T., Lien, J.-M., and Strzelec, R., 2013. "Multi-field responsive origami structures: Preliminary modeling and experiments". In Proceedings of the ASME 2013 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference.

[6] An, B., Benbernou, N., Demaine, E. D., and Rus, D., 2011. "Planning to fold multiple objects from a single self-folding sheet". *Robotica,* ***29***(1), pp. 87–102.

[7] Swanstrom, L., Whiteford, M., and Khajanchee, Y., 2008. "Developing essential tools to enable transgastric surgery". *Surgical endoscopy,* ***22***(3), pp. 600–604.

[8] Hull, T., 1994. "On the mathematics of flat origamis". *Congressus numerantium*, pp. 215–224.

[9] Bern, M., and Hayes, B., 1996. "The complexity of flat origami". In Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, pp. 175–183.

[10] Miyazaki, S., Yasuda, T., Yokoi, S., and Toriwaki, J.-i., 1996. "An origami playing simulator in the virtual space". *Journal of Visualization and Computer Animation,* ***7***(1), pp. 25–42.

[11] Belcastro, S.-M., and Hull, T., 2002. "A mathematical model for non-flat origami". In Origami3: Proc. the 3rd International Meeting of Origami Mathematics, Science, and Education, pp. 39–51.

[12] Tachi, T., 2009. "Simulation of rigid origami". In Origami4: Proceedings of The Fourth International Conference on Origami in Science, Mathematics, and Education.

[13] Song, G., and Amato, N. M., 2004. "A motion-planning approach to folding: From paper craft to protein folding". *Robotics and Automation, IEEE Transactions on,* ***20***(1), pp. 60–71.

[14] Yakey, J. H., LaValle, S. M., and Kavraki, L. E., 2001. "Randomized path planning for linkages with closed kinematic chains". *IEEE Transactions on Robotics and Automation,* ***17***(6), pp. 951–958.

[15] Cortes, J., Simeon, T., and Laumond, J. P., 2002. "A random loop generator for planning the motions of closed kinematic chains using PRM methods". In Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 2141–2146.

[16] Han, L., and Amato, N. M., 2000. "A kinematics-based probabilistic roadmap method for closed chain systems". In Robotics:New Directions, A K Peters, pp. 233–246. Book containts the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), Dartmouth, March 2000.

[17] Cortes, J., and Simeon, T., 2004. "Sampling-based motion planning under kinematic loop-closure constraints". In Proc. Int. Workshop Alg. Found. Robot.(WAFR). To appear.

[18] Xie, D., and Amato, N. M., 2004. "A kinematics-based probabilistic roadmap method for high dof closed chain systems". In Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, Vol. 1, IEEE, pp. 473–478.

[19] Tang, X., Thomas, S., Coleman, P., and Amato, N. M., 2010. "Reachable distance space: Efficient sampling-based planning for spatially constrained systems". *The international journal of robotics research,* ***29***(7), pp. 916–934.