An End-to-End Approach to Self-Folding Origami Structures

Byoungkwon An, Shuhei Miyashita, *Member, IEEE*, Aaron Ong, *Student Member, IEEE*, Michael T. Tolley, *Member, IEEE*, Martin L. Demaine, Erik D. Demaine, Robert J. Wood, *Senior Member, IEEE*, and Daniela Rus, *Fellow, IEEE*

***Abstract*—This paper presents an end-to-end approach to automate the design and fabrication process for self-folding origami structures. Self-folding origami structures are robotic sheets composed of rigid tiles and joint actuators. When they are exposed to heat, each joint folds into a preprogrammed angle. Those folding motions transform themselves into a structure, which can be used as body of 3-D origami robots, including walkers, analog circuits, rotational actuators, and microcell grippers. Given a 3-D model, the design algorithm automatically generates a layout printing design of the sheet form of the structure. The geometric information, such as the fold angles and the folding sequences, is embedded in the sheet design. When the sheet is printed and baked in an oven, the sheet self-folds into the given 3-D model. We discuss, first, the design algorithm generating multiple-step self-folding sheet designs, second, verification of the algorithm running in $O(n^2)$ time, where n is the number of the vertices, third, implementation of the algorithm, and finally, experimental results, several self-folded 3-D structures with up to 55 faces and two sequential folding steps.**

***Index Terms*—Cellular and modular robots, printable origami robots, self-folding, smart actuators.**

I. Introduction

FOLDING is a method to transform a device during or after fabrication. The foldings on a structure or a machine can yield the dimensional transformation of the device (see Fig. 1), such as a 2-D sheet of paper folding into a 3-D origami artwork, or the solar panels of a satellite unfolding on its orbit to make a wide 2-D surface receiving sun light. Folding is widely used for engineering applications, including space projects [1], [2], soft-robots [3], microscale fabrications [4], [5], and microrobotics [6]–[8]. Folding is also found in the nature, for example, in insect wings [9], leaves [10], [11], and proteins [12].

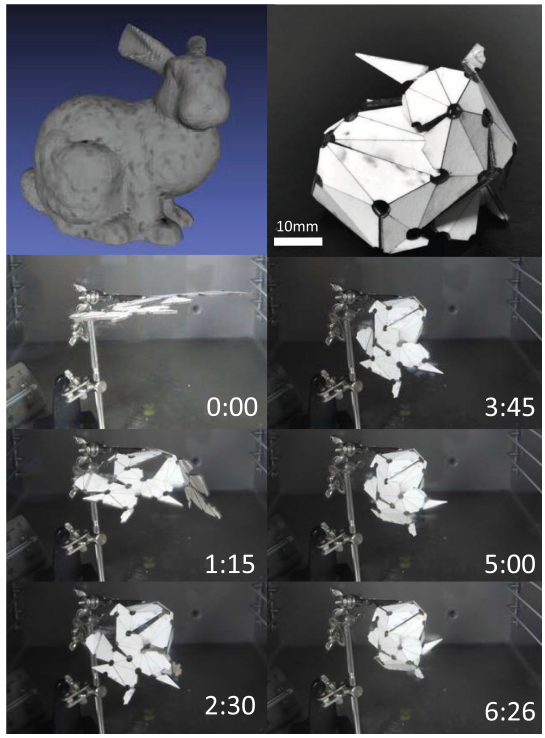


Fig. 1. Self-folding Stanford Bunny. (Top-left) Input 3-D graphic model. (Top-right) Three-dimensional self-folded structure. (Bottom) Frames from the experiment of self-folding by uniform heating. The time elapsed since exposure to uniform heating is indicated in the lower-right corner of each frame (in minutes and seconds).

Self-folding origami structures are robotic sheets composed of tiles and joint actuators [13], [14]. They are developed to simplify the folding process of folding-based designed devices. Each joint actuator holds the neighbor tiles [15]. When the joint receives a signal, it folds the neighbor tiles into a preprogrammed angle. These local foldings yield a global transformation of the sheets [16]. Self-folding origami structures

Manuscript received February 3, 2017; revised September 1, 2017; accepted November 1, 2017. Date of current version December 4, 2018. This paper was recommended for publication by Associate Editor K.-J. Cho and Editor C. Torras upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Grant EFRI-1240383 and Grant CCF-1138967. *(Corresponding author: Byoungkwon An.)*

B. An is with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA, and also with Autodesk Research, San Francisco, CA 94111 USA (e-mail: dran@csail.mit.edu).

S. Miyashita is with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA and also with the Department of Electric Engineering at the University of York, York, YO10 5DD, U.K. (e-mail: shuhei.miyashita@york.ac.uk).

M. L. Demaine, E. D. Demaine, and D. Rus are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: mdemaine@mit.edu; edemaine@mit.edu; rus@csail.mit.edu).

A. Ong and M. T. Tolley are with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA 92093 USA (e-mail: aco002@ucsd.edu; tolley@ucsd.edu).

R. J. Wood is with the School of Engineering and Applied Sciences and Wyss Institute for Biologically Inspired Engineering, Harvard University, Cambridge, MA 02138 USA (e-mail: rjwood@eecs.harvard.edu).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2018.2862882

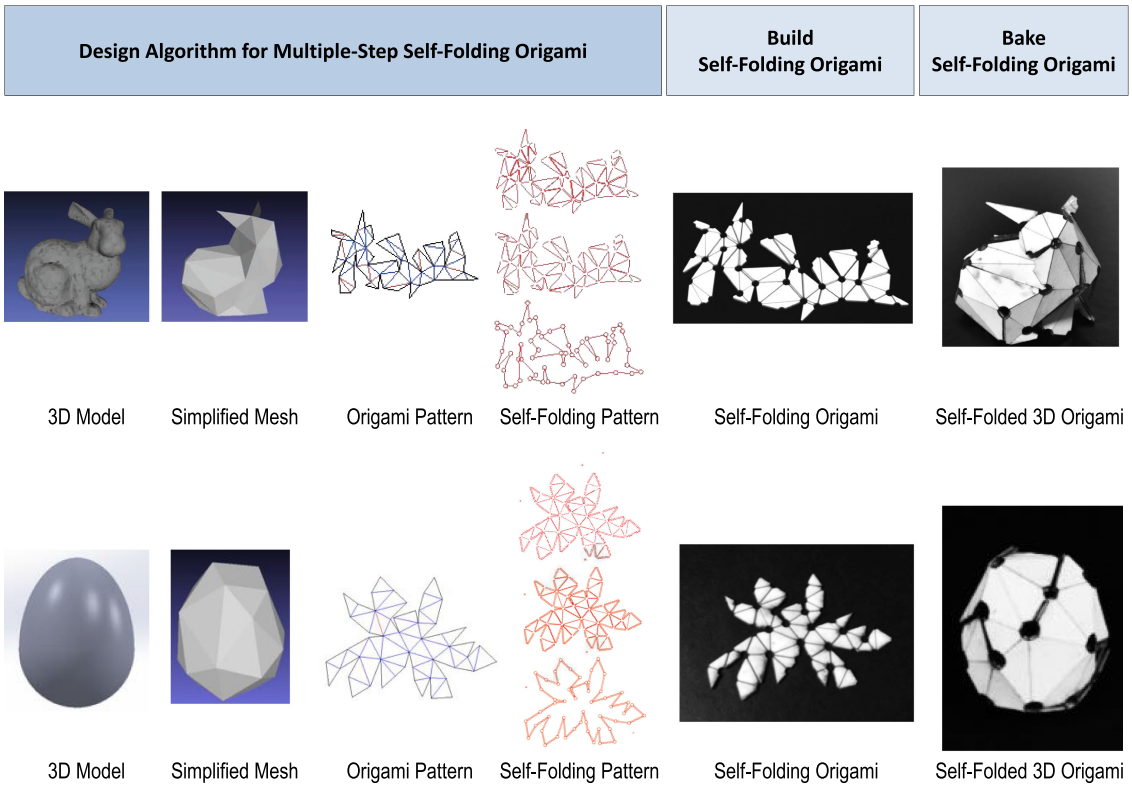


Fig. 2. Visual overview of the self-folding origami development process. Two examples are self-folding bunny and egg.

by uniform heat receive heat as a signal. When the sheet is uniformly exposed to heat, the actuators fold the sheet into the target robotic devices, such as printable robots [17]–[19], sensors [20], and microscale grippers that hold a single cell [21]. Because 2-D fabrication processes are used for making 3-D self-folding robots, the process of fabricating complex structures becomes relatively simple. A self-folding sheet transforms itself into arbitrary 3-D surfacial shapes on-demand. This process enables rapid prototyping with a relatively lower fabrication cost.

Folding fabricated robotic sheets into 3-D devices is relatively easy and simple because the general controllers and planners for the sheets have been studied. However, the design and building process of origami robotic sheets is difficult. This study aims to develop an automated design and fabrication process for self-folding origami robots. We explore an end-to-end approach, including an algorithm and a system that automates the design and fabrication. Given 3-D input models, the algorithm outputs the layouts of the self-folding origami. By printing the algorithmically designed layouts, the user builds robotic sheets. Upon being baked in an oven, these sheets transform into physical devices (see Fig. 2). We also develop a new method and algorithm to control the multiple-step folding by uniform heat. The edges of the sheets have predefined folding temperatures. This allows us to create 3-D devices that require multiple folding steps, advancing the prior work that supported only single-step self-folding [22].

Our contributions include the following:

1) a new method for achieving multiple-step self-folding under uniform heat;
2) a design algorithm that takes a 3-D model as an input and computes layouts of single- or multiple-step self-folding sheets in $O(n^2)$ time, where n is the number of vertices in the 3-D model;
3) an implemented design automation software system including the design algorithm;
4) demonstration of automatically designed self-folding sheets. The self-folded models are comprised up to 55 faces, and the sheet is self-folded in up to two steps (See supplemental videos S1 and S2).

The remainder of this paper is organized as follows. Section III describes and analyzes its model for self-folding origami. Section IV describes the design algorithm. Section V discusses the system implementation. Section VI explores the experiments. Section VII discusses the lessons learned and options for future research.

II. Related Work

A. Programmable Matter by Folding

Our prior work introduced universal self-folding devices called programmable matter by folding [13], [15]. We used a box-pleated crease pattern, which is a universal crease pattern [23], to transform a sheet of special material into any shape composed of $O(n)$ cubes, where n is the length of the side. Its reprogrammability (reusability), folding planning, programming methods, and design and programming automation have been studied theoretically and experimentally [13], [15], [16].

While general folding theory and algorithms for creating folding patterns have been studied for decades, design theory and algorithms for the self-folding origami using a uniform energy source is a recent direction of research interest. Various computational origami designs are introduced in [23]–[29], and the theoretical and experimental complexity of folding patterns are discussed in [13] and [30]. This paper introduces a design algorithm and its verification as well as a compilation-like approach to automate fabrication of self-folding origami.

B. Self-Folding Materials

The self-folding technique has been developed in a broad spectrum at the micrometer scale [31], [32], the millimeter scale [33], and the centimeter scale [34]. There are various self-folding materials that work with heat [19], [20], [35], [36], [20], electronics [17], light [37], cells [38], surface tension [39], and microwaves [40]. Recently, a 3-D printing technology has been proposed as an on-demand synthesis method for self-folding shape memory polymers (SMPs) [41], [42]. As a result, the complexity and scale of the fabricated structures has increased, and the development of the computational methods have become more important. In this paper, we explain the theoretical, system, and experimental aspects of our computational methods. We develop an algorithm to automate the design of sheets that will self-fold as a specified geometric shape. Furthermore, we develop a new method for multiple-step folding (sequential folding). We implement the algorithms as a software pipeline. We performed experiments with two selected self-folding materials reacting to uniform heat from our prior work [35], [36].

C. MultiStep Self-Folding

Most origami shapes are made using multiple folding steps. This process is called sequential folding. For each fold step, some hinges rotate from their original fold angles to other fold angles. Multiple-step folding allows to share some space for the hinge rotations because, after every fold step, the folding trajectory is cleared for the other hinges' trajectories. To control the folding trajectories of self-folding origami sheets, we introduced a fold planning algorithm [16] that determines an optimized folding sequence of a paper piece to achieve a given (or multiple given) origami structures, one or many desired origami shapes. The multiple-step folding plan, which the algorithm built, was implemented with a self-folding sheet [13]. The optimized plan was compiled to a flexible electronic circuit while the fold sequence was manually controlled with a switch. By transferring energy to selected folding hinge, the hinge was triggered by the electrically produced local heat [17], [18]. This paper introduces multiple-step self-folding origami sheets that fold into users' desired shapes with multiple fold steps with no manual intervention. The self-folding sheets work with uniform heat, no on-board controllers, and no local heat control.

III. Models and Definitions

A uniform heat self-folding sheet is defined as a crease pattern composed of cuts (outlines) and folding edges (hinges), as shown in Fig 3. Each edge contains a fold angle and folding group. All the edges of the sheet are controlled using global signals such as uniform heat. The folding group is identified by a predefined temperature, and when a folding group signal is transmitted to a sheet, the edges in the folding group simultaneously fold themselves. Then, when the signal for the second folding group is transmitted to the sheet, the edges of the second group fold. For example, when the uniform heat temperature surrounding a self-folding sheet is 60°, all the edges of the 60° group are self-folded, and when the uniform heat temperature reaches 120°, all the edges of the 120° group are self-folded.

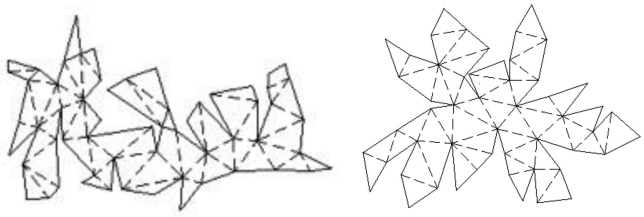


Fig. 3. Visualized self-folding crease pattern representing a bunny shape (left) and an egg shape (right). The solid lines are cuts and the dashed lines are edges (hinges). Each edge contains a fold angle.

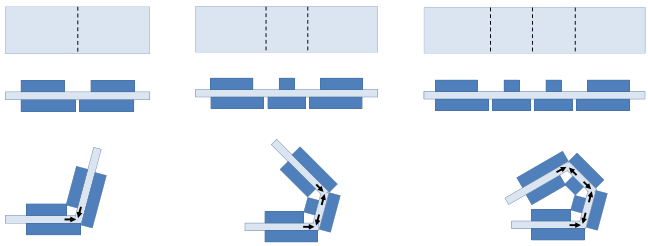


Fig. 4. Three self-folding origami with one, two, and three self-folding (hinge) actuators. The arrows show the shrinking directions. (Top row) Origami patterns. (Middle row) Initiation fold states. (Bottom row) Final fold states.

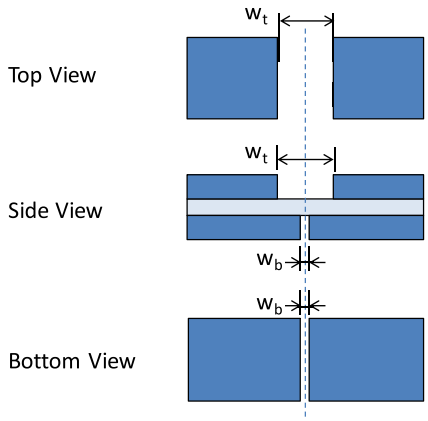


Fig. 5. Structure of self-folding actuator model.

A. Fold Angle

In this paper, a *folding actuator* is composed of three layers (see Figs. 4 and 6). The top and bottom layers of the actuator are heat resistant materials, while the middle layer is a shrinking material. Since all layers are firmly attached to each other, when

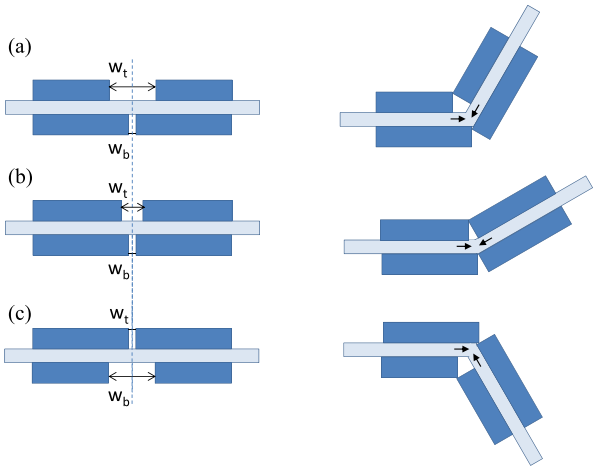


Fig. 6. Self-folding actuator models with three fold angles. (Left) before activation and (Right) after activation. The arrows show the shrinking directions.

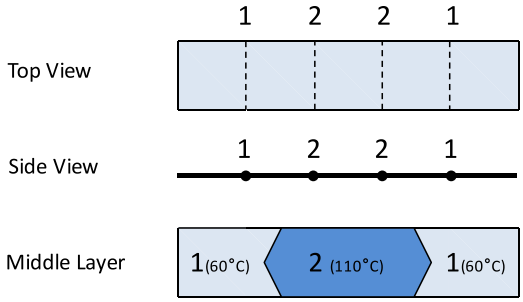


Fig. 7. Self-folding sheet with two-step folding. The middle layer is composed of two different materials: Material 1 reacts at 60 °C and material 2 reacts at 110 °C.

the actuator is exposed to heat, a section of the uncovered middle layer shrinks, allowing the hinge to fold. The size of the folding angle is controlled by the size of the gap (see w_t, w_b in Figs. 5 and 6)

The middle layer is made of a shape memory polymer (SMP), which has the property of shrinking in the presence of heat. The top and bottom layers of the composite are the structural elements of the object and can be made out of any structural material. We used polyvinyl chloride (PVC), prestrained polystyrene (PP, the material used in the children's toy "Shrinky Dinks"), and polyolefin (commonly used for shrink wrap) for the middle layer. We used polyester sheets and paper for the top and bottom layers.

The fold angle of each edge is encoded in the geometric structures of the hinges. Fig. 4 shows simplified models of self-folding sheets. The gaps w_t, w_b of the top and bottom layers determine the fold angles and directions (see Fig. 6). For example, if the gap [see Fig. 6(a)] is wider than the gap at another location [see Fig. 6(b)], the former [see Fig. 6(a)] folds to a greater extent. If the gap of the bottom layer is wider than the gap of the top layer, the actuating edge bends in the other direction [see Fig. 6(c)].

B. Time Step

We achieve sequential folding by using a multimaterial shrinking layer that is segmented into several regions, each capable of shrinking in a different temperature range. This layer is placed on the middle of the multiple-step self-folding sheet, and transforms uniform heat to fold angles of the hinges. In other words, each material shrinks sequentially after a material finishes the shrinking. While the temperature increases, different regions of the middle layer shrink at different times. Fig. 7 shows an example. The left and right edges of the self-folding sheet are placed on material 1, reacting to 60 °C. The two middle edges are on material 2, reacting to 110 °C. When this sheet is baked in the oven, the two outside edges fold first, and then the inside edges fold. We demonstrate this multimaterial middle layer by manual building with jigsaw-puzzle-like placement. A multimaterial printer, like MultiFab [43] or Objet Connex 500, can be used to automate this fabrication.

IV. Design Algorithm

The *multistep self-folding origami design algorithm* converts a shape represented as a 3-D mesh[1] or a 3-D origami model[2] into a *self-folding origami design*, which is the structural layout of the self-folding origami. Just as the crease pattern of an origami model contains the information required to produce a folded origami object, a self-folding origami design contains information to fabricate a multistep self-folding origami. The design is composed of the layers' layouts of the self-folding origami. The self-folding origami can be printed or fabricated according to the design.

The algorithm compiles an input model to the self-folding origami design with the following phases (see Fig. 8):

1) unfolding a given 3-D mesh;
2) computing fold angles;
3) constructing a self-folding crease pattern;
4) constructing a self-folding origami design;
5) drawing a self-folding origami layout;
6) compiling time-step information to the layout of the middle layer.

Phases 1)–5) of the algorithm first compiles structural information (see Section IV-A). If the multistep folding is necessary, the algorithm runs Phase 6) to compile the time information to the middle layer design of the layout (see Section VI-B).

If the input is an origami model associated to its crease pattern and fold angles, the algorithm starts in Phase 3) after skipping Phases 1) and 2).

Theorem 1: Any mesh with n vertices ($O(n)$ faces) can be folded from a multiple-step self-folding pattern built by a design algorithm in $O(n^2)$ time.

Theorem 1 provides the design algorithm's geometric correctness of output self-folding origami design. Section IV-C shows a proof of the theorem.

The notations of the paper are listed in Table I.

[1]A polygon mesh is a collection of faces that defines a polyhedral object.
[2]An origami model is a folded state of a paper structure, that is represented with a crease pattern and folded angles [16].

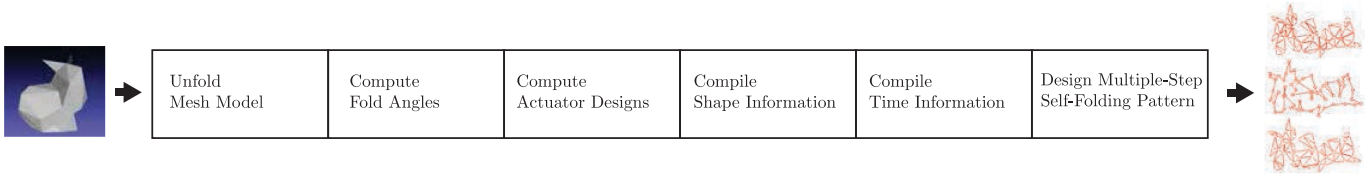


Fig. 8. Six phases of self-folding origami design algorithm.

TABLE I
NOTATIONS

Notation	Name
(Mesh)	
$M = (V, F)$	Mesh
V	Vertex set of M
F	Face set of M
U	Angle set of M
v	Vertex of M
e	Edge
f,f_i	Face in F
n_i	Normal vector of f_i
u	Fold angle
n	The number of vertices of M
$O(n)$	The number of faces of M
(Unfolding)	
$N = (V', E', F', T)$	Unfolding (Net)
V'	Vertex set of N
E'	Edge set of N
F'	Face set of N
T	State set of N
t	State; $t \in \{\langle cut\rangle, \langle hinge\rangle\}$
$N' = (V', E', F', T, U')$	Self-folding crease pattern
$M'(N')$	Folded state Mesh of N'
U'	Angle set of N'
$e(e')$	Original edge e of e'
$f(f')$	Original face f of f'
(Folding Actuator)	
$g(u)$; $g : A \to D$	Actuator design function of u
A	Fold angle set
D	Actuator design set
u	Fold angle$(-180 \leq u \leq 180)$;$u \in D$
$d = (w_t, w_c, w_b)$	Actuator design; $d \in D$
w_t	Gap on top layer of actuator
w_c	Gap on middle layer of actuator
w_b	Gap on the top layer of actuator
$w_t(d)$	Gap on top layer of d
$w_c(d)$	Gap on middle layer of d
$w_b(d)$	Gap on bottom layer of d
ε	None; No gap
S	Fold actuator sample set
$s = (u, d)$	Fold actuator sample
(Layout Design)	
H	Self-folding origami model
$L = (L_t, L_c, L_B)$	Self-folding origami layout
$L_t = (V_t, E_t)$	The top layer of L
$L_c = (V_c, E_c)$	The middle layer of L
$L_b = (V_b, E_b)$	The bottom layer of L

A. Compile Structural Information

1) Unfolding a 3-D Mesh: The objective of this phase is to compute the unfolding of a given 3-D shape. Several algorithms exist to unfold 3-D meshes or 3-D origami designs [44]–[46]. Given a mesh, a set of *nets*[3] is constructed on a plane without any collisions [47]. In this paper, we transform the 3-D mesh in a graph and unfold it using Prim's algorithm (a minimum spanning tree algorithm) [48]. As the algorithm unfolds the 3-D mesh, it maintains the relationship between the vertices of the unfolded 2-D structure and the 3-D mesh.

We define a mesh M is a pair (V, F), where V is a finite set of the vertices, and F is a finite set of the faces of the mesh. A unfolding (net) N is four-tuple (V', E', F', T), where V' is a finite set of the vertices, E' is a finite set of the edges $e' = \{a, b\}$, a and b are in V', F' is a finite set of the faces of the net, T is a finite set of (e', t), and t is a state of e' in $\{\langle cut\rangle, \langle hinge\rangle\}$. $e(e') \in E(M)$ is an original edge of $e' \in E'$. $f(f') \in F(M)$ is an original face of $f' \in F'$. Since all the vertices of the nets are originally from a mesh, during the unfolding process, tracking functions for $e(e')$ and $f(f')$ can be constructed.

2) Computing Fold Angles: The goal of this phase is to compute the fold angles associated with all the edges of a given mesh (see Fig. 9). In origami theory [49], an edge (hinge) is a line segment between two faces. A *fold angle* of an edge is the supplement of the dihedral angle between two faces (see Fig. 10). The sign of the fold angle is determined by the hinge: either a valley fold (+) or a mountain fold (-).

Lemma 1: Given a mesh, a finite set U of all fold angles of the mesh is computed in $O(n^2 \times m)$ time, where n vertices and m edges are in the mesh.

Proof: For each edge, if the edge is not cut, there are two neighboring faces sharing the edge (Algorithm 1 Step 1). Using the dot product and the cross product of their normal vectors, the algorithm calculates the fold angle [see Steps (b), (c)]. Since there are at most n^2 edges, the algorithm computes and stores all angles in $O(n^2 \times m)$ time. ■

Corollary 1: The angles of the mesh can be computed in $O(n^2)$ by update Step 1 (See footnote 4)

3) Constructing the Self-Folding Crease Pattern: This phase takes two inputs, a set of nets and fold angles and computes a *self-folding crease pattern* (the abstracted self-folding information), as shown in Fig. 9. In this section, we show that Algorithm 2 constructs a correct self-folding crease pattern (see Lemma 4). Lemma 2 shows the construction of a self-folding crease pattern, and Lemma 3 shows the correctness of the constructed crease patterns.

Lemma 2: Given a net N and a finite fold angle set U, Algorithm 2 constructs a self-folding crease pattern N' in $O(n^2)$ time.

Proof: Given N and U for each element $(e, u) \in U$, Algorithm 2 transforms the element into $(e'(e), u)$ and inserts it to U'. The algorithm builds a self-folding crease pattern

[3]A net of a mesh is an arrangement of edge-jointed faces in a plane.

[4]The time complexity improves from $O(n^2 \times m)$ to $O(n^2)$ when Step 1) of Algorithm 1 is replaced by following statement: For each face, $f_1 \in E(M)$ and for each edge $e = \{a, b\}$ of f_1, where $e' \neq \langle cut\rangle$.

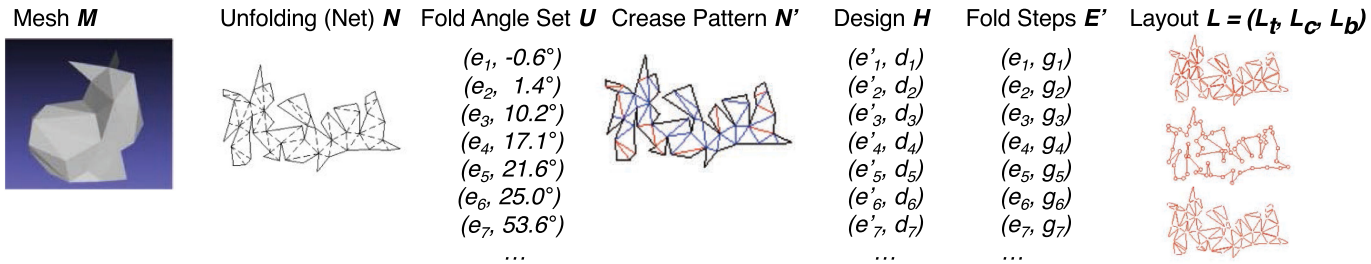


Fig. 9. Data structures of self folding origami design algorithm.

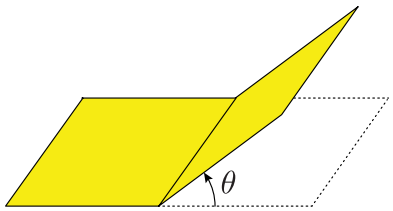


Fig. 10. Fold angle at a crease is the supplement of the dihedral angle.

Algorithm 1: Computing Fold Angles.

input: $M = (V, F)$, where all the normal vectors of the faces point outside and the vertices of each face $(v_1, v_2, \ldots, v_k)$ are positioned counterclockwise from the top view of each face.
output: U

1) For each edge $e = \{a, b\} \in E(M)$, where $e \neq \langle cut \rangle$.
   a) Find two faces f_1, f_2 where f_1 contains directional edge (a, b), and f_2 contains directional edge (b, a).
   b) Get $u = acos(\frac{n_1 \times n_2}{|n_1||n_2|})$, where n_1 and n_2 are the normal vectors of f_1, f_2, respectively.
   c) If $u \neq 0$, and directions of (a, b) and $n_1 \times n_2$ are different, assign '-' to u; otherwise, assign '+' to u.
   d) Insert (e, u) into U.

Algorithm 2: Constructing Self-Folding Crease Pattern.

input: $N = (V', E', F', T)$, U
output: $N' = (V', E', F', T, U')$

1) For each $(e, u) \in U$, insert $(e'(e), u)$ into U'
2) Construct $N' = (V', E', F', T, U')$

$N' = (V', E', F', T, U')$ by adding U' on N. The algorithm runs in $O(n^2)$ time. ■

Lemma 3: Given a mesh M, its net N, its angle set U the self-folding crease pattern N' generated by Algorithm 2, $M'(N')$ is equivalent to M, where $M'(N')$ is the folded state of N'.

Proof: Let $L = \{f'_1, f'_2, \ldots, f'_k\}$, where $L \subseteq F'$, $e(e') = \exists e(e'')$, e' is an edge of f'_i, e'' is an edge of f'_j, $j < i$ and $L = F'$. Let L_t be $\{f'_1, f'_2, \ldots, f'_t\} \subseteq L$. Let $F(M_t)$ be $\{f_i = f(f'_i) \mid f'_i \in L_t\}$. Let $F(M'_t)$ be $\{f''_1, f''_2, \ldots, f''_t\}$, where each f''_i is a face of the folded state of $f'_i \in L$.

For each $t \geq 1$, $P(t)$ is $M'_t = M_t$, where $L_t = F(N_t)$, and N_t is the crease pattern of M_t.

Basis: $P(1)$: $M'_1 = M_1$ because $f_1 = f''_1$.

Induction step: For each $k \geq 1$, we assume that $P(k)$ is true, and we show that it is true for $t = k + 1$.

Suppose the inductive hypothesis is that M'_k is equal to M_k, and f_{k+1} and f''_{k+1} are the same shape. By the definition of L_{k+1}, f'_{k+1} must be connected to $f'_s \in L_k$, and $f(f'_{k+1})$ is connected to $f(f'_s)$, where $s < k + 1$.

Let u' be the fold angle of e' between f''_s and f''_{k+1}. Then $u = u'$, where u is the fold angle of $e(e')$. Thus, $f_{k+1} = f''_{k+1}$ and $F(M'_{k+1}) = F(M_{k+1})$. Therefore $M'_{k+1} = M_{k+1}$, and $P(t)$ is true. ■

Lemma 4: Given M, N, and $U(M)$, Algorithm 2 correctly generates a self-folding crease pattern in $O(n^2)$ time.

Proof: Lemma 2 shows that Algorithm 2 builds a self-folding crease pattern in $O(n^2)$ time. Lemma 3 shows that this self-folding crease pattern is correct. Therefore, Lemma 4 is true. ■

4) Constructing a Self-Folding Origami Design: Given a self-folding origami crease pattern and actuator design function, this phase generates a self-folding origami design (see Fig. 9). A self-folding origami design is an abstracted model of the actuators and the outlines.

A self-folding origami design is a finite set of pair (e', d), where e' is an edge, and d is an actuator design. An *actuator design* d is (w_t, w_c, w_b), where w_t, w_c, and w_b are in $\mathbb{R} \cup \{\epsilon\}$ and are the gaps on the top, middle, and bottom layers, respectively (see Fig. 5). If a variable is in $\mathbb{R}$, the variable is a gap. If a variable is ϵ, then there is no gap. The model in Fig. 5 is (w_t, ϵ, w_b). The gaps of the top and bottom layers are w_t and w_b. Because w_c is ϵ, the middle layer has no gap.

An actuator design can express an outline. For example, if an actuator design is $(0, 0, 0)$, all three layers of this actuator have cuts, and these cuts become an outline.

$g : A \rightarrow D$ denote an *actuator design function*, where A is a set of angles between $-180°$ and $+180°$ and D is a set of actuator designs. The function is dependent on the self-folding material. Each type of self-folding material has a different function. The implementation of g for the experiments is discussed in Section V.

Proof: Algorithm 3 constructs self-folding origami design H. U' contains the fold angles of the edges, while T contains

Algorithm 3: Constructing Self-Folding Origami Design.

input: $N' = (V', E', F', T, U')$, $g : A \rightarrow D$
output: H

1) For each $(e', u) \in U'$:
   a) $d \leftarrow g(u)$
   b) If $t = \langle hinge \rangle$, where $(e', t) \in T$:
      i) Insert (e', d) into H
   c) If $t = \langle cut \rangle$:
      i) $d \leftarrow (w_t(d), 0, w_b(d))$
      ii) Insert (e', d) into H
      iii) $T \leftarrow T - \{(e', \langle cut \rangle)\}$
2) For each $(e', \langle cut \rangle) \in T$:
   a) $d \leftarrow (0, 0, 0)$
   b) Insert (e', d) into H

the types of the edges. Given angle u, $g(u)$ outputs actuator design d [see Step 1-(a)]. According to edge type t and $g(u)$, Algorithm 3 computes each design of the actuator.

For each edge, if the edge is a hinge, the algorithm inserts (e', d) into H. The algorithm removes the edge type from T after inserting the actuator design of the edge [see Step 1-(c)-(iii)]. After Step 1, all edges in T are the cuts of both input mesh and unfolding. Step 2 compiles these edges into actuator design $(0, 0, 0)$. All edges of N' are compiled to H. The algorithm runs in $O(n^2)$ time. ■

5) Constructing a Self-Folding Origami Layout: A self-folding origami layout contains the graphical information of each layer. Given a self-folding origami design, this phase generates three layers of the layout (see Fig. 9). For each element of a self-folding origami design, an actuator layout of a layer is drawn (see Fig. 11).

Lemma 5: A self-folding origami design has a valid self-folding origami layout, computable in $O(n^2)$ time.

Proof: The output of Algorithm 4 is the self-folding origami layout L. L composes three nets L_t, L_c, and L_b. They are the graphical information of the top, middle, and bottom layers, respectively. The algorithm builds the nets.

Each element (e', d) in D contains the gap of each layer and the shape of the bridge. Given an edge, the gap of an actuator of a layer, and a bridge shape, Algorithm 5 draw the layout of the actuator of the target layer. d contains correct actuator and outline information, and w_t, w_c, and w_b of d are correct values. Steps (a)–(c) construct the actuator layout for e'. Steps (d)–(i) add this layout of each layer. The algorithm runs $O(n^2)$ while Steps (a)–(c) are $O(1)$. ■

Lemma 6: Each edge of a self-folding origami design has a valid folding actuator.

Proof: All actuators and cuts of a self-folding crease pattern are described with fold actuators. (1, ϵ, 0) is an example of a valley fold actuator. (0, ϵ, 1) is an example of a mountain fold actuator. (0, 0, 0) is an example of a cut. Each actuator is composed of three layers. Steps (a)–(c) of Algorithm 4 draw an actuator or a cut using Algorithm 5, which draws each layer of the actuator. For example, if an actuator is (1, ϵ, 0), Step (a) of

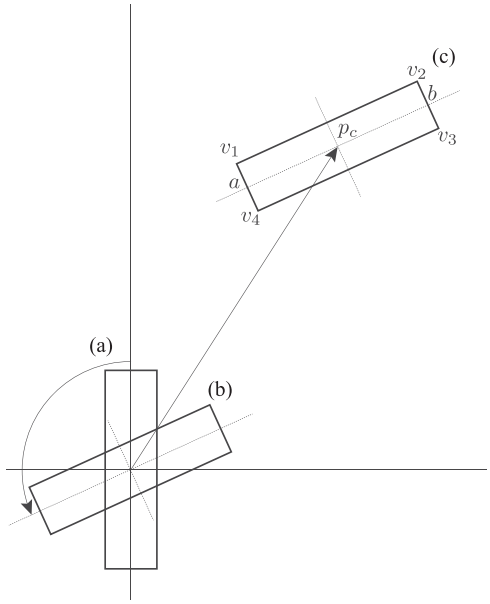


Fig. 11. For input edge $e_i = \{a, b\}$, Steps (a)–(g) of Algorithm 5 draws a rectangle as an actuator layout (a). Steps (h) and (i) rotates the layout (b). Steps (j)–(k) shifts the layout (c).

Algorithm 4: Drawing Self-Folding Origami Layout.

input: H
output: $L = (L_t, L_c, L_b)$

1) For each $(e', d = (w_t, w_c, w_b)) \in H$
   a) Run Algorithm 5 on e' and w_t as w_0, and Algorithm 5 returns $G_t = (V_t'', E_t'')$
   b) Run Algorithm 5 on e' and w_c as w_0, and Algorithm 5 returns $G_c = (V_c'', E_c'')$
   c) Run Algorithm 5 on e' and w_b as w_0, and Algorithm 5 returns $G_b = (V_b'', E_b'')$
   d) $V_t \leftarrow V_t \cup V_t''$ where $L_t = (V_t, E_t)$
   e) $E_t \leftarrow E_t \cup E_t''$
   f) $V_c \leftarrow V_c \cup V_c''$ where $L_c = (V_c, E_c)$
   g) $E_c \leftarrow E_c \cup E_c''$
   h) $V_b \leftarrow V_b \cup V_b''$ where $L_b = (V_b, E_b)$
   i) $E_b \leftarrow E_b \cup E_b''$
2) Construct $L = (L_t, L_c, L_b)$

Algorithm 4 runs Algorithm 5 on 1 as w_0. Algorithm 5 draws the top layer of the actuator with a gap. In Step (b), Algorithm 5 skips the drawing because w_0 is ϵ. In Step (c), Algorithm 5 draws a line $\{a, b\}$, because w_0 is 1. These three layers become an actuator like Fig. 6. Algorithm 5 draws a layer of an actuator, as shown in Fig. 11. Algorithm 5 is $O(1)$. Therefore, Steps (a)–(c) run in $O(1)$.

B. Compile Time Step Information

1) Construction the Middle Layer: In our previous paper [16], we introduced algorithms that, given the final folded state of an origami, determine a folding sequence. The folded state has

Algorithm 5: Construct Actuator Layout.

input: $e' = \{a, b\}$, w_0
output: $G = (V'', E'')$

1) If $w_0 = \epsilon$, then $V'' \leftarrow \phi$ and $E'' \leftarrow \phi$ and return.
2) If $w_0 = 0$, then insert a, b into V'' and $\{a, b\}$ into E''.
3) If $w_0 \neq 0$:
   a) $l \leftarrow$ (length of $e')/2$
   b) $v_1 \leftarrow (w_0, l)$
   c) $v_2 \leftarrow (w_0, -l)$
   d) $v_3 \leftarrow (-w_0, -l)$
   e) $v_4 \leftarrow (-w_0, l)$
   f) Insert v_1, v_2, v_3, v_4 into V''
   g) Insert $\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_1, v_4\}$ into E''
   h) $\theta \leftarrow arctan2(y_b - y_a, x_b - x_a)$
   i) Rotate all vertices in V'' through θ
   j) $c \leftarrow (a + b)/2$
   k) For each $v \in V''$, $v \leftarrow v + c$

information about the number of hinges and their final angles. The folding sequence has information about when the folding groups of hinges are folded, where a group of hinges fold simultaneously. We found that, in practice, some origami structures had to be constructed with more than one folding step. A collision is a common issue of failure, for this reason, the folding trajectory should be more accurately controlled. Fortunately, there are many origami shapes can be realized with multiple-folding steps. Our prior approach was to use an on-board electronic controller to selectively transfer energy to a folding hinge [17], [18]. The hinge was triggered by the local heat made by the energy. In this section, we introduce self-folding origami that transform themselves into user's desired shapes with multiple-folding steps. The origami work with uniform heat, no on-board controllers, and no local heat control.

To achieve multiple-step sequential folding with uniform heat, we extend the self-folding origami model with a multi-material shrinking layer (see Fig. 7). The top and bottom layers of this model are automatically designed by Algorithm 4. The middle layer is composed of multiple materials that react to different temperatures. Intuitively, the edges made of materials reacting to lower temperatures fold first. Then, the other folding edges reacting to higher temperatures fold after that. Additional details of the model are described in Section III-B. Given a self-folding crease pattern, a folding sequence can be automatically computed—in our prior work [16], we introduced a folding-planning algorithm that computes optimized folding sequences by grouping the simultaneously foldable edges and minimizing folding steps. For k-step sequential folding, k shrinking materials are used for the middle layer.

The middle layer of self-folding origami is algorithmically designed. In this section, we describe an algorithm for generating the design of a middle layer (see Fig. 12, Algorithm 6).

An edge e in this section is a three-tuple (a, b, g), where a and b are vertices, and g is a folding group. The edges with the same folding group are folded at the same time. The edges of the smaller folding groups always fold before the edges of larger folding groups. For example, the edges of group 1 fold before the edges of group 2.

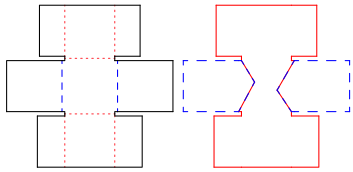


Fig. 12. Example of the construct of a multimaterial middle layer (see Algorithm 6). (Left) Crease Pattern N'. The red dotted lines are the first step folding creases, and the blue dashed lines are the second step folding creases. (Right) Middle Layer of Layout L_c. The red solid line polygon is shrinking material 1. The blue dashed line polygon is shrinking material 2. Material 1 reacts at the first folding step. Material 2 reacts at the second folding step.

Algorithm 6: Constructing Multi-Material Middle Layer.

input: $N' = (V', E', F', T, U'), L_c = (V_c, E_c)$
output: $L_c = (V_c, E_c)$

1) For each e in E', if e is an outline, set $\langle None \rangle$ to group $g(e)$.
2) Split all faces in F' into triangle faces, and set $\langle None \rangle$ to the groups of all newly made edges during the triangulation.
3) For each face $f = (a, b, c)$ in F':
   a) Insert a new vertex i in V', where i is the center of the incircle of the triangle f.
   b) For each edge $e = (v_1, v_2, g)$ of f:
      i) Insert face $((v_1, v_2, i), g(e))$ into F'', where $g(e)$ is the folding group of edge (v_1, v_2).
      ii) Insert $(v_1, i, g(e)), (v_2, i, g(e))$ into B.
      iii) If e is an outline, insert $(v_1, v_2, g(e))$ into B.
4) For each e in B, where $f \in F'$ and $f' \in F''$ are the neighbor faces of e, and $g(f)$ is $\langle None \rangle$ and $g(f')$ is not $\langle None \rangle$:
   a) $g(f) \leftarrow g(f')$.
   b) Change the groups of all edges of f to $g(f')$.
5) Repeat 4) until the group of no edges in B is $\langle None \rangle$.
6) For each e in B, where f and f' in F'' are sharing e, and $g(f)$ is equal to $g(f')$:
   a) Remove e from B.
7) $V_c \leftarrow V_c \cup V'$
8) For each $(v_1, v_2, g) \in B$, insert $\{v_1, v_2\}$ into E_c

Lemma 7: A self-folding crease pattern with sequential folding steps has a valid shrinking layer design, computable in $O(n^2)$ time, where n and $O(n)$ are the numbers of vertices and faces, respectively.

Proof: Algorithm 6 constructs a multimaterial shrinking layer. The algorithm is composed of four parts. Steps 1–3 prepare the geometry, Step 4 tessellates the possible boundaries of the materials, and Steps 5 and 6 assign all areas to a shrinking material. Step 7 removes unnecessary boundaries and merges

the areas. Step 8 outputs B, the design of the multimaterial shrinking layer. Each edge of B is assigned a folding group. All edges of each folding group represent the boundary of the shrinking material for this folding group.

Given a self-folding crease pattern N', the algorithm sets $\langle None \rangle$ to the groups of outline edges (see Step 2) and the groups of new edges generated during the triangulation (see Step 3). In Step 4, it splits each triangle into three small triangles. It adds vertex i, where i is the center of the inscribed circle of the triangle. In Step 4-b, the algorithm constructs a boundary edge set B and small triangle set F''. Step 4 runs in $O(n)$. After building F'', some small triangles (faces) in F'' are not assigned to any groups. The algorithm moves the faces in the $\langle None \rangle$ group to the group of a neighbor face (see Step 5). After this step, all faces are assigned to exactly one folding group. For these steps, we chose the triangle shape as it is the most commonly used polygon for mesh given its consistent convex property. In this regard, any partitioning algorithm, including Voronoi partitioning, shall work.

$O(n)$ is the number of the edges in the $\langle None \rangle$ group after Step 4. Each time Step 6 runs, at least one group of an edge in B is changed from $\langle None \rangle$. Thus, Steps 5 and 6 run in $O(n^2)$.

The algorithm merges the areas with the same material by removing the boundary edges in B (see Step 7). The algorithm exports a valid shrinking layer (see Step 8). Since all the small faces are assigned to a group, Step 7 runs in $O(n)$. The total running time is $O(n^2)$, and the running space is $O(n)$. ■

Fig. 13 shows the input and output of the algorithm. The inputs are the final folding states of the origami structures.

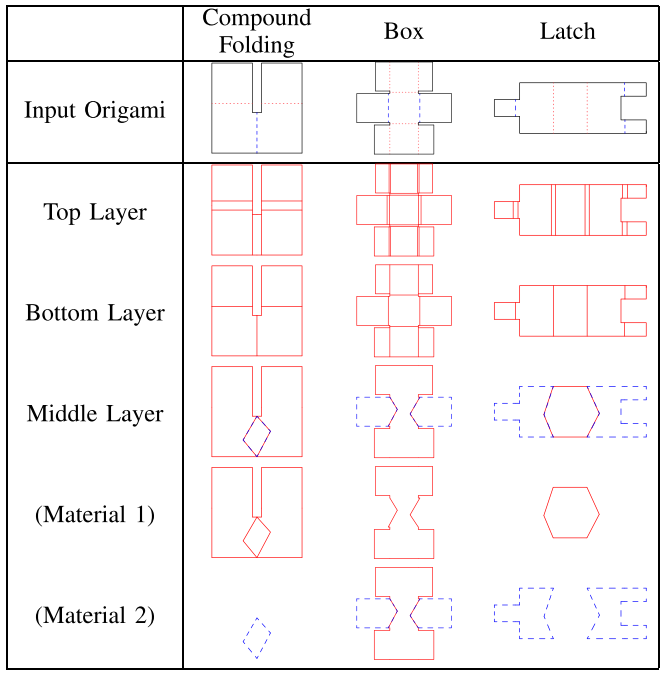


Fig. 13. Design of the multiple-step folding algorithm. (Input Origami) An input origami represents a final fold state of origami. Each colored crease line represents an angle and a step. Each line of an angle of compound folding is 180°. The red dotted lines are the first step folding creases, and the blue dashed lines are the second step folding creases. All lines are valley folds. (Top, Bottom Layers) The red solid lines are cut traces. The top and bottom layers are rigid materials. (Middle Layer) The red solid line polygons are shrinking material 1. The blue dashed line polygons are shrinking material 2. Materials 1 and 2 react sequentially.

C. Proof of Theorem

Now we are ready to prove Theorem 1.

Proof: We derive the layout of the self-folded origami whose folded state is equivalent to the input model (see Lemma 3). We also show how a sequence of origami folding is encoded into the associated self-folded origami (see Lemma 7). Each group of edges is associated with a middle layer material that reacts to heat. Thus each edge can be folded according to the target angle. The total required computation time is $O(n^2)$. ■

V. Implementation

A. Software for Compiling a Printable 2-D Design

We implemented the design algorithm in Java. The input file formats are Wavefront .obj for a 3-D mesh and AutoCAD .dxf for a 3-D origami design [16]. The output files are in the .dxf format.

To support the various manufacturing processes of the self-folding origami, the software supports script files to define the template of the fabrication files (outputs). To demonstrate automatically generated self-folding origami with two manufacturing processes, we built two template scripts: a *folding-alignment* manufacturing process [35] and a *pin-alignment* manufacturing process [36].

B. Actuator Design Function

The folding angle is determined by the combination of the thicknesses of three layers. Our previous work revealed that the torque inducible is proportional to the thickness [36], namely the mass of SMP, albeit the mass also increases in the same proportion. This implies that in order to exploit the maximum lifting torque of a hinge, using less dense structural sheet is a solution. We also identified various issues caused by the physical limitation associated with practical self-folding.

Given a fold angle u, an actuation design function g outputs an actuator design d. An actuator design is composed of three parameters (w_t, w_c, w_b) (see Section III). We implement this function by sampling the profile and construct a fold angle sample set S. When g receives u, if u is in S, g outputs d in S; otherwise, g approximates and outputs a design. This function is formally defined as shown in Definition 1.

Definition 1: An actuator design function is $g : A \rightarrow D$, where

1. A is a set of the angles u ($-180° \leq u \leq 180°$);
2. D is a set of the actuator designs $\{d_1, d_2, d_3, \ldots, d_i, \ldots\}$ (see Section IV-A4);
3. S is a finite set of the fold angle samples $s_i = (u, d)$ for $u(s_i) < u(s_{i+1})$;
4. $s_0 = (0, (0,\epsilon, 0)) \in S$;
5. if $(u, d) \in S$, then, $g(u) = d$;

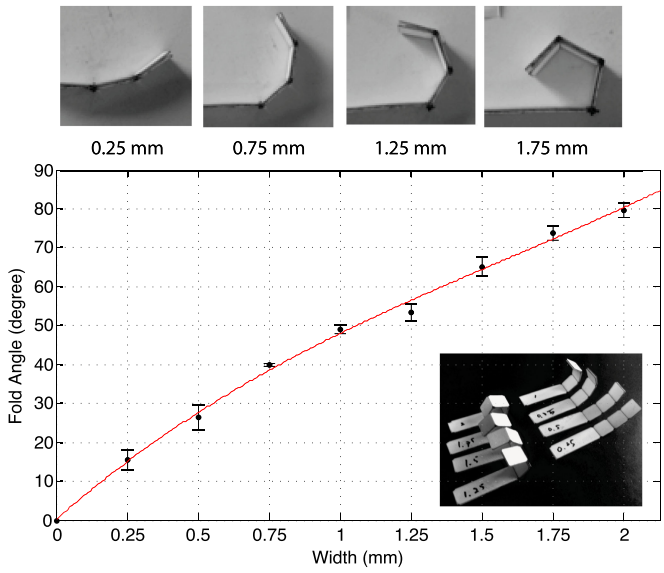


Fig. 14. Graph of an implemented actuator design function for the pin alignment process. The inset images show the test strips used to characterize the fold angle as a function of the size of the gap on the inner structural sheet. Each bar is the standard deviations from the average of the angles of three hinges (see Table II).

6. if $(u, d) \notin S$, then, $g(u) = (w(d_i) + (u - u_i/u_{i+1} - u_i) \times (w(d_{i+1}) - w(d_i)), \epsilon, b(d_i) + (u - u_i/u_{i+1} - u_i) \times (b(d_{i+1}) - b(d_i)))$, where $u_i = u(s_i)$, $u_{i+1} = u(s_{i+1})$, $d_i = d(s_i)$, $d_{i+1} = d(s_{i+1})$, $u_i < u < u_{i+1}$, and s_i, $s_{i+1} \in S$.

$g(u)$ is continuous for $u \in A$. If u is in $u(s)$ for $s = (u, d) \in S$, $g(u)$ outputs d. Otherwise, $g(u)$ constructs an actuator design d according to the actuator ratio $(u - u_1/u_2 - u_1)$ and designs d_1 and d_2, where u_1 and u_2 are the angles of d_1 and d_2, and $u_1 < u < u_2$.

Theoretical model covers geometric properties, such as collisions, edge types, or scalability. The geometry issues are characterized by material functions that can experimentally be built. The other practical issues include thickness, transition temperatures, force, gravity, and self-folding hinges connected with many faces. To handle these issues, we define an actuator design function and develop a planning algorithm. The function works as an interface between the algorithm and experiments. To minimize the gap between theory and experiment, we have implemented the function using experimental data. We plugged this function into the pipeline system (software), as an input. It covers the unpredictable characteristics of self-folding transitions.

To implement the actuator design function, we characterize the fold angle as a function of the actuator geometry. We built eight self-folding strips with gaps on the inner layer in the range of 0.25–2 mm and baked them at 170 °C. Each strip had three actuators with identical gap dimensions. After baking, we measured the fold angle of each self-folded actuator with a different gap, as shown in Fig. 14. This method is modified from our prior work in [36]. This time, we automated the design process of the strips using our self-folding origami design pipeline. We can easily generate another set of strips for a different range of gaps.

TABLE II
FOLDING ANGLES

Gap	Angle1	Angle2	Angle3
0.25 mm	11.58°	14.6°	20.09°
0.50 mm	22.85°	23.34°	33.25°
0.75 mm	39.49°	40.44°	39.79°
1.00 mm	47.6°	51.16°	48.42°
1.25 mm	56.62°	49.36°	54.51°
1.50 mm	69.57°	61.39°	64.39°
1.75 mm	77.44°	72.88°	71.36°
2.00 mm	80.34°	82.53°	76.13°

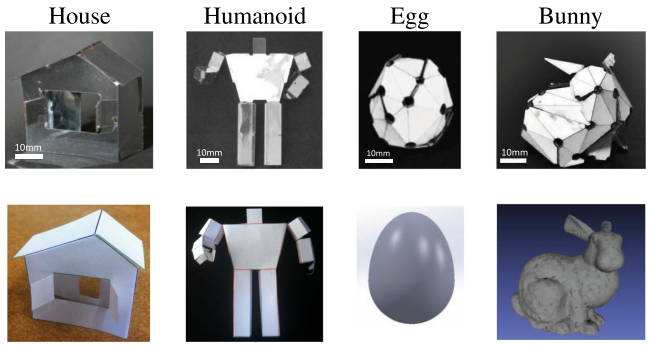


Fig. 15. (Top) Self-folded 3-D shapes: the house, humanoid, egg, and bunny shapes. Each scale bar is 10 mm. (Bottom) Input models. We modeled the house and humanoid designs with paper and coded them into origami patterns. We modeled the egg and bunny shapes using CAD software.

VI. EXPERIMENTS

A. Fold Structure Control

We evaluated the self-folding pipeline by building self-folding origami sheets for four shapes: a house, a humanoid, an egg, and a bunny (see Fig. 15). The bunny is the most complex shape we self-folded by heating. Given the 3-D models of these input shapes, the pipeline outputs a set of .dxf files containing the layout of each self-folding origami. We built and baked each self-folding origami according to two different fabrication processes: folding alignment [35] and pin alignment [36]. The pipeline successfully built the shapes in a relatively short time (see Table V).

We built the humanoid and house origami shapes using paper. The 3-D shape of the house was composed of nine faces, and its 2-D unfolding contained eight actuators. The 3-D shape of the humanoid was composed of 41 faces, and its 2-D sheet contained 44 self-folding actuators (see Table III). Fig. 16(a) and (b) shows the fabrication files of the house shape and the humanoid shape.

The egg shape was modeled in the CAD software (Solidworks, Dassault Systemes SolidWorks Corp.) and exported as a 3-D mesh with 2538 faces. We reduced the number of the faces to 50 (MeshLab, Visual Computing Lab, ISTI, CNR) and then unfolded it with our software. The 2-D sheet of the egg contained 48 actuators (see Table III). We generated the fabrication

TABLE III
COMPLEXITY OF TARGET MODEL

	House	Humanoid
# of Faces	9	41
# of Actuators	8	44
Fold Angle Range	-135.0°– 90.0°	-100.0°– 125.0°
	Egg	Bunny
# of Faces	50	55
# of Actuators	48	54
Fold Angle Range	-0.6°– 55.0°	-103.4°– 67.1°

TABLE IV
FABRICATION AND MATERIAL OF SELF-FOLDING SHEETS

	House & Humanoid	Egg & Bunny
Fabrication Process	Folding	Pin
Folding Temp.	65°C	120°C
Top & Bottom Layers	Mylar	Paper
Middle Layer	PVC (Polyvinyl Chloride)	PP (Prestrained Polystyrene)

TABLE V
COMPUTING AND SELF-FOLDING TIMES

	House	Humanoid
Computing Time	392.17 ms	478.17 ms
Folding Time	4m 57s	4m 58s
	Egg	Bunny
Computing Time	478.2 ms	464.5 ms
Folding Time	2m 37s	6m 26s

CPU	Intel Core i3-2350M (2.30 GHz)
RAM	4 GB
Storage	500 GB 5400 rpm 2.5” HDD (TOSHIBA MK5076GSX)
Graphics	Intel HD Graphics 3000

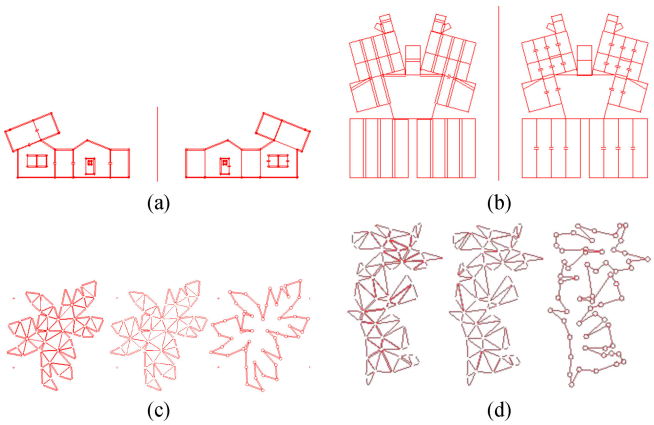


Fig. 16. Fabrication layout for self-folding origami. (a) and (b) are fabrication layouts of the folding alignment process generated for the house and humanoid. The left and right sides of each house and humanoid are the top and bottom layers, respectively. The line in the center guides the folding alignment while the top layer and the bottom layer are sandwiched. (c) and (d) are fabrication layouts of the pin alignment process generated for the egg and bunny. The tiny holes are for the pin alignments. The left, middle, and right sides of each egg and bunny are the top layer, the bottom layer, and the final outline.

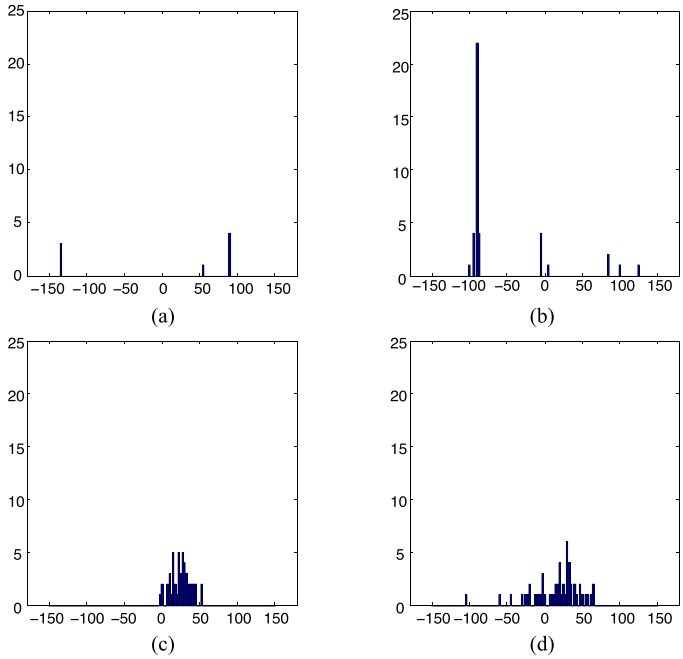


Fig. 17. Histograms of the fold angles. The x-axis is fold angles. The y-axis is the frequency. The width is 2.5°. (a) House. (b) Humanoid. (c) Egg. (d) Bunny.

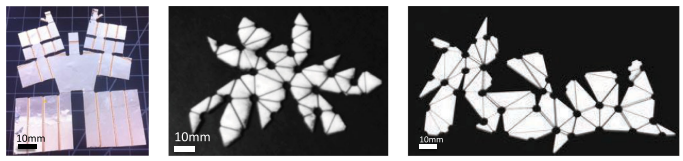


Fig. 18. Self-folding sheets (before baking) for humanoid (left), egg (center), and bunny (right). Each scale bar is 10 mm.

files for the egg shape from this model. Fig. 16(c) shows the fabrication files of the egg shape.

For the bunny shape, we downloaded the 3-D Stanford Bunny (Rev 4, Stanford Computer Graphics Laboratory), which contains 948 faces, and reduced the number of the faces to 55 using MeshLab. We unfolded this mesh and created the fabrication files with our software. Fig. 16(d) shows the fabrication files of the bunny shape.

After we built the fabrication files, we manufactured physical self-folding origami sheets for the house, humanoid, egg, and bunny shapes (Fig. 18). Folding alignment was used for the house and humanoid shapes, whereas pin alignment was used for the egg and bunny shapes. The algorithm of the pipeline is general enough to apply to two different self-folding approaches (see Table IV).

Each shape has various fold angles. The distributions of these angles are shown in the histograms in Fig. 17. The angles of the humanoid have the widest range, although the most frequent angles are 90°. The bunny includes the most diverse angles in both valley and mountain folds.

We heated the house and humanoid at 65 °C without preheating the oven. We put each sheet into the oven at the room

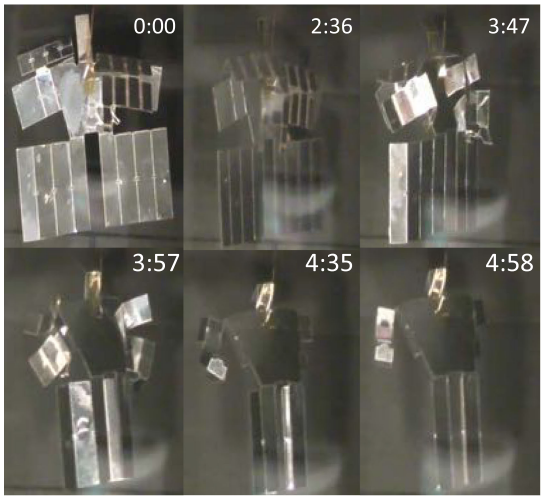


Fig. 19. Frames from the experiment of the self-folding humanoid shape by uniform heating. The sheet was built using the folding alignment process. The time elapsed since exposure to uniform heating is indicated in the upper-right corner of each frame (in minutes and seconds).

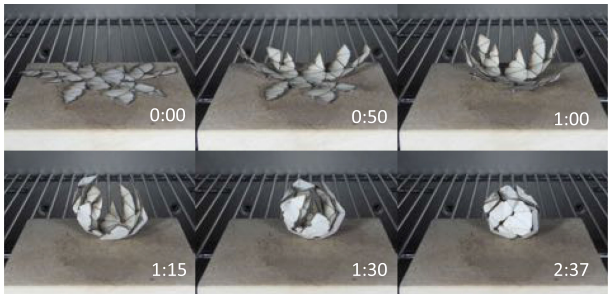


Fig. 20. Frames from the experiment of the self-folding egg shape by uniform heating. The sheet was built with the folding alignment process. The time elapsed since exposure to uniform heating is indicated in the lower-right corner of each frame (in minutes and seconds).

temperature and then increased the heat to 65 °C. The egg and bunny were baked in an oven preheated to 120 °C. While the sheet of the egg shape was placed on the preheated ceramic plate, the sheets of the humanoid, house, and bunny shapes were hung on bars in the oven to reduce the effect of gravity on the self-folding process.[5] Figs. 19, 20, and 1 show frames of the videos taken during the experiments with the self-folding bunny, humanoid, and egg shapes, respectively. To determine the reliability of the pipeline, we baked ten self-folding bunnies and eight eggs and measured their well-formed rates. When all vertices meet in a 3 mm (the circle size of the vertices) radius circle, the result is called a well-folded shape; otherwise, it is called a failed shape.

Our self-folding algorithm designed self-folding origami sheets that accurately reproduced the house and humanoid shapes. The house, humanoid, and bunny shapes were suspended while they were self-folding because the fold-force is not strong enough to lift the whole body. The egg shape is folded on a plate.

[5]For an analysis of the forces provided by such self-folding actuators in the presence of gravity as well as the resulting design constraints.

TABLE VI
FAILURE RATES

	Egg	Bunny	Total
Run	8	10	18
Failure	0	2	2
Failure Rate	0%	20%	11.1%

Fig. 21. Self-folded bunny and egg shapes. The scale bar is 10 mm.

Using the proposed pipeline, the self-folded structures were rapidly designed and built (see Table V). The computing time for each model was less than 0.5 s. The self-folding time was also relatively short. All shapes folded themselves in 7 min; the egg folded itself on a preheated ceramic plate in 3 min. The time to physically construct the 2-D self-folding origami sheets took longer than all of the other steps combined because the construction includes manual labor, such as CO_2 laser machining, alignment, layer lamination, and release cutting.

The failure rate of the egg shape was 0% while the failure rate of the bunny shape was 20.0%. Two out of the ten bunnies failed because of overfolding, creating collisions during the process. Delamination of the SMP layers from the structural layers was observed along the overfolded edges. The total failure rate was 11.1% (see Table VI, Fig. 21).

During the self-folding of some bunny shapes, slight collisions of the faces (which did not interrupt the folding procedure) were observed. This can be addressed by using a self-folding simulator to minimize the collision while the pipeline generates the design. Alternatively, we can use a multiple-step folding algorithm.

B. Time Control

Fig. 13 shows the multiple-step self-folding patterns used for the time-control experiments.

1) Compound Folding: To achieve multiple-step self-folding, two materials, PVC (SMP 1), which reacts at ∼65 °C, and polyolefin (SMP 2), which reacts at 80 °C, are used for actuation to enable a two-step self-folding process. The exper-

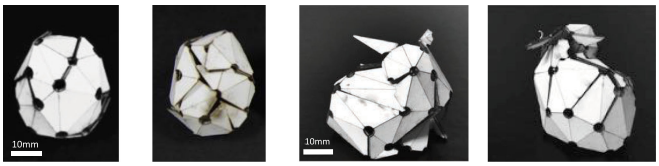


Fig. 22. Front and back sides of the self-folded egg and bunny. Each scale bar is 10 mm.

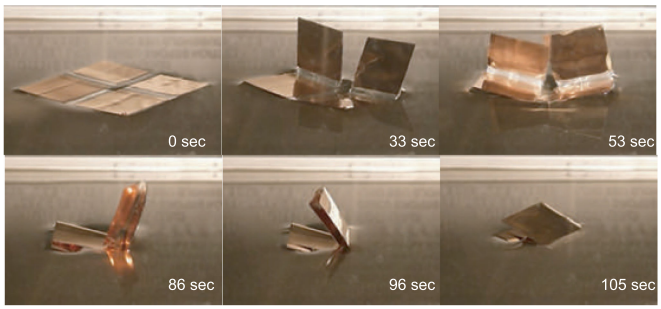


Fig. 23. Frames from the experiment of compound folding. Two actuation materials differentiate the timings of self-folding and enable compound folding.

TABLE VII
FABRICATION, MATERIAL, AND TIME SPECIFICATION OF BOX AND LATCH SHAPE

	Compound Folding	Box	Latch
Fabrication Process	Folding	Pin	Pin
Top & Bottom Layers	Mylar	Paper	Paper
Middle Layer (SMP 1)	PVC	Polyolefin	Polyolefin
Middle Layer (SMP 2)	Polyolefin	Polystyrene	Polystyrene
Folding Time of SMP 1	86 sec	82 sec	90 sec
Folding Time of SMP 2	105 sec	200 sec	118 sec

imental result of compound self-folding is shown in Fig. 23. The experiment was conducted on the water in an oven, and the temperature was raised to 80 °C from the room temperature. Note that the elapsed time shown was measured starting from the time that deformation on creases was observed. First, two creases actuated by PVC started self-folding (33–53 s) then a crease actuated by polyolefin followed (86–96 s). As a result, the structure was folded into a fourth of the original size (105 s).

2) Box and Latch: We designed two self-folding shapes to demonstrate the significance of sequential folding (see Figs. 24 and 25). The first design presents a folded box [see Fig. 25(d)], which requires sequential folding, while the second addresses the issue of latching in order to lock the assembled structure [see Fig. 25(h)]. Both shapes require a two-stage folding sequence for proper assembly. [Unsuccessful single-stage versions of these designs are shown in Fig. 25(b) and (f).]

To achieve sequential folding, we used a multimaterial layer (see Fig. 7) composed of polyolefin (SMP 1) for the first stage of folding and prestrained polystyrene (SMP 2) for the second. Fig. 25(a), (c), (e), and (g) show 2-D laminates, where the transparent hinges show the region composed of polyolefin, and the solid-colored hinges show the region composed of PP. To fabricate these laminates we used pin-alignment (see Fig. 5). We cut

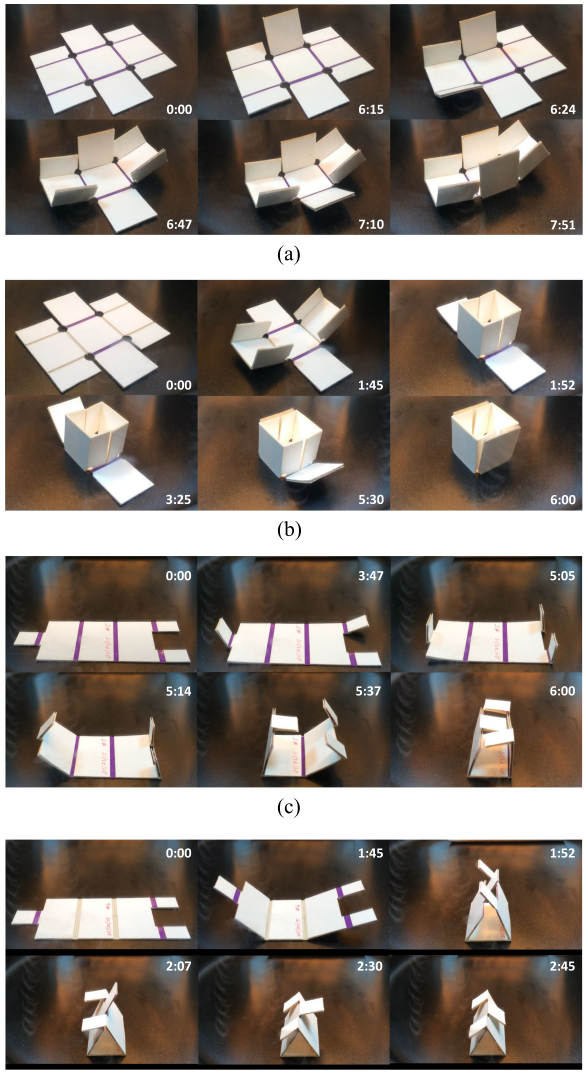


Fig. 24. Frames from the experiment of box and latch. (From the top left) (a) Single-material middle layer for box. (b) Multimaterial middle layer for box. (c) Single-material middle layer for latch. (d) Multimaterial middle layer for latch.

the generated .dxf files from the algorithm presented earlier for all the layers using a laser system (ULS PLS6MW). The layers were laminated using adhesive layers. Finally, the laminate was heated in a convection oven (12 qt. Fagor Halogen) until the final structure was achieved.

We performed eight trials for each shape with the oven starting from the room temperature and set to a target temperature of 175 °C. The sequential folding specified in input origami was achieved. Box SMP 1 reacted before Box SMP 2. Latch SMP 1 reacted before Latch SMP 2. Box SMP 1 reacted before Box SMP 2. Latch SMP 1 reacted before Latch SMP 2. As the oven heated, the region involving polyolefin actuated first at an average time and temperature of 82 s at 99 °C for the box and 90 s at 94 °C for the latch. The polystyrene began folding for the box at 200 s at 140 °C and at 118 s at 111 °C for the latch. Fig. 26 shows the relative temperature of actuation measured using a K-type thermocouple (Fluke 87 V Digital Multimeter). This graph

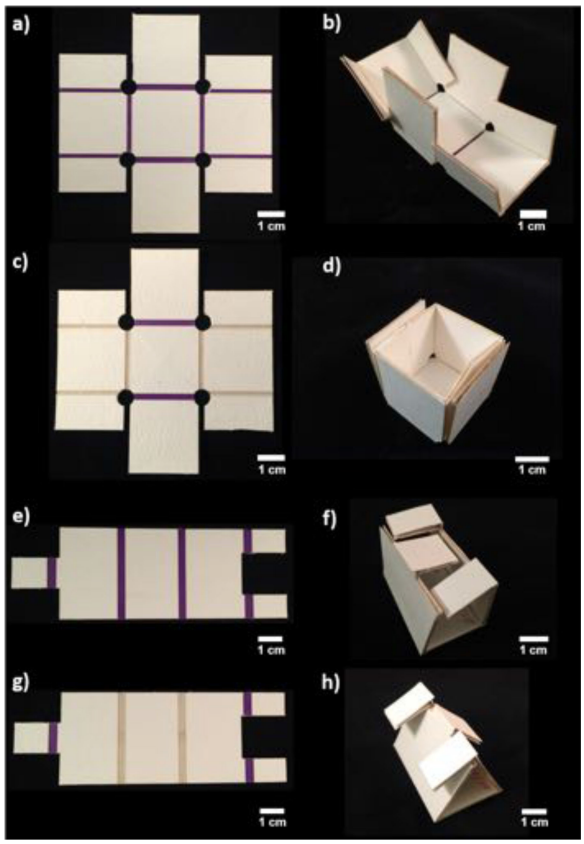


Fig. 25. Unfolded (left column) and folded (right column) structures for the box and latch. (a) single-material middle layer for box. (b) Failed box assembly for a single-material middle layer. (c) Multimaterial middle layer for box. (d) Successful sequential folding of box for a multimaterial middle layer. (e) Single-material middle layer for latch. (f) Failed latch assembly for a single-material middle layer. (g) Multimaterial middle layer for latch. (h) Successful sequential folding of latch for a multimaterial middle layer.

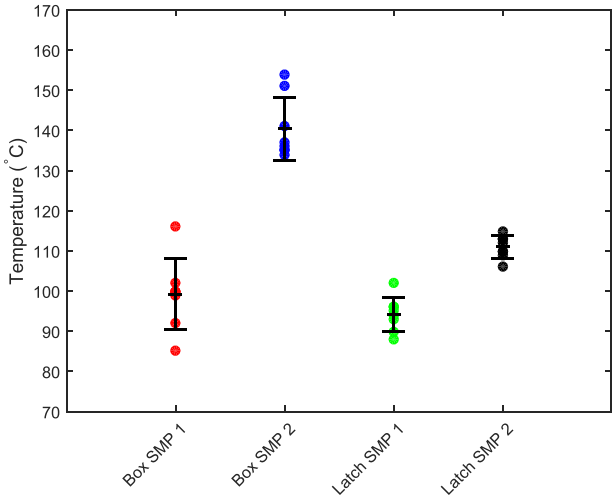


Fig. 26. Relative temperatures of actuations for SMP 1 and 2 for box and latch shapes. SMP 1 and 2 are used for the first and second folding steps of the shape, respectively. Each bar is the standard deviations from the average. Eight points of each material represent the relative temperatures of eight trials of each shape.

shows a distinct difference in the actuation temperature for the polyolefin and polystyrene SMPs for each shape.

VII. Conclusion and Future Work

In this paper, we described an end-to-end approach to designing and building self-folding origami sheets activated by uniform heat. We introduced a design pipeline that automatically generates folding information for an arbitrary 3-D shape and then compiles this information into fabrication files. We modeled single- and multiple-step self-folding origami sheets that fold into arbitrary fold angles. We proposed a design algorithm for such sheets and proved its correctness. We also demonstrated the implementation of this pipeline and characterized the actuator design function to convert the theoretical design into a physical self-folding origami. Finally, we validated this approach experimentally by generating self-folding origami for the fabrication of seven target shapes with up to 55 faces and up to 2 step folds. These were correctly designed and baked into their respective physical shapes under uniform heat.

Several practical challenges remain to be addressed in the physical fabrication of self-folding origami sheets. Delamination of the SMP layers from the structural layers occurred along the edges of our self-folding origami when baking the egg and bunny shapes. This can be mitigated by sealing the edges of the sheet or with improved adhesion.

Another challenge is the evaluation of self-folding origami. Although the back side of the bunny shape in Fig. 22 shows the completion of the shape, it was difficult to evaluate or analyze the completeness of the self-folded model. The development of benchmarks and criteria for evaluating the quality of self-folding origami would support a systematic approach to methodological improvements in this area. In our future work, we aim to extend this approach to create mobile/actuatable self-folded machines.

Acknowledgment

The authors would like to thank D. M. Aukes, L. Meeker, J. Romanishin, and M. Volkov for their insightful discussions and technical support for this research.

References

[1] K. Miura, "Method of packaging and deployment of large membranes in space," *Inst. Space Astronaut. Sci.*, vol. 618, pp. 1–9, 1985. [Online]. Available: https://repository.exst.jaxa.jp/dspace/handle/a-is/7293

[2] S. A. Zirbel *et al.*, "Accommodating thickness in origami-based deployable arrays 1," *J. Mech. Des.*, vol. 135, no. 11, Nov. 2013, Art. no. 111005.

[3] C. D. Onal, R. J. Wood, and D. Rus, "An origami-inspired approach to worm robots," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 2, pp. 430–438, Apr. 2013.

[4] S. T. Brittain, O. J. A. Schueller, H. Wu, S. Whitesides, and G. M. Whitesides, "Microorigami: Fabrication of small, three-dimensional, metallic structures," *J. Phys. Chem. B*, vol. 105, no. 2, pp. 347–350, Jan. 2001.

[5] A. P. Gerratt, I. Penskiy, and S. Bergbreiter, "Integrated silicon-PDMS process for microrobot mechanisms," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 3153–3158.

[6] A. M. Hoover, E. Steltz, and R. S. Fearing, "RoACH: An autonomous 2.4g crawling hexapod robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 26–33.

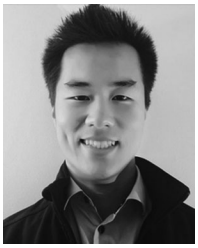
[7] P. S. Sreetharan, J. P. Whitney, M. D. Strauss, and R. J. Wood, "Monolithic fabrication of millimeter-scale machines," *J. Micromech. Microeng.*, vol. 22, no. 5, May 2012, Art. no. 055027.
[8] A. T. Baisch, O. Ozcan, B. Goldberg, D. Ithier, and R. J. Wood, "High speed locomotion for a quadrupedal microrobot," *Int. J. Robot. Res.*, vol. 33, no. 8, pp. 1063–1082, Jul. 2014.
[9] F. Haas and R. J. Wootton, "Two basic mechanisms in insect wing folding," *Proc. Roy. Soc. B, Biol. Sci.*, vol. 263, no. 1377, pp. 1651–1658, Dec. 1996.
[10] T. Eisner, "Leaf folding in a sensitive plant: A defensive thorn-exposure mechanism?" *Proc. Nat. Acad. Sci.*, vol. 78, no. 1, pp. 402–404, Jan. 1981.
[11] H. Kobayashi, B. Kresling, and J. F. V. Vincent, "The geometry of unfolding tree leaves," *Proc. Roy. Soc. B, Biol. Sci.*, vol. 265, no. 1391, pp. 147–154, Jan. 1998.
[12] M. Karplus and D. L. Weaver, "Protein-folding dynamics," *Nature*, vol. 260, no. 5, pp. 404–406, Apr. 1976.
[13] E. Hawkes *et al.*, "Programmable matter by folding," *Proc. Nat. Acad. Sci.*, vol. 13, pp. 12 441–12 445, Jun. 2010.
[14] A. Firouzeh, Y. Sun, H. Lee, and J. Paik, "Sensor and actuator integrated low-profile robotic origami," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 4937–4944.
[15] B. An and D. Rus, "Designing and programming self-folding sheets," *Robot. Auton. Syst.*, vol. 62, no. 7, pp. 976–1001, Jul. 2014.
[16] B. An, N. Benbernou, E. D. Demaine, and D. Rus, "Planning to fold multiple objects from a single self-folding sheet," *Robotica*, vol. 29, no. 01, pp. 87–102, Jan. 2011.
[17] S. Felton, M. Tolley, E. Demaine, D. Rus, and R. Wood, "A method for building self-folding machines," *Science*, vol. 345, no. 6, pp. 644–646, Aug. 2014.
[18] S. M. Felton, M. T. Tolley, C. D. Onal, D. Rus, and R. J. Wood, "Robot self-assembly by folding: A printed inchworm robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 277–282.
[19] S. Miyashita, S. Guitron, M. Ludersdorfer, C. R. Sung, and D. Rus, "An untethered miniature origami robot that self-folds, walks, swims, and degrades," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 1490–1496.
[20] S. Miyashita, L. Meeker, M. Gouldi, Y. Kawahara, and D. Rus, "Self-folding printable elastic electric devices: Resistor, capacitor, and inductor," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 1446–1453.
[21] K. Malachowski, M. Jamal, Q. Jin, B. Polat, C. J. Morris, and D. H. Gracias, "Self-folding single cell grippers," *Nano Lett.*, vol. 14, no. 7, pp. 4164–4170, Jul. 2014.
[22] B. An *et al.*,"An end-to-end approach to making self-folded 3D surface shapes by uniform heating," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 1466–1473.
[23] N. M. Benbernou, E. D. Demaine, M. L. Demaine, and A. Ovadya, "Universal hinge patterns to fold orthogonal shapes," in *Proc. 5th Int. Conf. Origami Sci. Math. Edu.*, Jul. 13–17, 2010, pp. 405–420.
[24] E. D. Demaine and J. O'Rourke, *Geometric Folding Algorithms (Series Linkages, Origami, Polyhedra)*. Cambridge, U.K.: Cambridge Univ. Press, Aug. 2008.
[25] T. Tachi, "Origamizing polyhedral surfaces," *IEEE Trans. Vis. Comput. Graph.*, vol. 16, no. 2, pp. 298–311, Mar. 2010.
[26] E. D. Demaine, S. L. Devadoss, J. S. B. Mitchell, and J. O'Rourke, "Continuous foldability of polygonal paper," in *Proc. 16th Can. Conf. Comput. Geometry*, 2004, pp. 64–67.
[27] E. Demaine, M. Demaine, and J. Ku, "Folding any orthogonal maze," in *Origami 5*. Boca Raton, FL, USA: CRC Press, Nov. 2011, pp. 449–454.
[28] E. Demaine, S. Fekete, and R. Lang, "Circle packing for origami design is hard," in *Origami 5*. Boca Raton, FL, USA: CRC Press, Nov. 2011, pp. 609–626.
[29] C. Sung, E. D. Demaine, M. L. Demaine, and D. Rus, "Edge-compositions of 3d surfaces," *J. Mech. Des.*, vol. 135, no. 11, Nov. 2013, Art. no. 111001.
[30] M. W. Bern and B. Hayes, "The complexity of flat origami," in *Proc. 7th Annu. ACM-SIAM Symp. Discrete Algorithms*, 1996, pp. 175–183.
[31] L. Ionov, "Soft microorigami: Self-folding polymer films," *Soft Matter*, vol. 7, no. 15, pp. 6786–6791, 2011.
[32] Y. W. Yi and C. Liu, "Magnetic actuation of hinged microstructures," *J. Microelectromech. Syst.*, vol. 8, no. 1, pp. 10–17, Mar. 1999.
[33] S. M. Felton, M. T. Tolley, and R. J. Wood, "Mechanically programmed self-folding at the millimeter scale," in *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, 2014, pp. 1232–1237.
[34] S. M. Felton *et al.*, "Self-folding with shape memory composites," *Soft Matter*, vol. 9, no. 32, pp. 7688–7694, 2013.
[35] S. Miyashita, C. D. Onal, and D. Rus, "Self-pop-up cylindrical structure by global heating," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 4065–4071.
[36] M. T. Tolley, S. M. Felton, S. Miyashita, D. Aukes, D. Rus, and R. J. Wood, "Self-folding origami: Shape memory composites activated by uniform heating," *Smart Mater. Struct.*, vol. 23, no. 9, 2014, Art. no. 094006.
[37] Y. Liu, J. K. Boyles, J. Genzer, and M. D. Dickey, "Self-folding of polymer sheets using local light absorption," *Soft Matter*, vol. 8, no. 6, pp. 1764–1769, 2012.
[38] K. Kuribayashi-Shigetomi, H. Onoe, and S. Takeuchi, "Cell Origami: Self-folding of three-dimensional cell-laden microstructures driven by cell traction force," *PLoS ONE*, vol. 7, Dec. 2012, Art. no. 51085.
[39] T. G. Leong, P. A. Lester, T. L. Koh, E. K. Call, and D. H. Gracias, "Surface tension-driven self-folding polyhedra," *Langmuir*, vol. 23, no. 17, pp. 8747–8751, Aug. 2007.
[40] K. Yasu and M. Inami, "POPAPY: Instant paper craft made up in a microwave oven," in *Advances in Computer Entertainment*. Berlin, Germany: Springer, 2012, pp. 406–420.
[41] Q. Ge, C. K. Dunn, H. J. Qi, and M. L. Dunn, "Active origami by 4D printing," *Smart Mater. Struct.*, vol. 23, no. 9, Sep. 2014, Art. no. 094007.
[42] Y. Mao, K. Yu, M. S. Isakov, J. Wu, M. L. Dunn, and H. Jerry Qi, "Sequential self-folding structures by 3d printed digital shape memory polymers," *Sci. Rep.*, vol. 5, 2015, Art. no. 13616. [Online]. Available: http://www.nature.com/articles/srep13616
[43] P. Sitthi-Amorn *et al.*, "MultiFab: A machine vision assisted platform for multi-material 3D printing," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 129–129:11, 2015.
[44] T. Tachi, "Origamizing polyhedral surfaces," *IEEE Trans. Vis. Comput. Graph.*, vol. 16, no. 2, pp. 298–311, Mar. 2010. [Online]. Available: https://doi.org/10.1109/TVCG.2009.67
[45] S. Takahashi, H.-Y. Wu, S. H. Saw, C.-C. Lin, and H.-C. Yen, "Optimized topological surgery for unfolding 3d meshes," *Comput. Graph. Forum*, vol. 30, no. 7, pp. 2077–2086, Nov. 2011.
[46] T. Tachi, "Simulation of rigid origami," in *Origami 4*. Boca Raton, FL, USA: CRC Press, Apr. 2011, pp. 175–187.
[47] M. Bern, E. D. Demaine, D. Eppstein, E. Kuo, A. Mantler, and J. Snoeyink, "Ununfoldable polyhedra with convex faces," *Comput. Geom.*, vol. 24, no. 2, pp. 51–62, Feb. 2003.
[48] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Syst. Tech. J.*, vol. 36, no. 6, pp. 1389–1401, 1957.
[49] E. D. Demaine and J. O'Rourke, "Flattening polyhedra," in *Geometric Folding Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2010, pp. 279–284.

Byoungkwon An received the B.Sc. degree in physics from Soongsil University, Seoul, South Korea, in 2004 and the M.Sc. degree in computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2011.

He is interested in algorithms, computation, and computational geometry, including computational self-assembly of biological molecular and artificial programmable matter, computational origami, origami computing, and distributed systems and robotics.

Shuhei Miyashita (M'11) received the Ph.D. degree in mathematics and natural science from the University of Zurich, Zurich, Switzerland, in 2011.

He is currently a Lecturer at the University of York, York, U.K. Prior to this position, he was a Postdoctoral Research Associate at the Massachusetts Institute of Technology, Cambridge, MA, USA, and Carnegie Mellon University, Pittsburgh, PA, USA. His research interests include self-assembly, microrobotics, self-reconfigurable systems, biomedical robotics, and origin of life.



Aaron Ong (S'15) received the B.Sc. degree in biomedical engineering from the University of California, San Diego, CA, USA, in 2018. He is currently working toward the M.Eng. degree in mechanical engineering at the University of California, Berkeley, CA, USA.

From 2015 to 2018, he was involved in research at the Bioinspired Robotics and Design Lab and worked on self-folding and soft robots. Previously, he was a mechanical design intern with Tesla Inc., Palo Alto, CA, USA and a research intern with the Harvard Microrobotics Lab, Cambridge, MA, USA. His current research interests include design, manufacturing, and assembly of bioinspired systems.

Michael T. Tolley (S'09–M'11) received the B.Eng. degree in mechanical engineering from McGill University, Montreal, QC, Canada, in 2005, and the M.S. and Ph.D. degrees in mechanical engineering with a minor in computer science from Cornell University, Ithaca, NY, USA, in 2009 and 2011, respectively.

He was a Postdoctoral Associate at the Harvard Microrobotics Lab and the Wyss Institute for Biologically Inspired Engineering, Harvard University, Boston, MA, USA, from 2011 to 2014. He is currently an Assistant Professor of Mechanical Engineering and Materials Science, and the Founder and Director of the Bioinspired Robotics and Design Lab, University of California, San Diego, CA, USA. His research focuses on the design, rapid fabrication, and control of biologically inspired robotics, with topics including actuation for soft bodied robots, soft sensors, origami-inspired fabrication, and autonomous self-assembly.

Dr. Tolley received the Young Investigator Program Award from the US Office of Naval Research, and the Nontenured Faculty Award from the 3M Company.

Martin L. Demaine is an Artist and Mathematician. He started the first private hot glass studio in Canada and has been called the father of Canadian glass. Since 2005, he has been the Angelika and Barton Weller Artist-in-Residence with the Massachusetts Institute of Technology, Cambridge, MA, USA. He works in paper, glass, and other material. He uses his exploration in sculpture to help visualize and understand unsolved problems in mathematics, and his scientific abilities to inspire new art forms. His artistic work includes curved origami sculptures in the permanent collections of the Museum of Modern Art (MoMA) in New York, and the Renwick Gallery in the Smithsonian. His scientific work includes more than 60 published joint papers, including several about combining mathematics and art.

Erik D. Demaine received the B.Sc. degree from Dalhousie University, Halifax, NS, Canada, in 1995, and the M.Math. and Ph.D. degrees in computer science from the University of Waterloo, Waterloo, ON, Canada, in 1996 and 2001, respectively.

Since 2001, he has been a Professor of Computer Science at the Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests range throughout algorithms, from data structures for improving web searches to the geometry of understanding how proteins fold to the computational difficulty of playing games. In 2003, he received a MacArthur Fellowship as a "computational geometer tackling and solving difficult problems related to folding and bending—moving readily between the theoretical and the playful, with a keen eye to revealing the former in the latter." He co-authored a book about the theory of folding, together with Joseph O'Rourke *Geometric Folding Algorithms* (Cambridge University Press, 2007), and a book about the computational complexity of games, together with Robert Hearn *Games, Puzzles, and Computation* (Boca Raton, FL, USA: CRC Press, 2009).

Robert J. Wood (M'01–SM'16) received the M.S. and Ph.D. degrees from the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, CA, USA, in 2001 and 2004, respectively.

He is currently the Charles River Professor of Engineering and Applied Sciences at the Harvard John A. Paulson School of Engineering and Applied Sciences, a Founding Core Faculty Member of the Wyss Institute for Biologically Inspired Engineering at Harvard, Harvard University, Cambridge, MA, USA, and a National Geographic Explorer. His current research interests include microrobotics, soft and wearable robots, and bioinspired robotics.

Daniela Rus (F'16) received the Ph.D. degree in computer science from Cornell University, Ithaca, NY, USA, in 1993.

She is the Andrew (1956) and Erna Viterbi Professor of Electrical Engineering and Computer Science and the Director of the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. Prior to joining MIT, she was a Professor at the Computer Science Department, Dartmouth College, Hanover, NH, USA. Her research interests include robotics, mobile computing, and data science.

Dr. Rus is a Class of 2002 MacArthur Fellow, a Fellow of ACM and AAAI, and a member of the National Academy of Engineering.