An End-to-End Approach to Self-Folding Origami Structures by Uniform Heat

Byoungkwon An, Shuhei Miyashita, Aaron Ong, Daniel M. Aukes, Michael T. Tolley, Erik D. Demaine, Martin L. Demaine, Robert J. Wood and Daniela Rus

***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 by uniform heat are robotic sheets composed of rigid tiles and joint actuators. When they are exposed to heat, each joint folds into a pre-programmed angle. Those folding motions transform themselves into a structure, which can be used as body of 3D origami robots, including walkers, analog circuits, rotational actuators, and micro cell grippers. Given a 3D 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 3D model. We discuss (1) the design algorithm generating multiple-step self-folding sheet designs, (2) verification of the algorithm running in $O(n^2 \times m)$ time, where n and m are vertices and face numbers, respectively, (3) implementation of the algorithm, and (4) experimental results, several self-folded 3D structures with up to 55 faces and two sequential folding steps.**

***Index Terms*—Cellular and Modular Robots; Smart Actuators; Printable Origami Robots; Self-Folding**

I. Introduction

Folding-based design is a method for fabricating a device with straight or curved folding crease lines. Each folding of an original material on a crease line yields dimensional transformations of the material from one- or two-dimensions to three-dimensions. Due to this characteristic, folding-based designs are widely used for engineering applications, including space projects [1], [2], soft-robots [3], micro-scale fabrications [4], [5], and microrobotics [6]–[8]. Such designs are also found in nature, for example, in insect wings [9], leaves [10], [11], and proteins [12].

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 an angle. These local foldings yield a global transformation of the sheets [16]. Self-folding origami structures by uniform heat receive heat as a signal. When the robotic 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 micro-scale grippers that hold a single cell [21]. Because 2D fabrication processes are used for making 3D self-folding robots, the process of fabricating complex structures becomes relatively simple. A self-folding sheet transforms itself into arbitrary 3D surfacial shapes on-demand. This process enables rapid prototyping with relatively lower fabrication cost.

Folding fabricated robotic sheets into 3D 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 an origami robotic sheet are 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 3D input models, the algorithm outputs the layouts of the self-folding sheets. By printing the algorithmically designed layouts, the user builds robotic sheets. Upon being baked in an oven, these sheets transform themselves into physical devices (Fig. 1). We also developed 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 3D 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 3D model as an input and computes layouts of single- or multiple-step self-folding sheets in $O(n^2 \times m)$ time, where n and m are the numbers of vertices and faces in the 3D model,
3) an implemented design automation system including the design algorithm, and
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 2 steps.

The remainder of this paper is organized as follows: Section III describes and analyzes its model for self-folding sheets. Section IV describes the design algorithm and its implementation. It then presents the single-step self-folding sheet experiments. This section explains the details of how to compute and control the angles. Section VI-A explores the algorithm, implementation and experiments of the multiple-step self-folding sheets. Section VII discusses the lessons learned and future research.

B. An is with Autodesk Research, San Francisco, CA 94111, USA, www.drancom.com, kwon.an@autodesk.com
S. Miyashita, E. Demaine, M. Demaine, and D. Rus are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.
A. Ong and M. Tolley are with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA 92093, USA.
D. Aukes and R. Wood is with the School of Engineering and Applied Sciences and Wyss Institute for Biologically Inspired Engineering, Harvard University, Cambridge, MA 02138, USA.
Manuscript received mmmmm dd, yyyy; revised mmmmm dd, yyyy.

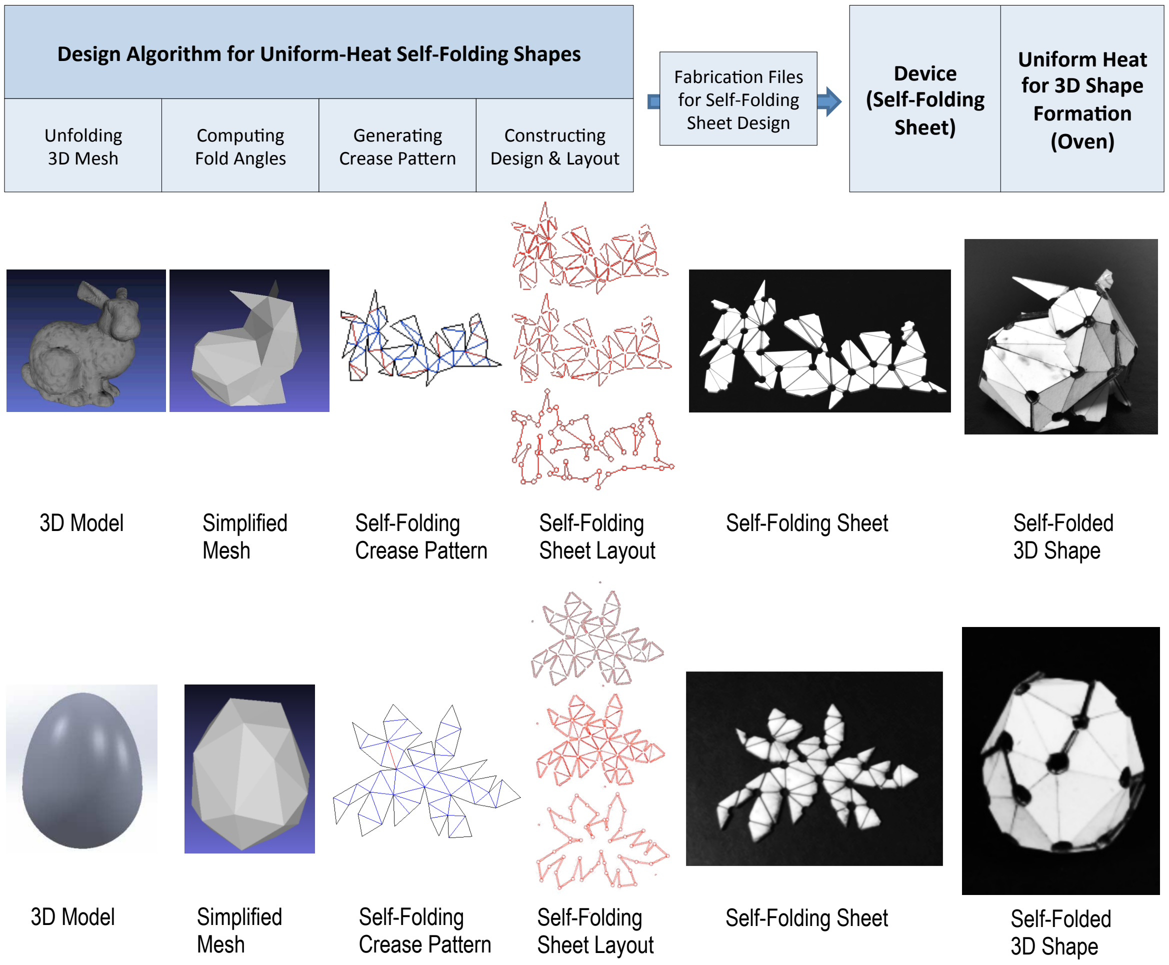
Design Algorithm for Uniform-Heat Self-Folding Shapes
Unfolding 3D Mesh
Computing Fold Angles
Generating Crease Pattern
Constructing Design & Layout
Fabrication Files for Self-Folding Sheet Design
Device (Self-Folding Sheet)
Uniform Heat for 3D Shape Formation (Oven)
3D Model
Simplified Mesh
Self-Folding Crease Pattern
Self-Folding Sheet Layout
Self-Folding Sheet
Self-Folded 3D Shape
3D Model
Simplified Mesh
Self-Folding Crease Pattern
Self-Folding Sheet Layout
Self-Folding Sheet
Self-Folded 3D Shape

Fig. 1. The visual overview of the self-folding sheet development process. The middle and bottom lines show the data transformation to develop a self-folding bunny and egg.

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 re-programmability (re-usability), 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 sheet using a uniform energy source is a recent research direction of interest. [23]–[29] introduce various computational origami designs, and [13], [30] discuss the theoretical and experimental complexity of folding patterns. This paper introduces a design algorithm and its verification as well as a compilation-like approach to automate fabrication of self-folding sheets.

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, 3D printing technology has been proposed as an on-demand synthesis method for self-folding shape memory polymers [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].

III. Model: Self-Folding Sheet Activated by Uniform Heating

A uniform heat self-folding sheet is defined as a crease pattern composed of cuts (outlines) and folding edges (hinges), as shown in Fig 2. Each edge contains a fold angle and folding group. All the edges of the sheet are controlled using global

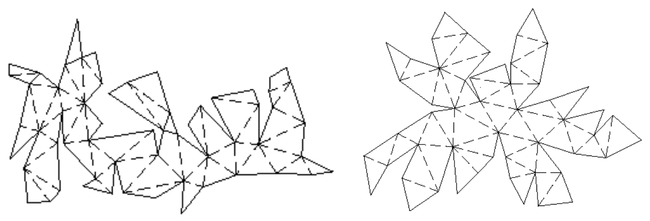

Fig. 2. Visualized self-folding crease pattern representing a bunny shape (left) and an egg shape (right). The solid lines are cuts and the dashed lines are edges (hinges). Each edge contains a fold angle.

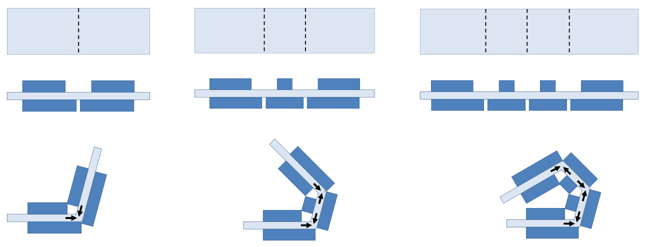

Fig. 3. Three examples of simple self-folding sheets embedding one, two, and three actuators. The arrows show the shrinking directions.

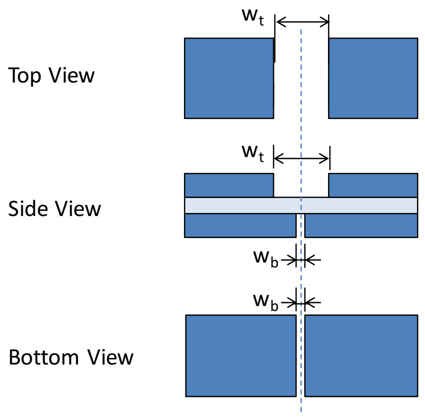

Fig. 4. Overview of self-folding actuator model.

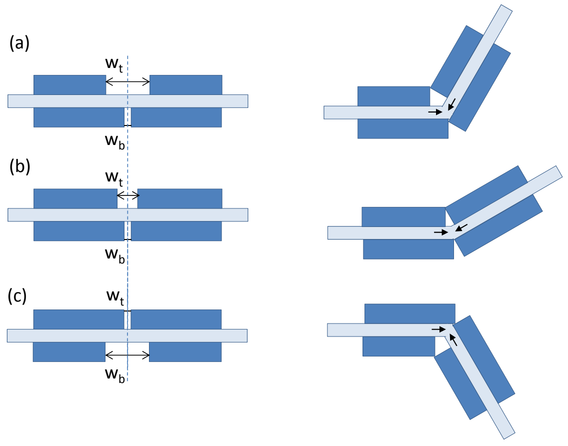

Fig. 5. Actuator model for self-folding. (Left) before activation and (Right) after activiation. The arrows show the shrinking directions.

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 p degrees, all the edges of the p degree group are self-folded, and when the uniform heat temperature reaches q degrees where $q > p$, all the edges of the q degree group are self-folded.

A. Fold Angle

In this paper, a *folding actuator* is composed of three layers (Fig. 3, 5). 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 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 Fig. 4, 5)

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. 3 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. 5). For example, if the gap (Fig. 5(a)) is wider than the gap at another location (Fig. 5(b)), the former (Fig. 5(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 (Fig. 5(c)).

B. Time Step

We achieve sequential folding by using a multi-material shrinking layer as the middle layer. The layer is segmented in several regions, each capable of shrinking in different temperature range. In other words, each material shrinks sequentially after a material finishes the shrinking. While the temperature increases, different regions of the layer shrink at different times. Fig. 6 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 can build multi-material middle layers manually, with jigsaw-puzzle-like placement. A multi-material printer, like MultiFab [43] or Objet Connex 500, can be used to automate this fabrication.

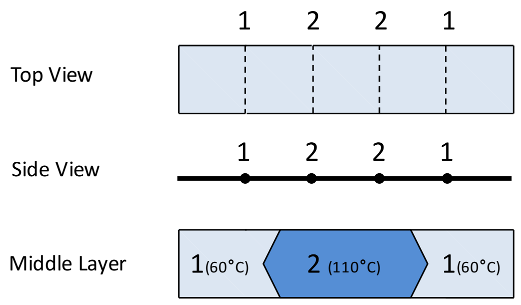

Fig. 6. Self-folding sheet with two-step folding. The middle layer is composed of two different materials: Material 1 reacts at 60°C and material 2 reacts at 110°C.

IV. DESIGN ALGORITHM

A. Compile Structural Information

The *self-folding sheet design algorithm* converts a shape represented as a 3D mesh[1] or a 3D origami design[2] into a *self-folding sheet design*, which is the structural layout of the self-folding sheet. Just as an origami crease pattern contains the information required to produce a folded origami object, a self-folding sheet design contains information to automatically fabricate an object when it is subjected to uniform heat. The design is composed of the layouts of each layer. The layouts contain the folding information. The self-folding sheet is printed or fabricated according to the design. Given a 3D mesh, the algorithm compiles the design of the self-folding sheet following these steps (Fig. 7): (1) unfolding a given 3D structure, (2) computing fold angles, (3) constructing a self-folding sheet crease pattern, (4) constructing a self-folding sheet design, and (5) constructing a self-folding sheet layout. The notations of the paper are listed in Table I.

1) Unfolding a 3D Mesh: The objective of this step is to compute the unfolding of a given 3D shape. Several algorithms exist to unfold 3D meshes or 3D 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 3D mesh in a graph and unfold it using Prim's algorithm (a minimum spanning tree algorithm) [48]. As the algorithm unfolds the 3D mesh, it maintains the relationship between the vertices of the unfolded 2D structure and the 3D 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 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.

[1]A polygon mesh is a collection of faces that defines a polyhedral object.

[2]An origami design is a folded state of a paper structure encoded with a crease pattern and folded angles [16].

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

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
m	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$(-180 \le u \le 180)$;$u \in D$
$d = (w_t, w_c, w_b)$	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 sheet design
$L = (L_t, L_c, L_B)$	Self-folding sheet layout
$L_t = (V_t, E_t)$	The top layer of L
$L_c = (V_c, E_c)$	The center layer of L
$L_b = (V_b, E_b)$	The bottom layer of L

2) Computing Fold Angles: The goal of this step is to compute the fold angles associated with all the edges of a given mesh (Fig. 10). 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 (Fig. 8). 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 faces 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 (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. □

3) Constructing the Self-Folding Crease Pattern: This step takes two inputs, a set of nets and fold angles and computes

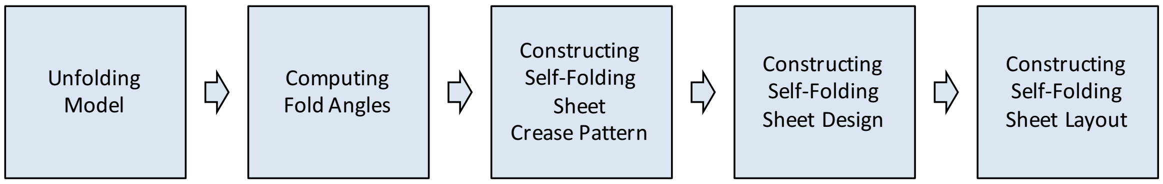
Unfolding Model
Computing Fold Angles
Constructing Self-Folding Sheet Crease Pattern
Constructing Self-Folding Sheet Design
Constructing Self-Folding Sheet Layout

Fig. 7. Overview of design pipeline

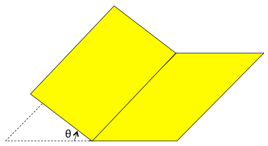

Fig. 8. The fold angle at a crease is the supplement of the dihedral angle.

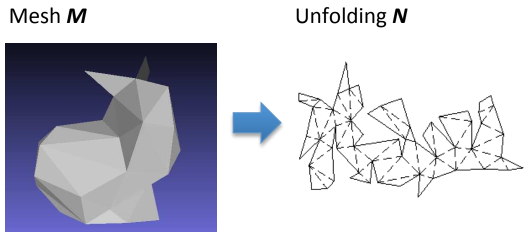

Fig. 9. Example of unfolding.

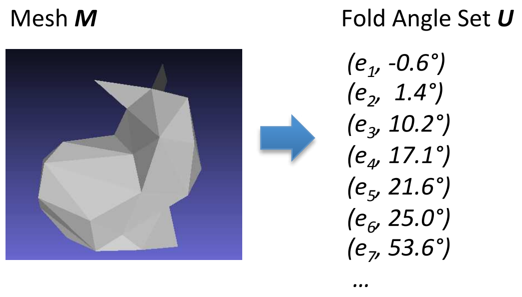

Fig. 10. Example of computing folding angles (Algorithm 1).

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
         $(v_1,v_2,...,v_k)$ are positioned counter-clockwise
         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 $u=acos(\frac{n_1 \cdot n_2}{|n_1||n_2|})$, 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 $U$.
```

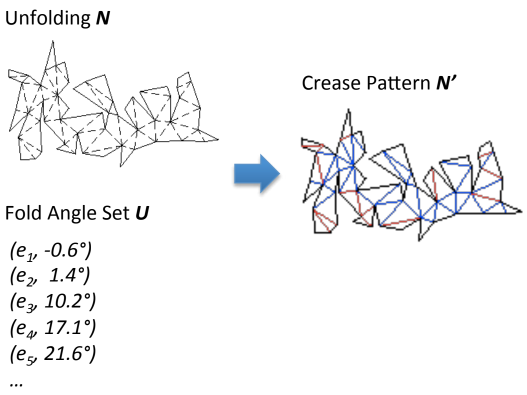

Fig. 11. Example of constructing a self-folding sheet crease pattern (Algorithm 2).

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 $N'=(V',E',F',T,U')$
```

a *self-folding crease pattern* (the abstracted self-folding information), as shown in Fig. 11. In this section, we show that Algorithm 2 constructs a correct self-folding crease pattern (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 $N'=(V',E',F',T,U')$ 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 $L=\{f'_1,f'_2,...,f'_k\}$, 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 $\{f'_1,f'_2,...,f'_t\}\subseteq L$. Let $F(M_t)$ be $\{f_i=f(f'_i)\mid f'_i\in L_t\}$. Let $F(M'_t)$ be $\{f''_1,f''_2,...,f''_t\}$, where each f''_i is a face of the folded state of $f'_i\in L$.

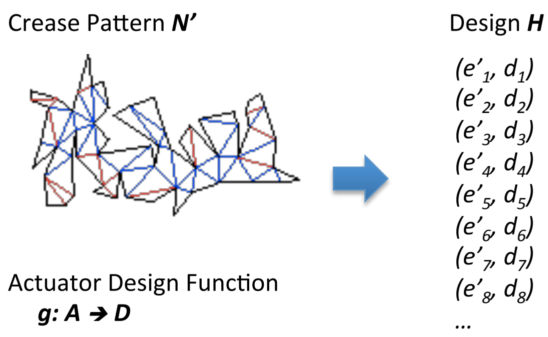

Fig. 12. Example of constructing a self-folding sheet design (Algorithm 3).

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 $t = k+1$.

The hypothesis states that $M'_k = M_k$, and f_{k+1}, 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 Sheet Design: Given a self-folding sheet crease pattern and actuator design function, this step generates a self-folding sheet design (Fig. 12). A self-folding sheet design is an abstracted model of the actuators and the outlines.

A self-folding sheet 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 $\mathbb{R} \cup \{\varepsilon\}$ and are the gaps on the top, middle, and bottom sheets, respectively (Fig. 4). 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. 4 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 Sec. V.

Algorithm 3: Constructing Self-Folding Sheet 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 $H$
```

Lemma 5. *A self-folding crease pattern has a valid self-folding sheet design, computable in $O(n^2)$ time.*

Proof. Algorithm 3 constructs self-folding sheet design H. U' contains the fold angles of the edges, while T contains the types of the edges. Given angle u, $g(u)$ outputs actuator design d (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 (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 Sheet Layout: A self-folding sheet layout contains the graphical information of each layer. Given a self-folding sheet design, this step generates three layers of the layout (Fig. 13). For each element of a self-folding sheet design, an actuator layout of a layer is drawn (Fig. 14).

Lemma 6. *A self-folding sheet design has a valid self-folding sheet layout, computable in $O(n^2)$ time.*

Proof. The output of Algorithm 4 is the self-folding sheet 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 (Lemma 7). d contains correct actuator and outline information (Lemma 5), 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 7. *Each edge of a self-folding sheet design has a valid folding actuator.*

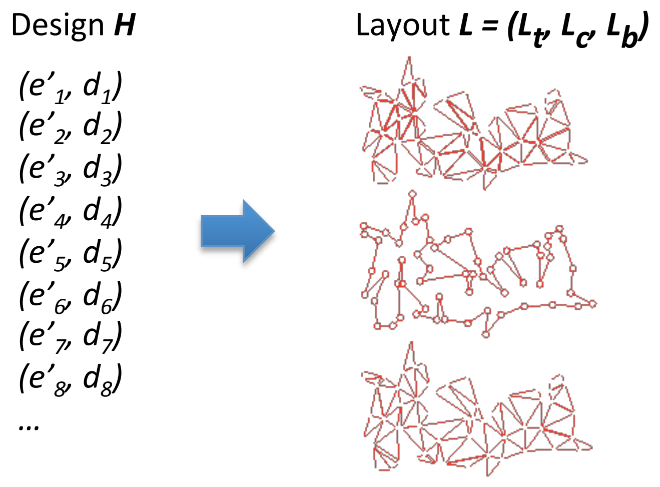

Fig. 13. Example of constructing layouts (Algorithm 4). The layout images (right) are drawn by the software implementation of the algorithm. While we implemented the algorithm, we added a feature, drawing the circles of the middle layer, to prevent the collisions at the vertices.

Algorithm 4: Drawing a Self-Folding Sheet 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 $L = (L_t, L_c, L_b)$
```

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 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. 5. Algorithm 5 draws a layer of an actuator, as shown in Fig. 14. Algorithm 5 is $O(1)$. Therefore, Steps (a)-(c) run in $O(1)$. □

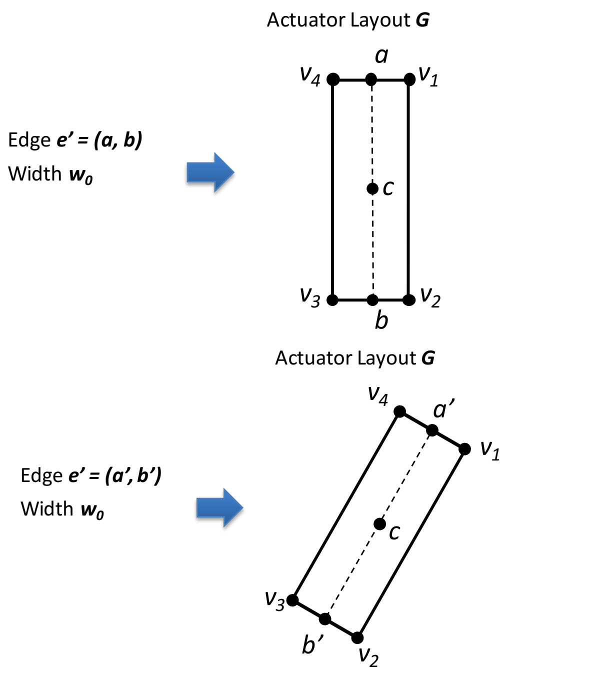

Fig. 14. Two examples of drawing an actuator (Algorithm 5). e' and w_0 are the input. w_0 is equal to the distances between v_1 and v_4, and between v_2 and v_3 of each example.

Algorithm 5: Construct Actuator Layout

```
Input  : $e' = \{a, b\}$, $w_0$
Output: $G = (V'', E'')$
1) If $w_0 = \varepsilon$, 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) $l \leftarrow (\text{length of } e')/2$
    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) $\theta \leftarrow atan2(y_b - y_a,\ x_b - x_a)$
    i) Rotate all vertices in $V''$ through $\theta$
    j) $c \leftarrow (a + b)/2$
    k) For each $v \in V''$, $v \leftarrow v + c$
```

B. Compile Time Step Information

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 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

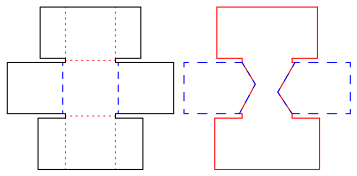

Fig. 15. Example of the construct of a multi-material middle layer (Algorithm 6). (Left) Crease Pattern N'. The red dotted lines are the first step folding creases, and the blue dashed lines are the second step folding creases. (Right) Middle Layer of Layout L_c. The red solid line polygon is shrinking 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.

steps. Our prior approach was to use an on-board electronic controller to selectively transfer was 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 sheets that transform themselves into user's desired shapes with multiple-folding steps. The sheets 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 sheet model with a multi-material shrinking layer (Fig. 6). 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 Sec. 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 layers of self-folding sheets are algorithmically designed. In this section, we describe an algorithm for generating the design of a middle layer (Fig. 15, 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.

Lemma 8. *A self-folding crease pattern with sequential folding steps has a valid shrinking layer design, computable in $O(m^2)$ time, where m is the number of faces.*

Proof. Algorithm 6 constructs a multi-material 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 5-6 assign all areas to a shrinking material. Step 7 removes unnecessary boundaries and merges the areas. Step 8 outputs B, the design of the multi-material 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

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 $E_c$
```

$\langle None \rangle$ to the groups of outline edges (Step 2) and the groups of new edges generated during the triangulation (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 $O(m)$. 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 (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.

$O(m)$ 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 5-6 run in $O(m^2)$.

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

Fig 16 shows the input and output of the algorithm. The inputs are the final folding states of the origami structures.

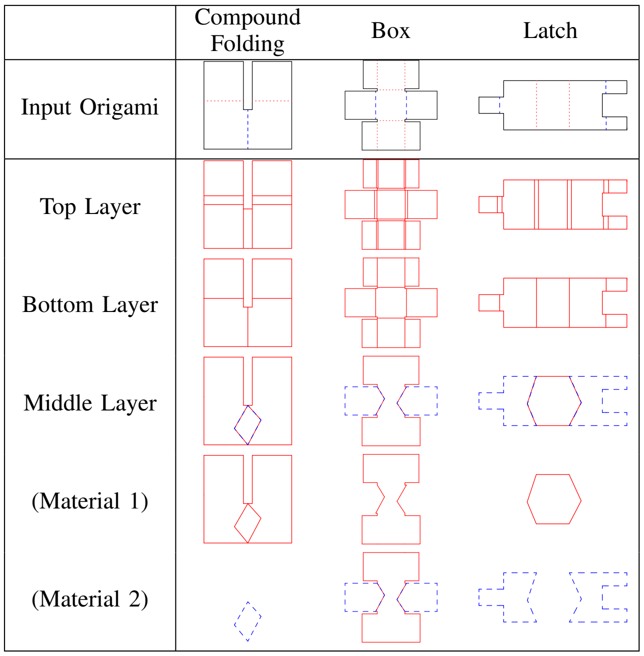

Fig. 16. Design of the multiple-step folding algorithm. (Input Origami) The red dotted lines are the first step folding creases, and the blue dashed lines are the second step folding creases. All lines are valley folds. (Top, Bottom Layers) The red solid lines are cut traces. The top and bottom layers are rigid materials. (Middle Layer) The red solid line polygons are shrinking material 1. The blue dashed line polygons are shrinking material 2. Material 1 and 2 react sequentially.

V. IMPLEMENTATION

A. Software for Compiling a Printable 2D Design

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

To support the various manufacturing processes of the self-folding sheets, the software supports script files to define the template of the fabrication files (outputs). To demonstrate automatically generated self-folding sheets 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.

Theoretical model covers geometric properties, such as collisions, edge types, or scalability. The geometry issues are characterized by material functions which 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

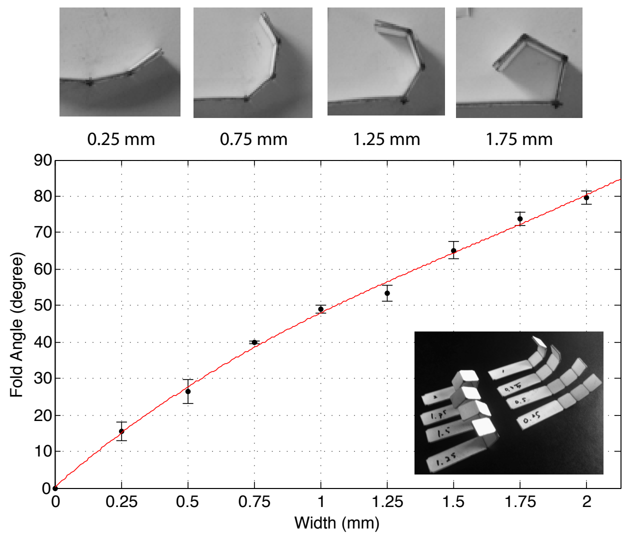

Fig. 17. Graph of an implemented actuator design function for the pin alignment process. The inset images show the test strips used to characterize the fold angle as a function of the size of the gap on the inner structural sheet. Each bar is the standard deviations from the average of the angles of three hinges (Tab.II).

TABLE II
FOLDING ANGLES

Gab (mm)	Angle1	Angle2	Angle3
0.25	11.58	14.6	20.09
0.5	22.85	23.34	33.25
0.75	39.49	40.44	39.79
1	47.6	51.16	48.42
1.25	56.62	49.36	54.51
1.5	69.57	61.39	64.39
1.75	77.44	72.88	71.36
2	80.34	82.53	76.13

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.

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) (Sec. 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 Def. 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 $\{d_1, d_2, d_3, ..., d_i, ...\}$ (Sec. 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. $s_0 = (0, (0, \varepsilon, 0)) \in S$,
5. if $(u, d) \in S$, then $g(u) = d$, and
6. if $(u, d) \notin S$, then $g(u) = (w(d_i) + \frac{u-u_i}{u_{i+1}-u_i} \times (w(d_{i+1}) - w(d_i)),\ \varepsilon,\ b(d_i) + \frac{u-u_i}{u_{i+1}-u_i} \times (b(d_{i+1}) - b(d_i)))$, where $u_i =$

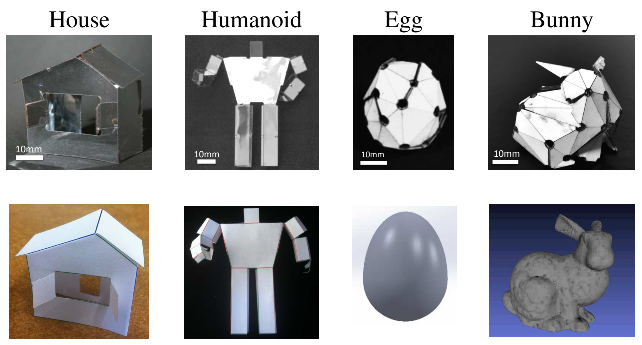

Fig. 18. (Top) Self-folded 3D shapes: the house, humanoid, egg, and bunny shapes. Each scale bar is 10mm. (Bottom) Input models. We modeled the house and humanoid designs with paper and coded them into origami designs. We modeled the egg and bunny shapes using CAD software.

$u(s_i)$,
$u_{i+1} = u(s_{i+1})$, $d_i = d(s_i)$, $d_{i+1} = d(s_{i+1})$, $u_i < u < u_{i+1}$, 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 $\frac{u-u_1}{u_2-u_1}$ and designs d_1 and d_2, where u_1 and u_2 are the angles of d_1 and d_2, and $u_1 < u < u_2$.

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 $0.25mm$–$2mm$ 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. 17. This method is modified from our prior work in [36]. This time, we automated the design process of the strips using our self-folding sheet design pipeline. We can easily generate another set of strips for a different range of gaps.

VI. Experiments

We evaluated the self-folding pipeline by building self-folding sheets for four shapes: a house, a humanoid, an egg, and a bunny (Fig. 18). The bunny is the most complex shape we self-folded by heating. Given the 3D models of these input shapes, the pipeline outputs a set of .dxf files containing the layout of each self-folding sheet. We built and baked each self-folding sheet 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 3D shape of the house was composed of 9 faces, and its 2D unfolding contained 8 actuators. The 3D shape of the humanoid was composed of 41 faces, and its 2D sheet contained 44 self-folding actuators (Table III). Fig. 19 (a), (b) shows the fabrication files of the house shape and the humanoid shape.

The egg shape was modeled in CAD software (Solidworks, Dassault Systemes SolidWorks Corp.) and exported as a 3D mesh with 2,538 faces. We reduced the number of the faces to 50 (MeshLab, Visual Computing Lab, ISTI, CNR), and then unfolded it with our software. The 2D sheet of the egg contained 48 actuators (Table III). We generated the fabrication files for the egg shape from this model. Fig. 19 (c) shows the fabrication files of the egg shape.

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°

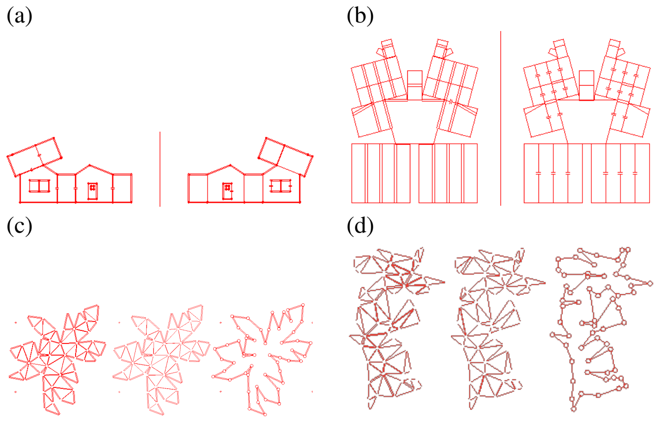

Fig. 19. Fabrication layout for self-folding sheets. (a), (b) Fabrication layouts of the folding alignment process generated for the house and humanoid. The left and right sides of each house and humanoid are the top and bottom layers, respectively. The line in the center guides the folding alignment, while the top layer and the bottom layer are sandwiched. (c), (d) 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.

For the bunny shape, we downloaded the 3D 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. 19(d) shows the fabrication files of the bunny shape.

After we built the fabrication files, we manufactured physical self-folding sheets for the house, humanoid, egg, and bunny shapes. 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 (Table IV).

Each shape has various fold angles. The distributions of these angles are shown in the histograms in Fig. 20. The angles of the humanoid are in the widest range, although the most frequent angles are 90°. The bunny includes the most diverse angles in both valley and mountain foldings.

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

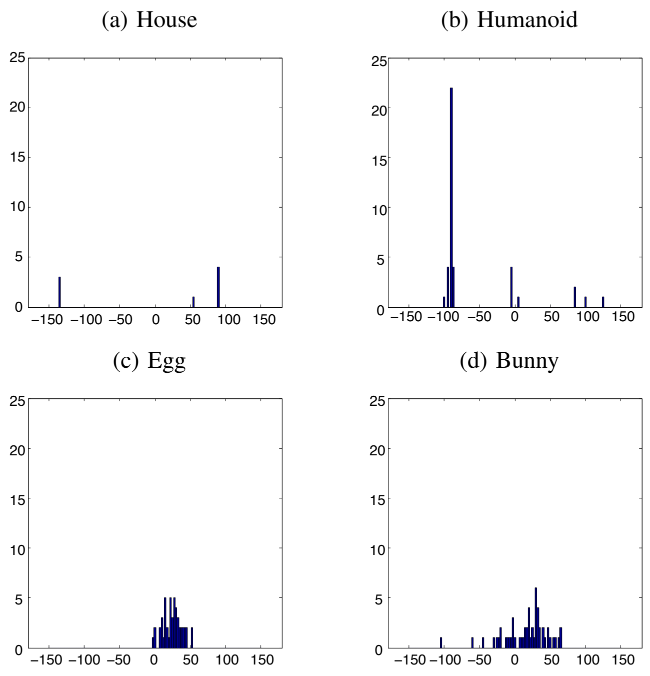

Fig. 20. The histograms of the fold angles. The x-axis is fold angles. The y-axis is frequency. The width is 2.5°.

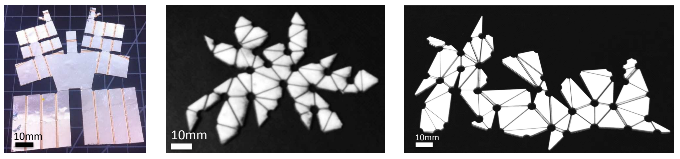

Fig. 21. Self-folding sheets (before baking) for humanoid (left), egg (center), and bunny (right). Each scale bar is 10mm.

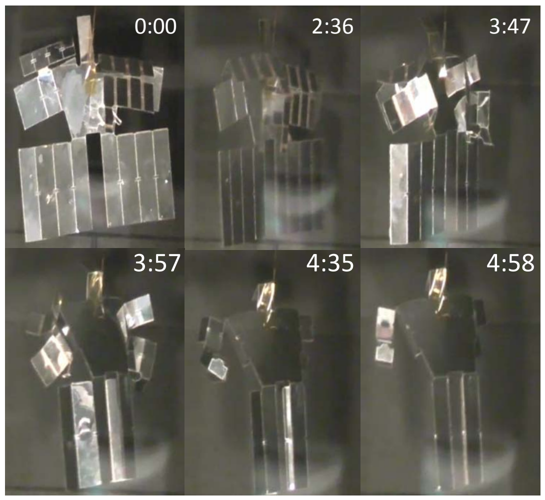

Fig. 22. Frames from experiment of the self-folding humanoid shape by uniform heating. The sheet was built with the folding alignment process. The time elapsed since exposure to uniform heating is indicated in the upper-right corner of each frame (in minutes and seconds).

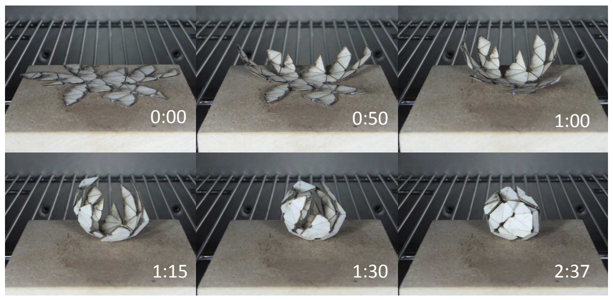

Fig. 23. Frames from the experiment of the self-folding egg shape by uniform heating. The sheet was built with the folding alignment process. The time elapsed since exposure to uniform heating is indicated in the lower-right corner of each frame (in minutes and seconds).

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[4]. Fig. 22, 23, and 24 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 10 self-folding bunnies and 8 eggs and measured their well-formed rates. When all vertices meet in a 3mm (circle size of the vertices) radius circle, the shape is called a well-folded shape; otherwise, we call it a failed shape.

Our self-folding algorithm designed self-folding 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 folded on a plate.

Using the proposed pipeline, the self-folded structures were rapidly designed and built (Table V). The computing time for each model was less than 0.5 sec. 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 2D self-folding 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 10 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% (Table VI, Fig. 25).

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

[4]See [36] 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 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)

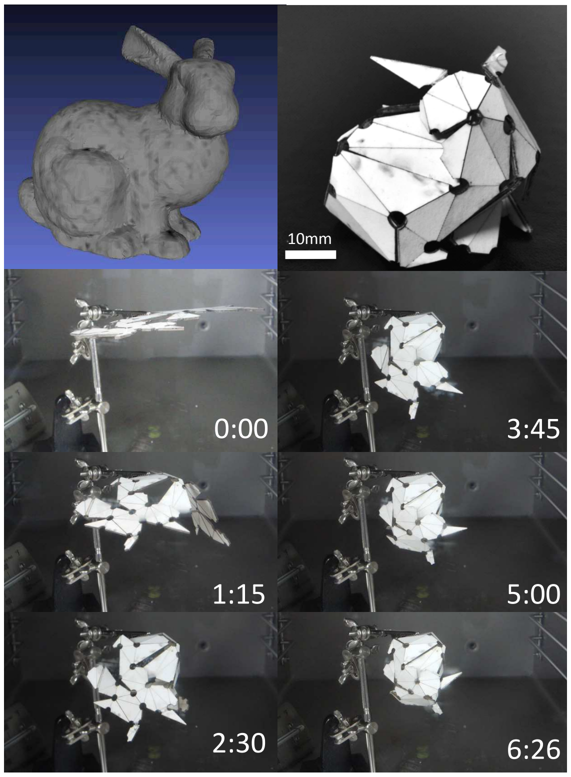

Fig. 24. Self-folding Stanford Bunny. (Top-left) Input 3D graphic model. (Top-right) 3D self-folded structure. (Bottom) Frames from the experiment of self-folding by uniform heating. The time elapsed since exposure to uniform heating is indicated in the lower-right corner of each frame (in minutes and seconds).

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 (2.30GHz)
RAM	4 GB
Storage	500GB 5400rpm 2.5” HDD (TOSHIBA MK5076GSX)
Graphics	Intel HD Graphics 3000

pipeline generates the design. Alternatively, we can use a multiple-step folding algorithm.

A. *Multiple-Step Self-Folding*

1) Compound Folding: To achieve multiple-step self-folding, two materials, PVC (SMP 1) that reacts at ∼65°C and polyolefin (SMP 2) that reacts at 80°C, are used for actuation to enable a two-step self-folding process. The experimental result of compound self-folding is shown in Fig. 27. The experiment was conducted on the water in an oven, and the temperature was raised to 80°C from room temperature. Note that the elapsed time shown was measured starting from the time deformation on creases was observed. First, two creases actuated by PVC started self-folding (33 sec - 53 sec), then a crease actuated by polyolefin followed (86 sec - 96 sec). As a result, the structure was folded into a fourth of the original size (105 sec).

2) Box and Latch: We designed two self-folding shapes to demonstrate the significance of sequential folding. The first design presents a folded box (Fig. 28(d)), which requires sequential folding, while the second addresses the issue of latching in order to lock the assembled structure (Fig. 28(h)); both shapes require a two-stage folding sequence for proper assembly (unsuccessful single-stage versions of these designs are shown in Fig. 28(b) and 28(f)).

To achieve sequential folding, we used a multi-material layer (Fig. 6) composed of polyolefin (SMP 1) for the first

TABLE VI
FAILURE RATES

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

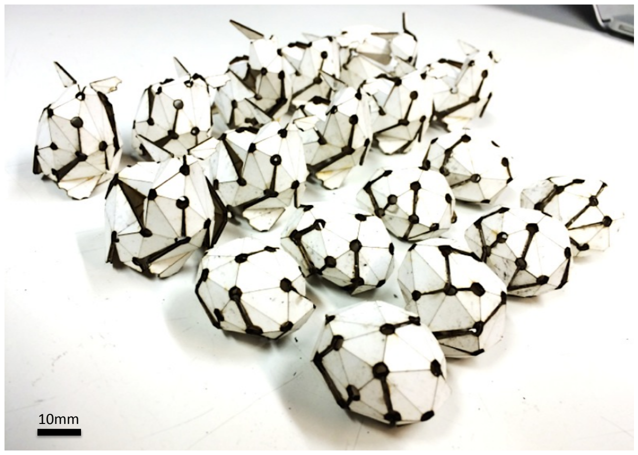

Fig. 25. Self-folded bunny and egg shapes. The scale bar is 10mm.

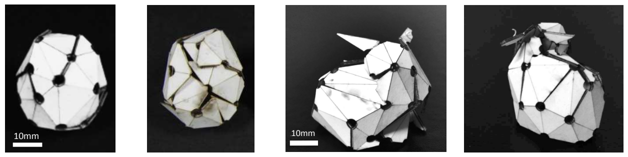

Fig. 26. Front and back sides of the self-folded egg and bunny. Each scale bar is 10mm.

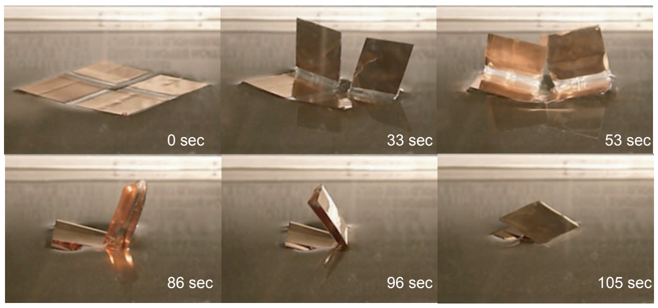

Fig. 27. Frames from the experiment of compound folding. Two actuation material differentiate the timings of self-folding and enable compound folding.

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

stage of folding and pre-strained polystyrene (SMP 2) for the second. Fig. 28 (a), (c), (e), (g) show 2D laminates, where the transparent hinges show the region composed of polyolefin, and the solid-colored hinges show the region composed of pre-strained polystyrene. To fabricate these laminates we used pin-alignment (Fig. 4). We cut 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 (12qt. Fagor Halogen) until the final structure was achieved.

We performed eight trials for each shape with the oven starting from room temperature and set to a target temperature of 175°C. As the oven heated, the region involving polyolefin actuated first at an average time and temperature of 82 sec

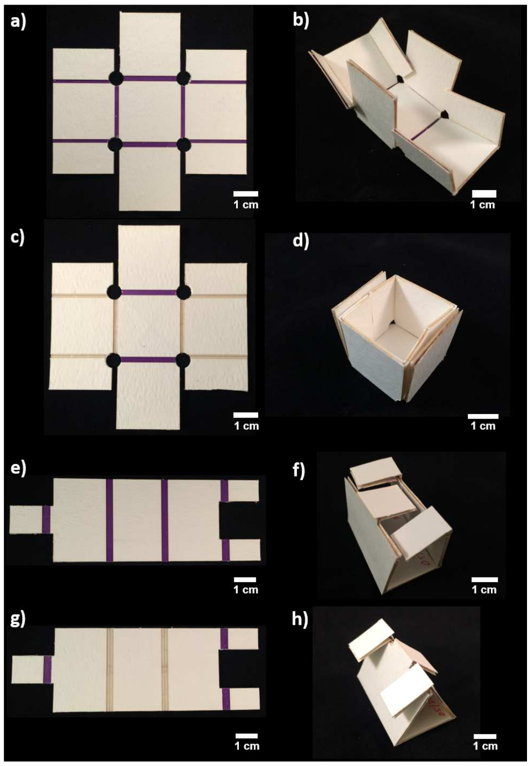

Fig. 28. Unfolded (left column) and folded (right column) structures for the box and latch. (a) single-material middle layer for box, (b) failed box assembly for a single-material middle layer, (c) multi-material middle layer for box, (d) successful sequential folding of box for multi-material middle layer, (e) single-material middle layer for latch, (f) failed latch assembly for a single-material middle layer, (g) multi-material middle layer for latch, (h) successful sequential folding of latch for multi-material middle layer.

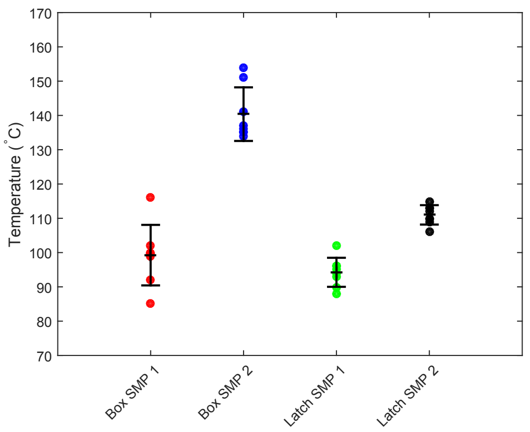

Fig. 29. Relative temperatures of actuations for SMP 1 and 2 for box and latch shapes. SMP 1 and 2 are used for the first and second folding steps of the shape, respectively. Each bar is the standard deviations from the average. Eight points of each material represent the relative temperatures of 8 trials of each shape.

at 99°C for the box and 90 sec at 94°C for the latch. The polystyrene began folding for the box at 200 sec at 140°C and at 118 sec at 111°C for the latch. Fig. 29 shows the relative temperature of actuation measured using a K-type thermocouple (Fluke 87 V Digital Multi-meter). This graph shows a distinct difference in actuation temperature for the polyolefin and polystyrene shape memory polymers for each shape.The dependencies of the folding transition temperatures are complex. Future work will include characterizing the parameters that affect the transition temperature for self-folding structures, which may include design complexity, gap width, actuator loading, etc.

VII. CONCLUSION AND FUTURE WORK

In this paper we described an end-to-end approach to making self-folding sheets activated by uniform-heat. We introduced a design pipeline which automatically generates folding information for an arbitrary 3D shape, and then compiles this information into fabrication files. We modeled single- and multiple-step self-folding 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 sheet. Finally, we validated this approach experimentally by generating self-folding sheets for the fabrication of seven target shapes with up to 55 faces and up to 2 step folds 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 sheets. Delamination of the SMP layers from the structural layers occurred along the edges of our self-folding sheets 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 sheets. Although the back side of the bunny shape in Fig. 26 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 sheets 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 Mikhail Volkov, John Romanishin, and Laura Meeker for their insightful discussions and technical support for this research. We are grateful for the support for this work provided in part by NSF grants EFRI-1240383 and CCF-1138967.

REFERENCES

[1] K. Miura, "Method of Packaging and Deployment of Large Membranes in Space," The Institute of Space and Astronautical Science, 1985.

[2] S. A. Zirbel, R. J. Lang, M. W. Thomson, D. A. Sigel, P. E. Walkemeyer, B. P. Trease, S. P. Magleby, and L. L. Howell, "Accommodating Thickness in Origami-Based Deployable Arrays 1," *Journal of Mechanical Design*, vol. 135, no. 11, p. 111005, Nov. 2013.

[3] C. D. Onal, R. J. Wood, and D. Rus, "An Origami-Inspired Approach to Worm Robots," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 2, pp. 430–438, 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," *The Journal of Physical Chemistry 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." *ICRA*, pp. 3153–3158, 2010.

[6] A. M. Hoover, E. Steltz, and R. S. Fearing, "RoACH: An autonomous 2.4g crawling hexapod robot." *IROS*, pp. 26–33, 2008.

[7] P. S. Sreetharan, J. P. Whitney, M. D. Strauss, and R. J. Wood, "Monolithic fabrication of millimeter-scale machines," *Journal of Micromechanics and Microengineering*, vol. 22, no. 5, p. 055027, May 2012.

[8] A. T. Baisch, O. Ozcan, B. Goldberg, D. Ithier, and R. J. Wood, "High speed locomotion for a quadrupedal microrobot," *The International Journal of Robotics Research*, vol. 33, no. 8, pp. 1063–1082, Jul. 2014.

[9] F. Haas and R. J. Wootton, "Two Basic Mechanisms in Insect Wing Folding," *Proceedings of the Royal Society B: Biological Sciences*, vol. 263, no. 1377, pp. 1651–1658, Dec. 1996.

[10] T. Eisner, "Leaf folding in a sensitive plant: A defensive thorn-exposure mechanism?" *Proceedings of the National Academy of Sciences*, 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," *Proceedings of the Royal Society B: Biological Sciences*, 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, B. An, N. M. Benbernou, H. Tanaka, S. Kim, E. D. Demaine, D. Rus, and R. J. Wood, "Programmable matter by folding," in *Proceedings of the National Academy of Sciences*, Jun. 2010, pp. 12 441–12 445.

[14] A. Firouzeh, Y. Sun, H. Lee, and J. Paik, "Sensor and actuator integrated low-profile robotic origami," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*. IEEE, pp. 4937–4944.

[15] B. An and D. Rus, "Designing and programming self-folding sheets," *Robotics and Autonomous Systems*, 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 *2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 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 *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 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 *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 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 Letters*, vol. 14, no. 7, pp. 4164–4170, Jul. 2014.

[22] B. An, S. Miyashita, M. T. Tolley, D. M. Aukes, L. Meeker, E. D. Demaine, M. L. Demaine, R. J. Wood, and D. Rus, "An end-to-end approach to making self-folded 3D surface shapes by uniform heating," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1466–1473, 2014.

[23] "A Universal Crease Pattern for Folding Orthogonal Shapes," Tech. Rep., Sep. 2009.

[24] E. D. Demaine and J. O'Rourke, *Geometric Folding Algorithms*, ser. Linkages, Origami, Polyhedra. Cambridge University Press, Aug. 2008.

[25] T. Tachi, "Origamizing Polyhedral Surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 2, pp. 298–311, 2010.

[26] E. D. Demaine, S. L. Devadoss, J. S. B. Mitchell, and J. O'Rourke, "Continuous foldability of polygonal paper." *CCCG*, pp. 64–67, 2004.

[27] E. Demaine, M. Demaine, and J. Ku, "Folding Any Orthogonal Maze," in *Origami 5*. A K Peters/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*. A K Peters/CRC Press, Nov. 2011, pp. 609–626.

[29] C. Sung, E. D. Demaine, M. L. Demaine, and D. Rus, "Edge-Compositions of 3D Surfaces," *Journal of Mechanical Design*, vol. 135, no. 11, p. 111001, Nov. 2013.

[30] M. W. Bern and B. Hayes, "The Complexity of Flat Origami." *SODA*, pp. 175–183, 1996.

[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," *Journal of Microelectromechanical Systems*, 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 *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1232–1237.

[34] S. M. Felton, M. T. Tolley, B. Shin, C. D. Onal, E. D. Demaine, D. Rus, and R. J. Wood, "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 *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*. IEEE, 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 Materials and Structures*, vol. 23, no. 9, p. 094006, 2014.

[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, p. 1764, 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, p. 51085, Dec. 2012.

[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, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 406–420.

[41] Q. Ge, C. K. Dunn, H. J. Qi, and M. L. Dunn, "Active origami by 4D printing," *Smart Materials and Structures*, vol. 23, no. 9, p. 094007, Sep. 2014.

[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," vol. 5, pp. 13 616 EP –.

[43] P. Sitthi-Amorn, J. E. Ramos, Y. Wang, J. Kwan, J. T. Lan, W. Wang, and W. Matusik, "MultiFab: a machine vision assisted platform for multi-material 3D printing." *ACM Trans. Graph. ()*, vol. 34, no. 4, pp. 129–129:11, 2015.

[44] "Origamizing Polyhedral Surfaces," vol. 16, no. 2, pp. 298–311, 2010.

[45] S. Takahashi, H.-Y. Wu, S. H. Saw, C.-C. Lin, and H.-C. Yen, “Optimized Topological Surgery for Unfolding 3D Meshes,” *Computer Graphics Forum*, vol. 30, no. 7, pp. 2077–2086, Nov. 2011.
[46] T. Tachi, “Simulation of Rigid Origami,” in *Origami 4*. A K Peters/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,” *Computational Geometry*, vol. 24, no. 2, pp. 51–62, Feb. 2003.
[48] R. C. Prim, “Shortest connection networks and some generalizations,” *Bell System Technical Journal, The*, vol. 36, no. 6, pp. 1389–1401, 1957.
[49] E. D. Demaine and J. O’Rourke, “Flattening Polyhedra,” in *Geometric Folding Algorithms*. Cambridge: Cambridge University Press, 2010, pp. 279–284.