2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Programming and Controlling Self-Folding Robots

Byoungkwon An*
dran@csail.mit.edu

Daniela Rus*
rus@csail.mit.edu

Abstract— **This paper describes a robot in the form of a self-folding sheet that is capable of origami-style autonomous folding. We describe the hardware device we designed and fabricated. The device is a sheet with a box-pleated pattern and an integrated electronic substrate and actuators. The sheet is programmed and controlled to achieve different shapes using an idea called sticker programming. We describe the sticker controller and its instantiation. We also describe the algorithms for programming and controlling a given sheet to self-fold into a desired shape. Finally we present experiments with a 4×4 hardware device and an 8×8 hardware device.**

I. Introduction

A self-folding sheet is a robotic sheet that autonomously transforms its shape by folding into the users' desired shapes. Our vision is to develop the hardware and software technology that will allow users to make shapes by starting with a self-folding sheet and adding physical stickers to select and trigger a control sequence guaranteed to achieve the desired shape. We imagine sheets capable of folding as a variety of objects, such as a table, an airplane, or a tent. Applications include digital fabrication, on-demand construction of objects in remote environments, on-demand creation of tools, etc. We aim to automate the creation of origami objects.

We have developed a novel device called the self-folding sheet (Fig. 1). This device is a sheet patterned using an $n \times n$ box-pleated pattern (for $n = 4$ and $n = 8$). We associate an SMA actuator with each edge of the sheet and embed supporting electronics. The sheet can be viewed as a modular robot system, where each tile in the system corresponds to a module. The sheet can fold following planning algorithms, such as those described in [3], to achieve a three-dimensional shape. The planner provides the required sequence of origami folds, which can be executed using the actuators embedded on the sheet.

Making three-dimensional shapes by folding has several advantages over achieving shape formation using modular self-reconfiguring robot systems composed of individual independent modules. Since the modules are connected at all times, the self-folding sheet is less prone to the type of connection and disconnection errors that occur in unit-modular systems. The planning system can be computed in a centralized fashion and executed in a highly parallel fashion. The folding operation is relatively easy to control. The challenge, however, is in fabricating a self-folding sheet with embedded actuators and supporting electronics that is capable of physically delivering self-folding actions, especially with multiple folds on the same edge, and the planning algorithm that will synthesize the covert folding sequence.

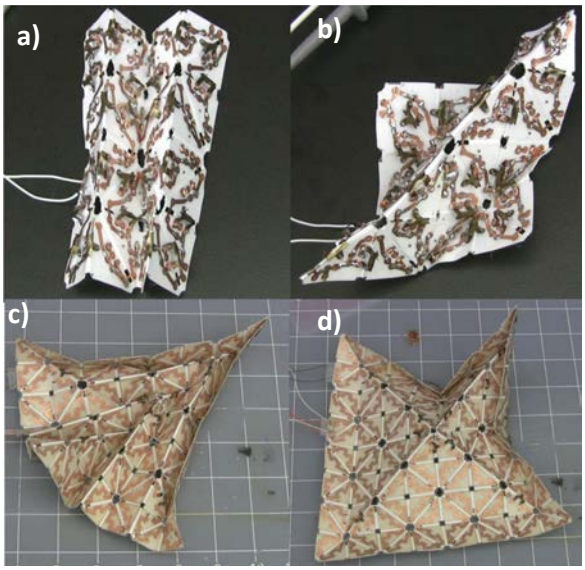

Fig. 1. Two self-folding sheets transform itself into programmed objects. (a) Vertical Folding. (b) Diagonal Folding. (c) Space Shuttle. (d) Hat.

In our prior work we described the self-folding concept [13] and a centralized planner for multi-origami folding from a single sheet [3]. In this paper we present the design and fabrication of a method for specifying the desired shape to be achieved by folding, out of the set of shapes supported by the multi-origami planner and device. By adding physical patches of material to the sheet at shape-dependent critical places (computable by a planner), the the control for the desired shape is triggered and actuates the folding of the shape. We call this approach *sticker control and programming*. In sticker programming, the control sequence is achieved by adding stickers, which are small segments of conductive materials, to key locations on the sheet. The addition of each sticker completes a circuit that triggers the function of an actuator. By adding/removing different stickers at different locations, we select different control sequences for achieving different desired shapes. Given a self-folding sheet and desired shapes, we can compute automatically the number of stickers and their placement on the sheet. We also describe two self-folding devices we have built and used to test sticker programming. We give experimented results collected using a 4×4 self-folding sheet and an 8×8 self-folding sheet (Fig. 1).

Our contributions in this paper are (1) sticker control archi-

*Distributed Robotics Lab, CSAIL, MIT

**Support for this work was provided in part by the DARPA Programmable Matter and NSF EFRI programs (NSF EFRI-0710252). We are grateful for it. We thank Jamie Baik, Rob Wood, Erik Demaine for insightful discussions and feedback on this research.

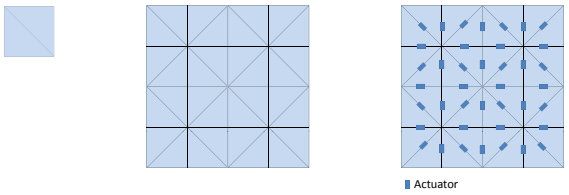

Fig. 2. A 1×1 self-folding sheet (left) is a basic unit of self-folding sheets. A 4×4 self-folding sheet (middle), and with folding actuators (right). The 4×4 sheet is composed of 16 ($= 4 \times 4$) 1×1 self-folding sheets.

tecture, (2) algorithm for the synthesis of sticker placement and (3) devices and experiments with two different self-folding sheet robots.

A. Related Works

1) Self-Reconfigurable System and Algorithm: Most prior research in the modular robots addresses the design of self-reconfiguring systems [1], [15], [14] and related shape-planning algorithms [4], [9], [12].

The self-folding robot is different than modular self-reconfiguring robots in that the modules in a self-reconfiguring system are disconnected, while the self-folding sheet has a mesh of connected tiles, each tile serving the role of a module.

2) Origami Theory: Our theoretical model for the self-folding sheet has a box-pleated pattern (Fig. 2). Demaine and et al. [8] recently proved that an $n \times n$ box-pleated tiling can fold into any polyhedral surface made up of $O(n)$ unit cubes on a cubic lattice. Further, [11] shows that any folded state can be reached by a continuous folding motion without the material penetrating itself.

3) Robotic Origami Folding: Prior work on robotic origami folding considered the design of a robot that folds a sheet into a 3D structure and its supporting algorithms, and thus relies on external actuation for each folding operation. Balkcom and Mason [7], [6], [5] have built a robot that makes a sequence of *simple folds*—folds along a single line at a time. The robot folds a restrictive class of origami models. By contrast, our folds are generally more complicated, involving several simultaneous creases. Many other works considered robots for automatic folding of cartons and packaging [16], [10]. In our work, the actuation of the sheet is internal; the sheet itself is a self-folding robot that transforms itself into the target object.

Nagpal [18], [17] developed and simulated algorithms for folding origami from a sheet of identically programmed autonomous units named "cells." She described a language for instructing the cells to assemble themselves into a global shape. We achieve reconfiguration from a single connected robot, which simplifies manufacturing.

II. Technical Approach

Given k desired 3D objects and a self-folding sheet robot, our goal is to compute and program the control sequence required to fold the 3D object from the sheet. The planning algorithm has been described in [3]. Here we discuss how to go from the theoretical plan to an executable sequence by exploring a new programming and controlling method.

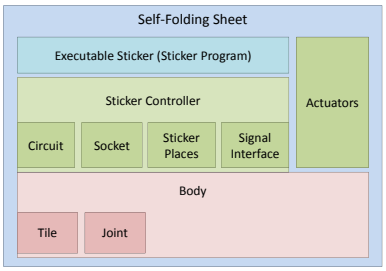

Fig. 3. Sticker control architecture of self-folding sheets

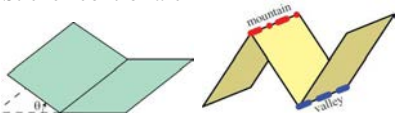

Fig. 4. (left) The fold angle at a crease is the supplement of the dihedral angle. (right) A crease can be folded as either a mountain fold or a valley fold. [3]

To facilitate programming, we design self-folding sheets that support multiple target 3D shapes using the layered sticker control architecture in Fig. 3. The bottom layer represents the box-pleated body of the self-folding sheet. The next layer represents the electronic infrastructure and the actuation system associated with each fold. The actuators are not connected in this layer. The geometry of the electronic substrate is shown in Fig. 5. The third layer is the *programming* layer. It represents the sticker programming configuration. Physical stickers are added to the sheet at computable locations, to connect and trigger the action of an actuator (Fig. 15). The stickers complete a circuit that triggers the folding execution sequence to achieve the desired 3D shape. A sticker set can be removed and replaced by a different sticker set to fold a different 3D object. Section IV describes the algorithm for automatically generating the sticker locations for a fixed self-folding sheet and a desired object.

III. Architecture for the Self-Folding Sheet

A self-folding sheet is a box-pleated 2-dimensional sheet designed to transform itself into the desired shapes by folding selected edges. Figure 2 shows the simplified structure of the 4×4 self-folding sheet. The kinematic components of the sheet include tiles, joints (hinges), and actuators. The controlling components include a sticker controller and sticker programs. The tiles locally rotate (fold) around their joints. The dihedral angle is in the range 0° to $\pm 180^\circ$.

The fold angle is the supplement of the dihedral angle between the two face meeting at the hinge (Fig. 4). The sign of the fold angle determines the crease as either a mountain fold or a valley fold (Fig. 4).

The edges of the sheet are actuated using SMA actuators embedded in the sheet. When the actuators are added to the sheet, they are not connected to the electronic substrate. The connection is done selectively, according to the goal shape, by adding stickers.

A. Model of Sticker Controller

The sticker controller is a module that contains the electronic substrate required to fold the self-folding sheet into users' desired shapes, when the users provide sticker

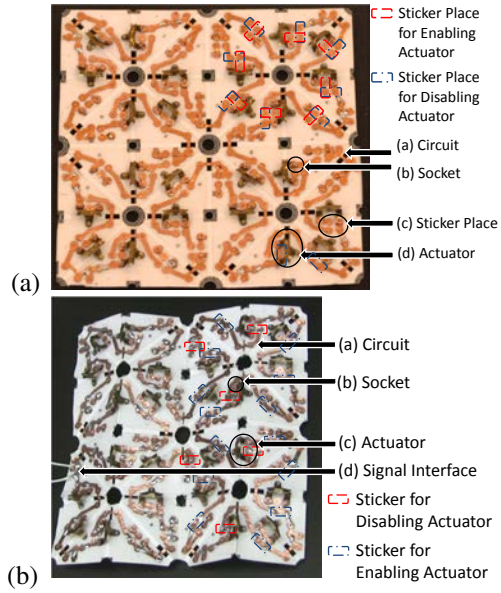

Fig. 5. 4×4 sticker controller (a) with no sticker (with no program) and (b) with a sticker set (vertical folding program).

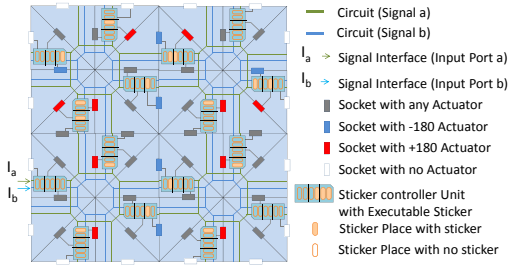

Fig. 6. Model of 4×4 sticker controller.

programs to the controller. It provides the user with a programming interface which is implemented using physical materials.

Figure 5(a) shows the 4×4 self-folding sheet including a sticker controller with no sticker. Figure 5(b) shows the sticker controller after we input the program for vertical folding to achieve the shape in Fig. 1(a). Figure 6 shows a model for the sticker controller. The sticker controller is composed of a signal interface, a circuit, (actuator) sockets, and sticker controller units (sticker places) [2].

B. Sticker Controller Unit

The *sticker controller unit* is a group of sticker places for each $1\times$ 1 sticker controller (Fig. 5, 6). The unit has *sticker places*, *input ports*, and *output ports*. The sticker places are locations within the controller substrate for the program. The input ports are connected to the circuit. The output ports are connected to three actuator sockets on left, diagonal, and bottom edges. When the input ports of the unit receive energy, the unit passes the signal to the selected outputs.

A *sticker controller unit* is named by the *n-i-o* sticker controller unit, where:

- n is the number of output port groups,
- i is the number of input ports, and
- o is the number of output ports for each output port group.

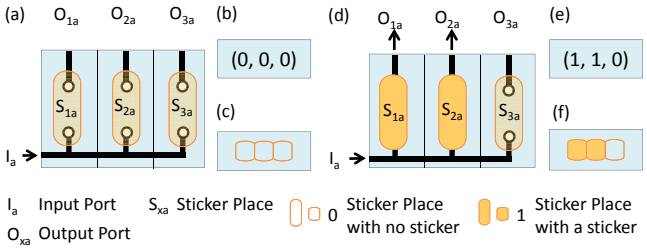

Fig. 7. (a)(b)(c) represent the same 3-1-1 sticker controller unit with no sticker. (d)(e)(f) represent the same 3-1-1 sticker controller unit with a sticker set. We draw two sticker controller units in three different diagrams. The model diagrams(a)(d) show detailed information of the sticker controller unit. The same information can be abstracted to a set of actuator codes displayed by the code diagrams(b)(e). It is also depicted as a graphical image in the sticker diagrams(c)(f).

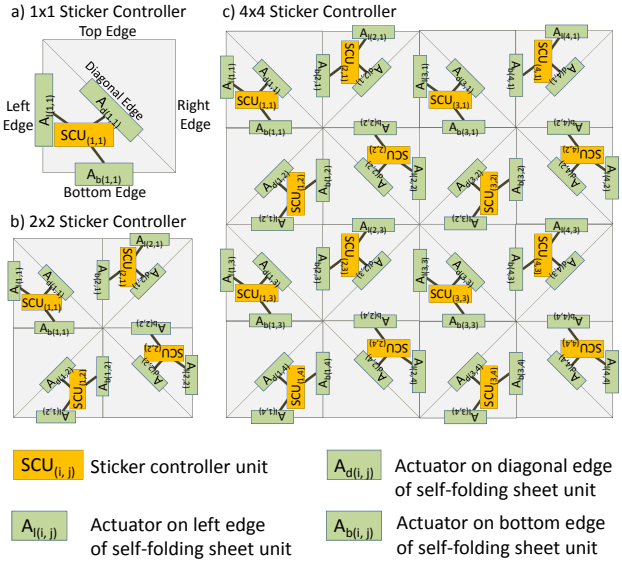

Fig. 8. Simplified model for the 1×1, 2×2 and 4×4 sticker controllers. 1×1 self-folding sheet is composed of one sticker controller unit and three actuator sockets on left, diagonal, and bottom edges (a). A 2×2 self-folding sheet is composed of four 1×1 sticker controllers (b). A 4×4 self-folding sheet is composed of four 2×2 sticker controllers (c).

Figure 7 shows the 3-1-1 sticker controller unit. Each 1×1 self-folding sheet module has three actuators: left, diagonal, and bottom actuators, as shown in Figure 8 (c). We select the outputs by adding conductive material to the selected sticker places. Figure 7 shows two 3-1-1 sticker controller units with no sticker and with a sticker set. In Figure 7 (d), when input port I_a gets energy, O_{1a} and O_{2a} receive the energy; O_{1a} and O_{2a} are connected to S_{1a} and S_{2a}. This causes the actuators connected to O_{1a} and O_{2a} to be activated. The input voltage of I_a and the output voltage of O_{1a} and O_{2a} are the same.

IV. Sticker Programming Algorithm

Given a self-folding sheet and k target shapes, the sticker programming algorithm generates a sticker program; a sticker program is composed of a *sticker place design* and a *sticker command script*. The sticker design contains the places and the shapes of the stickers. The sticker script contains the k sequences of the triggering signals to transform the programmed self-folding sheet into the k shapes.

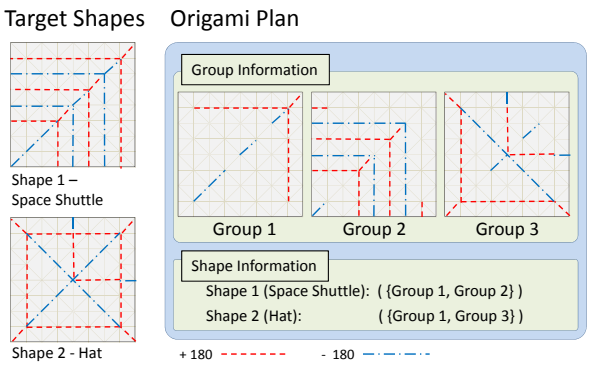

Fig. 9. (left) Input target shapes for the 8×8 sheet. (right) Origami plan from origami planner.

The sticker programming algorithm is composed of three components: an origami planner, a sticker compiler, and a sticker linker.

A. Origami Planner

Given multiple 3D target shapes, the origami planner generates an optimized origami plan. For each shape, the planner determines the sequence of folds required to achieve the shape. Details about the origami planner are presented in [3]. Figure 9 shows an example of an origami plan for space shuttle and hat shapes. According to Shape Information, Group 1 $\cup$ 2 folds the space shuttle and Group 1 $\cup$ 3 folds the hat.

B. Sticker Compiler

Given the group information of an origami plan and an actuator model for the self-folding sheet, the sticker compiler generates an executable sticker object and a sticker command script. The actuator model is a function $f(A) \rightarrow B$, where A is a fold angle and B is a binary signal (actuator code) for the fold angle A. Figure 10 shows the five step process overview.

Sticker Compiler

1) Given the group information of an origami plan and an actuator model of a fixed self-folding sheet, convert each actuation angle to its corresponding actuator codes.
2) Combine the actuator codes of all groups of each edge.
3) Construct an executable sticker object by collecting the combined actuator codes on the edges (Fig. 11).
4) Construct a sticker command script by converting the folding sequence of each shape (the shape information of the origami plan (i.e. Fig. 9)) into a sequence of commends (i.e. Fig. 13).
5) Output the executable sticker object and the sticker command script.

Fig. 10. Algorithmic overview of sticker compiler.

Constructing Executable Sticker Object (Step 3)

- Given the converted actuator codes of each edge (From step 2).
- Let $ac_{(i,j)-(k,l)}$ be a converted actuator code on an edge $e = (v_{(i,j)}, v_{(k,l)})$, where v is a vertex. $ac_{(i,j)-(k,l)} \leftarrow f($ fold angle of $e)$.
- Let $SO = \{sou_{i,j} \mid i,j \leq n\}$ be an executable sticker object.
- Let $sou_{i,j}$ be a executable sticker object unit of SO.
- For each column i and row j:
  1) If i = odd and j = odd, $sou_{i,j} \leftarrow (ac_{(i,j)-(i,j+1)}, ac_{(i,j)-(i+1,j+1)}, ac_{(i,j+1)-(i+1,j+1)})$.
  2) If i = even and j = odd, $sou_{i,j} \leftarrow (ac_{(i,j)-(i+1,j)}, ac_{(i+1,j)-(i,j+1)}, ac_{(i,j)-(i,j+1)})$.
  3) If i = odd and j = even, $sou_{i,j} \leftarrow (ac_{(i,j+1)-(i+1,j+1)}, ac_{(i+1,j)-(i,j+1)}, ac_{(i+1,j)-(i+1,j+1)})$.
  4) If i = even and j = even, $sou_{i,j} \leftarrow (ac_{(i+1,j)-(i+,j+1)}, ac_{(i,j)-(i+1,j+1)}, ac_{(i,j+1)-(i+1,j)})$.

Fig. 11. Details of sticker compiler step 3.

Sticker Linker

1) For each alphabet (bit) of a given executable sticker object,
   a) Construct the *id* of the alphabet.
   b) Given a sticker place set, find a sticker place of *id*.
   c) Copy the polygon p to the sticker place design.
2) Output the sticker place design.

Fig. 12. Algorithmic overview of sticker linker.

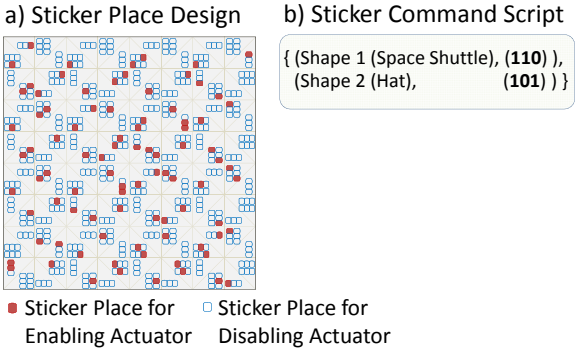

Fig. 13. Result of sticker programming algorithm for the 8×8 sheet. The space shuttle and the hat are input target shapes (Fig. 9 (left)). The sticker compiler and the sticker linker generate (a). The sticker compiler generates (b). After we input (a) to the the 8×8 sheet, signal (110) of (b) transforms the sheet into the space shuttle and signal (101) of (b) transforms the sheet into the hat.

1) Generating the Executable Sticker Object: The first step (Step 1 in Figure 10) is to convert all angles from the planner to their corresponding actuator codes (Sec. III-B).

The second step is to combine all actuator codes of each edge into the combined actuator codes for the sheet.

The third step is to construct an executable sticker object by collecting the combined actuator codes of the edges. The details of the step are in Figure 11.

2) Generating Sticker Command Script: In the fourth step, the sticker compiler converts shape information of an origami plan into a sticker command script by replacing the group names to binary codes, which are signals for the signal interface (Fig. 13 (b)).

C. Sticker Linker

Given an executable sticker object and a sticker place set, the sticker linker generates a sticker place design. Figure 12 shows an overview of the sticker linker algorithm.

Each alphabet (each bit of the actuator code) of an executable sticker object is represented by *id*, where *id* = (a socket-id *sid*, a position l, an alphabet a) and *sid* = $\{A_{l(i,j)}, A_{d(i,j)}, A_{b(i,j)}\}$. For example, $A_{l(3,4)}$, a *sid*, represents the left actuator on column 3 and row 4 of the self-folding sheet. A sticker place set is a finite set of sticker places (id, p) (id and polygon pairs). The polygon is graphical information of a patch of the sticker; a polygon p is $(c, (p_1, p_2, p_3, ..., p_k))$, where c is a color and p_i is a coordinate (x, y) of a point.

The sticker linker generates a sticker place design by copying polygons ps. For each *id* of the given executable sticker object, the linker finds (id, p) of the sticker place design.

TABLE I

OVERVIEW OF 4×4 AND 8×8 SELF-FOLDING SHEETS

	4×4 sheet	8×8 sheet
Crease Pattern	4×4 Box-Pleated	8×8 Box-Pleated
Size	$96mm \times 96mm$	$192mm \times 192mm$
# of Edges	40	176
# of Actuators	40	36
Max. Power	360 W	360 W
Current Setting	1.5 A	5.0 A
Ave. Folding Time	21.6 s	5.0 s
Sticker Controller	Sticker Controller	Socket Controller
Reprogram	Very Easy	Easy
Body	LP Body	LP Body
Actuator	Y-type Actuator	Y-type Actuator
Programming	By Sticker Programming Algorithm	By Sticker Programming Algorithm

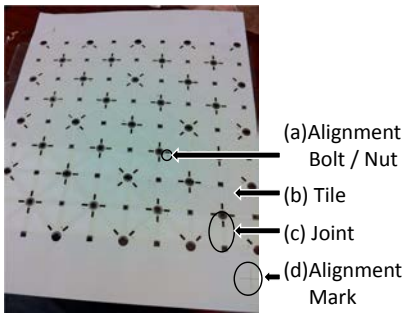

Fig. 14. Lamination tile and paper joint body (LP Body)

Figure 13 shows output of sticker programming algorithm for the space shuttle and hat shapes; the input target shapes are shown in Figure 9 (left).

V. PHYSICAL SELF-FOLDING DEVICES

We have built several 4×4 and 8×8 self-folding sheets; Table I shows the overview of the target sheets. The 4×4 sheet was used to evaluate the low-level self-folding control using straight-line folding and diagonal folding as target shapes. The 8×8 sheet was used to evaluate the self-folding of two complex shapes: a space shuttle and a hat.

A. The Body

The mechanical structure of the self-folding sheets is composed of tiles and joints with the box-pleated crease pattern (Fig. 14). Both devices are built from *Lamination tile and paper joint body (LP body)*.

The body is made of paper sandwiched between lamination film, and micro bolts and nuts (Scale Hardware, 0.5mm). We use paper to form joints and the lamination sheet as tiles. We cut each material with the Versalaser Cutting System. To attach the three layers, we stack them and put it into a laminator. To align the layers, we used the micro bolts and nuts (Fig. 14).

B. Actuators

Both self-folding sheets use the Y-shape SMA actuators (Fig. 15), which are developed by modifying Z-type actuators in [19].

When current passes to the actuators, the actuators are heated and they transform into the annealed shape. This motion generates the folding force.

C. Sticker Controller for 4×4 Self-Folding Sheet

The sticker controller is composed of a circuit, sockets, sticker places, and a signal interface (Fig. 5).

1) The Circuit: The circuit (a) is a network that passes the energy for control. All parts of the sticker controller are on the serial circuit (Fig. 17(right)). The ends (+ and -) of the serial circuit are marked on the figure. The circuit is a symmetric pattern composed of right triangles (Fig. 17). The circuit is scalable with the circuit scaling algorithm depicted in Figure 16.

Copper tape is the material used for the circuit. The copper tape is also used for the socket, and the sticker place. We cut copper tape with DPSS Laser Micromachining System (Custom Build, at the Micro Robotics Lab, Harvard Univ.).

2) The Socket: The sockets connect the circuit and the actuators (Fig 5). *Tail knot sockets* are used for the sheet controller (Fig. 15).

We used the 0.5 mm micro bolts and nuts to attach an actuator. When we attach the actuator, we make a knot on the bolt for better electronic connection.

3) The Sticker Place: The executable sticker for the sticker controller of the 4×4 sheet is composed of $2mm \times 5.6mm$ of copper tape. When placing a sticker at a place for enabling the actuator (Fig. 15), current passes and activates the actuator. When placing a sticker at a place for disabling the actuator (Fig. 15), current does not pass to the actuator.

Each edge has a sticker place. The sticker place has enable and disable actuator areas (Fig. 15). We can add or remove stickers in different combinations. Each set of actuators is triggered by a fixed set of stickers. By replacing the stickers, we can reprogram the sticker controller.

4) The Signal Interface: The sticker controller receives runtime signals through a signal interface (Sec. IV). Because

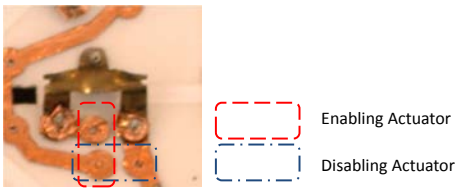

Fig. 15. Y-type actuator and sticker place of 4×4 self-folding sheet.

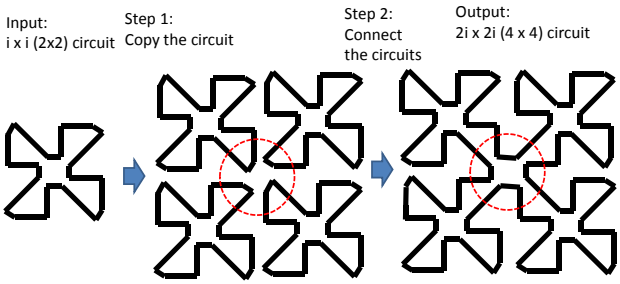

Fig. 16. The circuit scaling algorithm

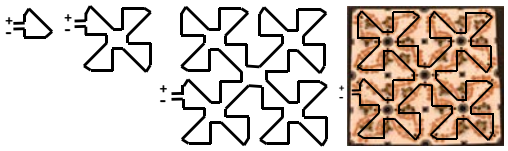

Fig. 17. The circuits for 1×1, 2×2, and 4×4 self-folding sheets

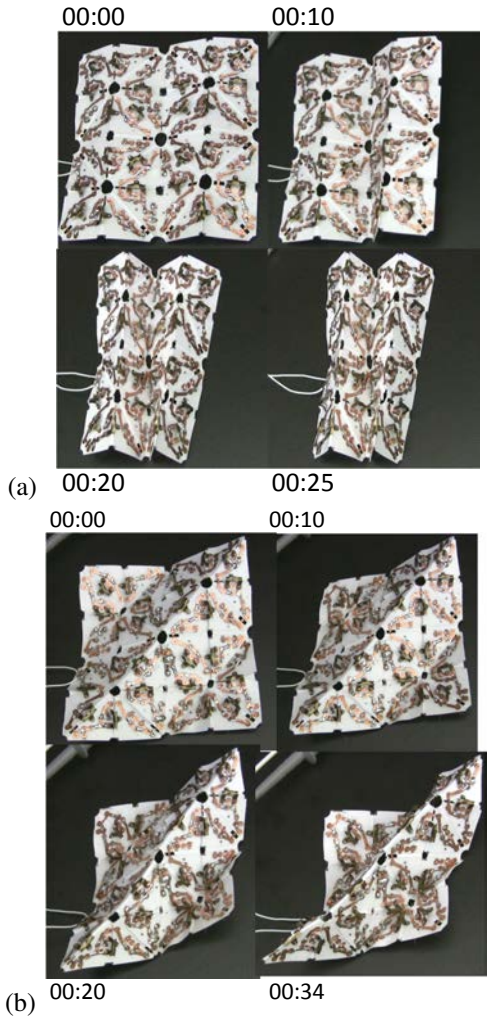

Fig. 18. Snapshots from programming and controlling 4×4 self-folding sheet for diagonal folding (a) and vertical folding (b).

the sticker controller of the 4×4 sheet controls one actuator group, we have one input (one + and one ground) interface.

5) The Executable Sticker: The executable sticker for this controller is composed of the $2mm \times 5.6mm$ patches of copper tape material. We manually placed the stickers on the device, according to the sticker place design that the sticker programming algorithm generates (Sec. IV).

VI. Experiment with the 4×4 Self-Folding Sheet

The 4×4 self-folding sheet runs two basic motions: vertical and diagonal folding (Fig. 18). The sheet contains 3-1-1 sticker controller units. A sticker program for the sheet normally contains one shape.

We implemented and evaluated the following four steps: (1) we generated two sticker place designs for the vertical shape and diagonal shape; (2) we placed and executed the executable sticker for the vertical folding; (3) we removed the executable sticker; (4) we placed and executed the executable sticker for the diagonal folding.

A. Sticker Programming for Vertical and Diagonal Folding

We generated two sticker place designs for the two basic shapes with the sticker programming algorithm (Sec. IV). Table II shows the planning times on hardware.

TABLE II
Origami Planning Time for Vertical and Diagonal Folding

Analysis Time for Vertical	3.6s (3600ms)
Building Time for Vertical	17ms
Analysis Time for Diagonal	4.2s (4200ms)
Building Time for Diagonal	16ms
CPU	Intel Core 2 Quad 2.83GHz (Q9550)
Storage	3 GB RAM, Seagate 750GB 300MBps 7200rpm HDD
Graphics	NVIDIA Quadro FX 1700

B. Results

The 4×4 sheet has 40 actuators and 40 edges. 42.5% of the actuators were used for each of the two shapes (Table III).

TABLE III
Actuators (Ac.) of 4×4 Sheet

	Folding Ac.	Total Ac.	Total Edges	Folding Ac. / Total Ac.	Total Ac. / Total Edges
Vertical	12	40	40	30.0%	100.0%
Diagonal	10	40	40	25.0%	100.0%
Total	11	40	40	42.5%	100.0%

First, we executed the vertical folding program on the 4×4 self-folding sheet 14 times. Second, we removed the program and reprogrammed the sheet diagonal folding program on the sheet. Then we executed the diagonal folding 13 times. The 4×4 sheet achieved the vertical and diagonal folding reliably (Fig. 18). The 4×4 self-folding sheet runs with current set at 1.5A. The average folding time of both shapes is 21.6s (Table IV).

TABLE IV
Folding Time of 4×4 Sheet

	# of Runs	Current Setting	Ave. Folding Time
Vertical	14	1.5 A	21.0 s $\pm$26.7%
Diagonal	13	1.5 A	22.4 s $\pm$17.9%
Total	27	1.5 A	21.6 s $\pm$22.5%

We measured the folding angle for several snapshots and found that the basic folding motion is $134.0^\circ \pm 12.1\%$ (Table V). Our target folding angle for the basic folding motion was 180.0°. We achieved 74.5% of the target angle.

TABLE V
Folding Angle and Folding Achievement of 4×4 Sheet

	Ave. Folding Angles	Target Angles	Folding Achievement (Folding Angle / Target Angle)
Vertical	$141.6^\circ \pm 7.9\%$	180.0°	78.7%
Diagonal	$126.4^\circ \pm 16.3\%$	180.0°	70.2%
Total	$134.0^\circ \pm 12.1\%$	180.0°	74.5%

While we folded the 4×4 sheet 27 times, the experiment failed to meet the goal three times (Table VI). Most of failures were due to broken or weak connections between the socket and the actuator (SMA is hard to solder.) We made the electronic connection not only with solder but also with conductive bolts and nuts. However, while the sheet folded several times, the electronic connection was weak. Once the connection was loose, the socket was hard to recover. In this case, we fixed the system by disabling the broken actuators. The average number of disabled actuators was 1.04 (Table VII). The sheet achieved its goal shapes reliably despite the number of the disabled actuators.

Most of the results of the two basic shapes on the 4×4 sheet are similar. However, the resistance was 19.1 Ω for vertical folding while the resistance was 28.9 Ω for diagonal folding. The resistance of the sheet increased 1.5 times

TABLE VI

FAILURE OF 4×4 SHEET

	# of Runs	# of Failure	Ave. Failure
Vertical	14	1 (of 14 runs)	0.7 (of 10 runs)
Diagonal	13	2 (of 13 runs)	1.5 (of 10 runs)
Total	27	3 (of 27 runs)	1.1 (of 10 runs)

TABLE VII

DISABLED ACTUATORS (AC.) OF 4×4 SHEET

	Ave. # of Disabled Ac.	Folding Ac.	Disabled Ac. / Folding Ac.	Disabled Ac. / Total Ac.
Vertical	0.77	12	6.4 %	1.9 %
Diagonal	1.36	10	13.6 %	3.4 %
Total	1.04	11	9.7 %	2.6 %

TABLE VIII

RESISTANCE OF 4×4 SHEET

	Ave. Resistance
Vertical	19.1 Ω
Diagonal	28.9 Ω
Total	23.6 Ω

after we reprogrammed the sheet. (Table VIII). Because the number of folding actuators is almost same in the two experiments, we can say the connectivity decreases after reprogramming the sheet.

VII. EXPERIMENT WITH THE 8×8 SELF-FOLDING SHEET

We designed the 8×8 sheet to test self-folding planning for more complex shapes. We selected a space shuttle-like shape and a hat-like shap1e (Fig. 19). The sheet contains 3-3-1 sticker controller units. A sticker program for the sheet normally contains two shapes.

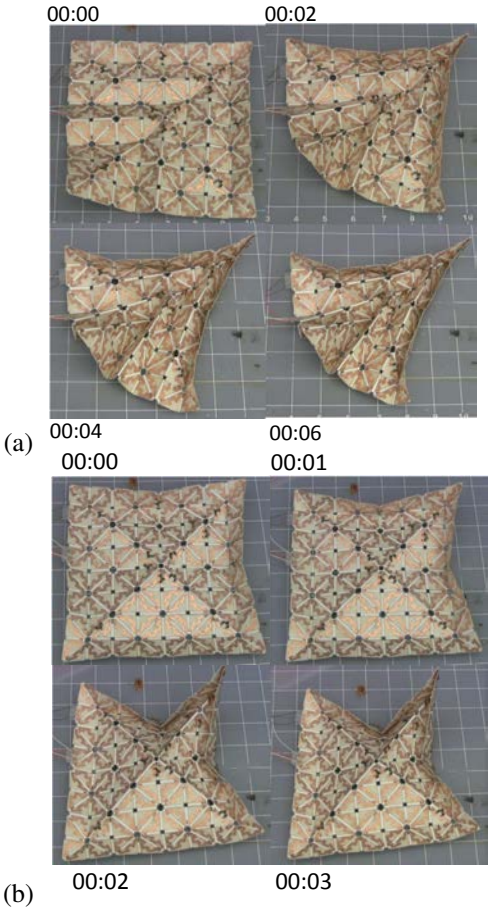

Fig. 19. Snapshots from programming and controlling 8×8 self-folding sheet for space shuttle (a) and hat (b).

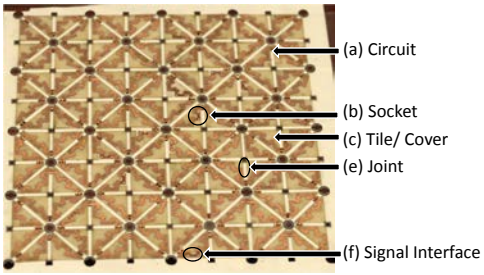

Fig. 20. 8×8 Self-folding sheet after opening the coverlay (electrical insulating film) of the sockets for programming with actuators (before inserting actuators). The area of the coverlay for the socket has a cut line, we can easily take out that area. An actuator will be inserted in the socket (b).

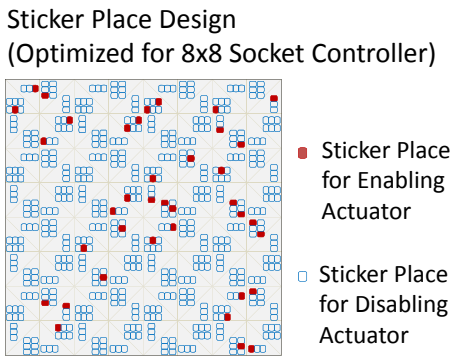

Fig. 21. Sticker place design optimized for the socket controller of the 8×8 sheet. Input target shapes are the space shuttle and the hat (Fig. 9 (left)). Fig. 13 (b) shows the sticker command script for this design. Fig. 13 (a) shows the sticker place design before optimizing.

We implemented and evaluated the following five steps: (1) we generated a sticker place design for the space shuttle and hat shapes; (2) we optimized the sticker place design for the socket type sticker controller; (3) we placed the executable sticker for the two shapes; (4) we executed the executable sticker for the space shuttle shape; (5) we executed the executable sticker for the hat shape.

The 8×8 sheet includes a socket controller (Table I, Fig. 20) that is a type of sticker controllers having hybrid-socket sticker places. Instead of adding or removing the stickers, we input the program to the sheet by inserting or ejecting the actuators.

A. Sticker Programming for Folding of Space Shuttle and Hat Shapes

TABLE IX

MULTIPLE ORIGAMI PLANNING TIME

Analysis Time for Space Shuttle	5.3s (5300ms)
Building Time for Space Shuttle	19ms
Analysis Time for Hat	4.9s (4900ms)
Building Time for Hat	17ms
Building Time for Multiple Origami Plan	25ms
Total Time	10.0s (10261ms)

We generated the sticker place design for the two shapes with the sticker programming algorithm (Sec. IV). The origami planner automatically planned the folding of the two target shapes. (Fig. 9) Using the compiling and linking algorithms, We manually computed the sticker place design and the sticker command script (Fig. 13). Then, we manually optimized the sticker place design for the socket controller (Fig. 21). Table IX shows the planning times.

B. Results

The socket controller controlled the 8×8 sheet with the relatively small number of actuators (only 20.5% of edges have the actuators) (Tables X, I). We populated only the 36 edges relevant to our self-folding target shapes out of the 176 edges of the 8×8 sheet with actuators, while we populated the 40 edges of the 4×4 sheet with 40 actuators. The 8×8 self-folding sheet has 18.2% less actuators then 4×4 self-folding sheet, while the 8×8 sheets has 4.4 times more edges than the edges of the 4×4 sheet (Tables III, I). 61.1% of the actuators are used, when the 8×8 sheet transformed into the both shapes (Table X).

TABLE X

ACTUATORS (AC.) OF 8×8 SHEET

	Folding Ac.	Total Ac.	Total Edges	Folding Ac. / Total Ac.	Total Ac. / Total Edges
S. Shuttle	20	36	176	55.6%	20.5%
Hat	24	36	176	66.7%	20.5%
Total	22	36	176	61.1%	20.5%

We executed the space shuttle shape folding on the 8×8 device 14 times. Then, we executed the folding of hat shape 12 times. The 8×8 sheet achieved the space shuttle and hat shapes reliably with the optimized number of actuators (Fig. 19). The 8×8 self-folding sheet ran with current set at 5.0A. The average folding time was 5.0s (Table XI).

TABLE XI

FOLDING TIME OF 8×8 SHEET

	# of Runs	Current Setting	Ave. Folding Time
Space Shuttle	14	5.0 A	5.9 s $\pm$16.9%
Hat	12	5.0 A	4.5 s $\pm$23.4%
Total	26	5.0 A	5.0 s $\pm$19.9%

While we folded the 8×8 sheet 26 times with the two complex shapes, the experiment failed five times (Table XII). Like the 4×4 sheet, most of the failures were due to broken or weak connections between a socket and an actuator. We resolved these failures by disabling the broken actuators.

TABLE XII

FAILURE OF 8×8 SHEET

	# of Runs	# of Failure	Ave. Failure
Space Shuttle	14	3 (of 14 runs)	2.1 (of 10 runs)
Hat	12	2 (of 12 runs)	1.6 (of 10 runs)
Total	26	5 (of 26 runs)	1.9 (of 10 runs)

The average number of disabled actuators (for fix) was 0.81. It is 3.7 % of the folding actuators and 2.2 % of the total actuators (Table XIII). The sheet achieved their shapes reliably with this number of the disabled actuators.

TABLE XIII

DISABLED ACTUATORS (AC.) OF 8×8 SHEET

	Ave. # of Disabled Ac.	Folding Ac.	Disabled Ac. / Folding Ac.	Disabled Ac. / Total Ac.
S. Shuttle	0.82	20	4.1 %	2.3 %
Hat	0.80	24	3.3 %	2.2 %
Total	0.81	22	3.7 %	2.2 %

We enabled the actuator group 1 and 2 for the space shuttle-like shape. The resistance for the space shuttle shape was $17.4k\Omega$. We enabled the actuator group 1 and 3 for the hat-like shape. The resistance for the hat shape was 80.15Ω. While we executed the space shuttle shape, the average resistance of the group 3 was $1.71M\Omega$. However, because we did not use the group 3 for the space shuttle shape, there was no problem to achieve the shape.

TABLE XIV

RESISTANCE OF 8×8 SHEET

	Ave. Resistance Group1, Group2, Group3	Ave. Resistance of Folding Groups
Space Shuttle	44.8Ω, $34.8k\Omega$, $1.71M\Omega$	$17.4k\Omega$ (Group1 + Group2) / 2
Hat	109.3Ω, $14.7k\Omega$, 51.0Ω	80.15Ω (Group1 + Group3) / 2

VIII. CONCLUSIONS

We developed two different hardware devices and conducted experiments with the sticker placement and self-folding control algorithms, which can be used as an automatic programming method for robots with printer-like programming machines. We achieved four target shapes reliably. Finally, we collected and analyzed self-folding data during these experiments. The devices were programmed using the concept of sticker programming and the sticker placement algorithm introduced in this work. The results in this paper enable a path from theoretical origami folding algorithms to experimental self-folding sheet robots capable of simple autonomous 3D shape formation.

REFERENCES

[1] B. An. Em-cube: Cube-shaped, self-reconfigurable robots sliding on structure surfaces. In *ICRA*, 2008.

[2] B. An. Sticker controller and sticker programming for smart sheets (self-folding sheets). Master's thesis, CSAIL, MIT, Cambridge, MA, September 2011.

[3] B. An, N. Benbernou, E. Demaine, and D. Rus. Planning to fold multiple objects from a single self-folding sheet. *Robotica*, 2011.

[4] B. An and D. Rus. Making shapes from modules by magnification. In *IROS*, 2010.

[5] D. Balkcom. *Robotic Origami Folding*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2004.

[6] D. Balkcom and M. Mason. Introducing robotic origami folding. In *ICRA*, 2004.

[7] D. Balkcom and M. Mason. Robotic origami folding. *IJRR*, 27(5):613 – 627, May 2008.

[8] N. Benbernou, E. Demaine, M. Demaine, and A. Ovadya. A universal crease pattern for folding orthogonal shapes. arXiv:0909.5388, September 2009.

[9] Z. Butler, K. Kotay, D. Rus, and K. Tomita. Generic decentralized control for lattice-based self-reconfigurable robots. *IJRR*, 23(9):919–937, 2004.

[10] J. Dai and D. Caldwell. Origami-based robotic paper-and-board packaging for food industry. *Trends in Food Science & Technology*, 21(3):153–157, March 2010.

[11] E. Demaine, S. Devadoss, J. Mitchell, and J. O'Rourke. Continuous foldability of polygonal paper. In *Proceedings of the 16th Canadian Conference on Computational Geometry (CCCG'04)*, 2004.

[12] K. Gilpin, K. Kotay, D. Rus, and I. Vasilescu. Miche: Modular shape formation by self-disassembly. *IJRR*, 27(3-4):345–372, 2008.

[13] E. Hawkes, B. An, N. Benbernou, H. Tanaka, S. Kim, E. Demaine, D. Rus, and R. Wood. Programmable matter by folding. *Proceedings of the National Academy of Sciences*, 107(28):12441–12445, 2010.

[14] K. Kotay and D. Rus. Algorithms for self-reconfiguring molecule motion planning. In *IROS*, 2000.

[15] K. Kotay and D. Rus. Locomotion versatility through self-reconfiguration. *In Robots and Autonomous Systems*, 2000.

[16] L. Lu and S. Akella. Folding cartons with fixtures: A motion planning approach. *IEEE Transactions on Robotics and Automation*, 16(4):346–356, 2000.

[17] R. Nagpal. *Programmable self-assembly: constructing global shape using biologically-inspired local interactions and origami mathematics*. PhD thesis, Massachusetts Institute of Technology, 2001.

[18] R. Nagpal. Self-assembling global shape, using ideas from biology and origami. In Thomas Hull, editor, *Origami^3: Third International Meeting of Origami Science, Math and Education*. A K Peters, 2002.

[19] J. Paik, E. Hawkes, and R. Wood. A novel low-profile shape memory alloy torsional actuator. *Smart Materials and Structures*, 19(12):125014, 2010.