A TOOLCHAIN FOR THE DESIGN AND SIMULATION OF FOLDABLE PROGRAMMABLE MATTER

Steven R. Gray, Jungwon Seo, Paul J. White, Nathan J. Zeichner, Mark Yim, Vijay Kumar
General Robotics, Automation, Sensing, and Perception (GRASP) Laboratory
University of Pennsylvania
Philadelphia, PA 19104
Email: {stgray, juse, whitepj, nathanz, yim, kumar}@seas.upenn.edu

ABSTRACT
This paper presents a toolchain for the design and simulation of reconfigurable robots that can be built from a single rigid sheet of smart material with embedded actuators and sensors along regular crease patterns. We call such sheets Foldable Programmable Matter (FPM). The toolchain we have created comprises an editor for drafting or modifying FPM, including locations and angles for folds. Algorithms for generating a class of folding structures are available for use. Also included is a dynamic simulation of the fold process, which provides collision detection and visualization. Thus our Foldable Programmable Matter Editor allows us to synthesize and design FPMs and simulate them in a virtual environment before committing to manufacture. The toolchain also incorporates a method of strength analysis, which is used to determine the suitability of a folded shape for specific loadings. Examples are shown for each subsection, including a beam example spanning the toolchain.

1 INTRODUCTION
We present a toolchain enabling the design of systems in which a single rigid sheet of creased material with embedded actuators reconfigures into multiple target forms. Recent work has demonstrated the feasibility of embedding actuators in sheets [1] and folding elements to form larger structures [2, 3]. The sheets are referred to as Foldable Programmable Matter (FPM). The results of this paper enable the design, planning, and analysis of such actuated, rigid sheets of smart materials. Figure 1 illustrates the role of the toolchain in designing a physical device.

The design of folded structures is closely related to the algorithmics of origami [5, 6]. Although robots have been used to fold origami [7], we are interested in the problem in which the folded structure is the robot. Traditional origami involves developable surfaces; paper can be deformed by bending, but not by stretching. In contrast, FPM crease patterns represent rigid sheets connected by revolute joints. As a cursory study of origami would suggest, foldable sheet structures are able to approximate arbitrary shapes ranging from animals to boxes, airplanes, and boats. The limiting factor in the design is the size of the sheet and the minimum spacing between creases; it has been proven that every polyhedral surface can be folded given a large enough sheet [8]. Autonomously shape changing FPM can be viewed as a versatile multi-tool, endlessly reconfiguring into the right tool at the right time. What was once a sheet could become a hammer, a screwdriver, or a wrench as needed. Alternatively, consider a FPM robot designed to walk across rough terrain. Upon encountering a cliff overlooking a lake, the robot could take the shape of an airplane to glide down and reconfigure into a boat to cross the water.

A sought-after goal of programmable matter has been to create structures with specified mechanical properties [9]. As the building blocks (modules) decrease in size, the resolution of the shapes created increases, as does the ability to fine-tune the mechanical properties of the structure. The tradeoff for decreased module size is an increase the number of modules needed to fill
Apart from folding, approaches to programmable matter include Modular Self-reconfigurable Robots (MSRs) and Self-assembling Structures. These approaches typically utilize 3D modules with nice space-filling properties [2, 10], such as cubes or right-angle tetrahedrons. Modular robotic systems, with single modules typically in the centimeter scale, allow for control over each module as needed to make a snake-type crawler, four-legged walker [11], self-reconfiguring structures [12, 13], stochastically formed shapes [14,15], etc. However, large module size and limited numbers of modules prevent a high resolution solution.

Folding is a promising actuation method for programmable matter that is applicable to systems in the micro or nano-scale [16]. Much recent progress has been made to bring folded or reconfigurable devices into existence. Folding approaches have led to centimeter-scale walkers [17]. Developments in reversible permanent magnets [18], nano-adhesives [19] and novel microactuation [1] have enabled new designs and improved function.

The work we present here provides tools to aid in the design of FPM structures, from initial concept to fold sequences to testing mechanical properties. Section 2 presents information on the structure of FPM. Section 3 presents known motion planning methods used for FPM, the results of which can be implemented using our editor described in Sec. 4. The strength of the resulting folded structure is analyzed in Sec. 5.

2 Foldable Programmable Matter

2.1 Single-Vertex Foldable Programmable Matter

The single-vertex FPM is the basic building block of all FPM structures. By definition, all crease lines in a single-vertex FPM are incident upon one vertex which we denote as the origami vertex. The regions surrounded by creases or boundaries are rigid. The degree of the single-vertex FPM refers to the number of incident creases.

Intuitively, intersecting the single-vertex FPM with a sufficiently small sphere yields a kinematically-equivalent spherical chain. An origami vertex located inside the FPM results in a spherical closed chain, while an origami vertex located on the outer boundary results in a spherical open chain. Let us consider the joint angle parameterization of a single-vertex FPM. A configuration is given by sector angles, ω, and dihedral angles, θ, as shown in Fig. 2. Structural symmetry is not required, but is beneficial when creating reconfigurable multi-vertex FPM patterns from single-vertex units. For a given sheet, the sector angles are properties of the rigid facets and thus constant. Changing configurations requires only the change in dihedral angles. The configuration space contains all valid angle configurations, allowing self-touching, but not self-crossing configurations.

A necessary condition for a rigidly foldable origami vertex is that it satisfies the condition for loop closure; the composition of rotation matrices representing the motion about the vertex must be identity [6, 20]. Let matrix A represent the composition of the rotation matrices, where \( R_x(\omega) \) is a rotation about the x-axis by \( \omega \) and \( R_z(\theta) \) is a rotation about the z-axis by \( \theta \).

\[
A = R_x(\omega_0)R_z(\theta_0)\ldots R_x(\omega_2)R_z(\theta_2)R_x(\omega_1)R_z(\theta_1) = I
\]  (1)

Copyright © 2010 by ASME
Typically, a simple spherical chain with revolute joints has \( n \) simultaneously available degrees of freedom for the given configuration. The boat shown in Fig. 2 consists of degree-4 vertices. The boat shown in Fig. 3(a) provides an analysis of simple multi-vertex networks which arise when origami vertices have common creases. [22] provides an analysis of simple multi-vertex networks consisting of degree-4 vertices. The boat shown in Fig. 3(a), with 4-bar spherical linkages connected in serial, is such a structure. Actuating a single DOF of any of the spherical 4-bar chains will allow the structure to fold as shown in Fig. 3b, assuming the linkages are not in the flat configuration and are biased in the right direction. In the flat configuration, each spherical 4-bar has 2 DOF instantaneously.

Despite having results valid for all single-vertex FPM, generalization of the mobility analysis to the multi-vertex case is nontrivial except for simple examples like the boat; the single-vertex results are only sufficient to explain local behavior of multi-vertex designs.

### Multi-Vertex Foldable Programmable Matter

Multi-Vertex FPM can be represented by crease patterns, networks which arise when origami vertices have common creases. [22] provides an analysis of simple multi-vertex networks consisting of degree-4 vertices. The boat shown in Fig. 3(a) has 2 DOF instantaneously.

\[
R_x(\omega_i) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\omega_i) - \sin(\omega_i) & 0 \\ 0 & \sin(\omega_i) & \cos(\omega_i) \end{bmatrix}
\]

\[
R_z(\theta_i) = \begin{bmatrix} \cos(\theta_i) - \sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

The partial derivative of an orthogonal matrix evaluated at identity is skew-symmetric. Thus, \( \frac{\partial A}{\partial \theta} \) is a skew symmetric matrix. The nonzero components of the matrix can be represented as a vector, and each vector is an axis of rotation of the spherical linkage [21]. We write the rotation axis corresponding to the rotation \( \theta_i \) as \( \hat{\theta}_i \). Valid velocities for the linkage are the set of \( \dot{\theta}_i \) which satisfy Eqn. (4) for a set of \( \dot{\theta}_i \).

\[
\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_n \end{bmatrix} = 0
\]

The nullity of the matrix \( [\hat{\theta}_1 \ \hat{\theta}_2 \ \ldots \ \hat{\theta}_n] \) yields the instantaneously available degrees of freedom for the given configuration. Typically, a simple spherical chain with revolute joints has \( n - 3 \) DOF, where \( n \) is the number of joints, but internal symmetry may lead to additional freedoms. For instance, the flat configuration in Fig. 2a has \( n - 2 = 6 \) DOF instantaneously because the rotation axes span a 2D surface only (rank = 2). Observe that the \( n - 3 \) formula is retrieved when the axes span 3D space (rank = 3) as in Fig. 2b.

### 3 Motion Planning

#### 3.1 Single-Vertex FPM

Single-Vertex FPM is closely related to a spherical (open or closed) chain in terms of motion planning as we saw in Fig. 2. To better understand planning for a spherical chain, it may be necessary to begin with a planar chain.

Motion planning for a planar chain is a well-researched field and the literature is extensive both in deterministic approaches [23, 24] which exploit the topological structure of configuration space as well as in probabilistic approaches [25–27] which attempt to connect valid points in configuration space generated by random sampling. Avoiding obstacles or self-collision when analyzing the configuration space or when sampling has been regarded as a difficult problem although [28] tried to combine both approaches to find a collision-free motion. Sampling solutions usually perform collision detection on generated configurations and discard those that fail.

In the mathematics and computational geometry community, however, collision-free motion planning for a planar chain has been solved using the Carpenter’s Rule Theorem [29, 30]. The theorem states that any kind of simple planar open (closed) chain can be unfolded to form a straight line (convex polygon) while maintaining simplicity. According to this theorem, it is always possible to pass through a straightened (convexified) configuration to get a collision-free motion between two configurations.

![FIGURE 3: A simple multi-vertex crease pattern (a) and the resulting folded sheet (b).](image-url)
Determining the existence of a path which does not pass through the trivial configuration remains an open problem. Algorithmic approaches to unfolding using energy functions have also been developed using gradient-based control laws [31].

FIGURE 4: Folding a corner-cube as implemented in our simulator.

Similar unfolding results have been derived for spherical chains which in turn show how single-vertex origami can be unfolded using noncolliding motions [32, 33]. Although there is no guarantee if the motion planning based on the Carpenter’s Rule Theorem is optimal, for example, in terms of energy consumption for physical robotic origami device, it is obviously applicable to the single-vertex FPM because it avoids self-collisions. Figure 4 shows an example using Carpenter’s Rule based algorithm to fold a corner-cube. The joint angles generated for the unfolding were simply replayed in reverse. Such motion plans will be adapted for controlling physical devices by mapping the joint angles to the corresponding embedded actuator inputs and sensor readings.

3.2 Multi-Vertex FPM

As applications for single-vertex FPM are limited, generalization to the multi-vertex case is highly desired to make more complicated structures. In terms of linkage theory, we can say that the multi-vertex case is kinematically equivalent to a complex of spherical closed chains. The single-vertex results are, however, limited to local behavior of multi-vertex designs. Thus, the corresponding motion planning method for the complex will not be obtained by naive generalization, as stated in Sec. 2.2. [34] provides a simulator for multi-vertex FPM designs in which all joints move simultaneously, based on projecting desired velocities for all joints into the constraint space. The approach is to augment the matrices of Eqn. (4) to include all rotation axes about all vertices and solve for the velocities. The directionality of each fold must be known beforehand. In general, whereas any two configurations of a single-vertex FPM have been shown to be connected, the same cannot be said for multi-vertex designs. The initial and goal configurations may be in disconnected components of the configuration space.

Consider an analogous case in the 2D plane. [35] successfully studied the motion planning problem for planar star-shaped manipulators formed by joining one end of planar open chains to a movable common point and fixing the other end, forming a planar closed chain complex. Collision avoidance, needed for construction of a physical device, has yet to be incorporated.

3.3 Folding into intermediate shapes

Instead of folding a complicated shape directly from a flat sheet, a very successful approach has been to first fold an intermediate shape such as a chain of edge-connected cubes or right-angle tetrahedrons which can be universally transformed to 2D or 3D voxelated shapes [2]. Moreover, in 2D it is possible to obtain (asymptotically) collision-free motion by finding a Hamiltonian path which traverses all pixels in any given continuous 2D shape and then applying the Carpenter’s Rule Theorem. Figure 5a shows the initial folding steps in our simulator, and Fig. 5b shows the cubes reconfiguring from a line into the Hamiltonian path of a wrench head using strictly-expansive motions [29]. 3D shapes can also be voxelated using a similar linear chain of primitives [2, 3]. However, the connectivity needed between adjacent modules in the chain to span 3D prevents folding from a single flat sheet.

FIGURE 5: Folding a chain of cubes from a single sheet in the Foldable Programmable Matter Simulator (a) and folding the cubes into the Hamiltonian path for a wrench head in 2D (b).
4 FOLDABLE PROGRAMMABLE MATTER EDITOR

The Foldable Programmable Matter Editor (FPME) was created (1) to provide a convenient interface for a user to specify the layout and folds of multi-vertex FPM and (2) to simulate the folding of that FPM. The interface is similar to a 2D CAD package, with additional options for entering data pertaining to folds. In general, the information necessary for representing FPM can be separated into either crease pattern or fold information. This data is then stored in our two XML schemas, the Programmable Matter Markup Language (PMML) for geometric and physical property data and the Controllable Matter Markup Language (CMML) for fold sequences and angles. These schemas are covered in 4.1. The topics from Sec. 2 and 3 are closely related to the PMML and CMML, respectively, in that Sec. 2 discussed the structure of FPM covered by PMML and Sec. 3 discussed motion planning which results in CMML. For physical devices, the CMML will be translated into embedded control software.

Separation of the layout from the folding allows a modular design which benefits reconfigurability. Multiple fold sequences may be specified for the same crease pattern. For simplicity, each fold sequence starts from the flat state; all configurations are linked through the flat state. Note that this ignores, but does not preclude, the possible existence of another path between folded configurations. The data is then used to carry out the fold sequence in our simulator, which utilizes nVidia PhysX for dynamic physics simulation.

4.1 PMML & CMML

The Programmable Matter Markup Language (PMML) is an XML schema designed to hold the necessary information to create a specific multi-vertex FPM crease network. The PMML schema is shown in Tab. 1. The associated data includes the name of the sheet, the geometry of the rigid bodies, and the geometry and physical properties of the joints. The format and nomenclature have been chosen to closely represent data structures present in our PhysX-based simulator. Thus, individual rigid bodies are referred to as actors and are composed of triangles either created by the user or created by applying Delaunay triangulation and constraining to the rigid body outline. The triangulated bodies are used for collision detection. Joint edges are those connecting two rigid bodies; PMML specifies the IDs of the connected bodies, coordinates which give the axis of rotation, and spring and damping constants. Thus, single-vertex FPM subunits are not specified directly; rather, they occur at the intersections of the joint axes of adjacent rigid bodies.

Folding sequences have been added using Controllable Matter Markup Language (CMML), an XML schema for sequences of fold events. The CMML schema is displayed in Tab. 2. For each fold, the joints involved and their desired final angles are specified. There is no limit to the number of joints that can be moved concurrently, nor to the number of fold steps. A desired speed is stored to be used when motor actuation (rather than spring-actuation) is used.

Fold sequences can be replayed in the simulation environment. Any motion planning algorithm for FPM can be simulated if the output compiles to the CMML format and the PMML document precisely describes the geometry of the given FPM.

TABLE 1: The PMML Schema.

```xml
<PMML>
  <Title> title </Title>
  <Actor>
    <ID> actor id </ID>
    <Triangle> x1 y1 x2 y2 x3 y3 </Triangle>
  </Actor>
  ...
  <Joint>
    <ID> joint id </ID>
    <Coords> x1 y1 x2 y2 </Coords>
    <ActorIDs> id1 id2 </ActorIDs>
    <SpringConst> k </SpringConst>
    <DampConst> c </DampConst>
  </Joint>
  ...
</PMML>
```

TABLE 2: The CMML Schema.

```xml
<CMML>
  <Title> title </Title>
  <Fold>
    <EventNumber> event number </EventNumber>
    <ID> joint id </ID>
    <FoldAngle> desired final angle </FoldAngle>
    <Velocity> desired velocity </Velocity>
  </Fold>
  ...
</CMML>
```

4.2 Implementation and Interface

The interface of the FPME was created to facilitate the design of multi-vertex FPM. Fundamentally, it is a visual tool to create, save, and modify PMML and CMML. The editor has a separate internal data representation, but imports and exports PMML and CMML. Graphics and interface design are handled using OpenGL and FLTK, respectively; the program is coded in C++.
The editor encompasses various tools for inputting information for PMML: drawing, translation, rotation, scaling, mirroring, angle and length measurement, merging vertices, merging edges, splitting edges, copying and pasting triangles, creating a grid or circular lattice, and grouping triangles into rigid bodies. Rigid bodies, or actors, are currently created by drawing individual triangles and grouping them together. Once two rigid bodies sharing a common edge have been created, the edge is automatically made into a revolute joint. The lattice generation tool is helpful for large FPM with repeated structures, such as the cubes in Fig. 5a.

Once the layout of the flat folded FPM has been entered, the user is able to specify actuation. Selecting one or more joints, the user enters a fold angle, spring constant, and damping constant. Different line styles are used to display mountain (positive angle) and valley (negative angle) folds. The user can step through a fold sequence, entering different fold angles for each joint at each step. If no new fold angle is entered, the angle from the previous step is carried through. For larger FPM, the preferred method of CMML generation is to have the motion planning software generate CMML as output.

The current implementation of the editor uses a brute force data structure, scaling linearly with the number of triangles present. Each triangle points to three edges and each edge points to the two vertices it contains. Although edges and vertices may be shared between triangles, they are maintained separately for each triangle. Several operations (marquee selection, lattice creation, copy, paste) run in $O(n^2)$, where $n$ is the number of triangles in an actor. Adding, removing, and merging vertices and edges are $O(n)$ operations. Upcoming revisions to the editor will feature a half-edge data structure, reducing the runtimes for most functions and making larger designs practical.

### 4.3 Simulation Environment

An important feature of the FPME is its ability to simulate and display fold sequences stored in CMML. The user is then able to use feedback about collision points and velocities to adjust PMML properties like geometry and spring constants or CMML properties such as angular velocities, desired fold angle, and fold order. Depending on the FPM being created, the user may select whether the actuators are motors or torsion springs. The motors are velocity-controlled, exerting up to a maximum torque on a revolute joint to reach a desired angular velocity and rotating until they reach a desired position. Spring-based actuation is meant to mimic the action of tensile actuators such as Nitinol wire. Additionally, the simulator may directly load PMML and CMML templates for evaluation.

The fold simulator incorporates nVidia’s PhysX technology for dynamic physics simulation. GPU-based acceleration is available when using nVidia hardware. As numerical error is a concern, simulation properties have been adjusted so that length scales are on the order of one. Current simulations use a single set of material properties; we will work with our collaborators to model the materials used in FPM and add a corresponding ‘materials’ tag to the PMML specification. Once physical prototypes have been tested, joint angles will be mapped to the corresponding actuator inputs and sensor readings. CMML will then be translated into control inputs.

The simulation environment is integrated into the editor. The combination has been able to create and fold intricate structures,
such as the chain of cubes shown in Fig. 5a. Also refer to the origami bird base from Fig. 6 shown folding in Fig. 7. Successful iterations of design and testing expedite the design process. Once a promising design has been reached, the folded structure’s response to an applied load can be analyzed.

5 STRENGTH ANALYSIS

Modeling the behavior of a programmable matter system under mechanical load is an important phase of the design process. A model allows the designer to determine both the module design and configuration that best suits the mechanical requirements.

The strength of a programmable matter system is determined by (1) the properties of the constituent modules and (2) the strength of the connections between modules. In order to develop a lumped parameter model, we assume the connections between modules are more compliant than the modules themselves.

5.1 Model

In the early design stage, the exact geometry and physical method for connecting modules is not precisely defined. Indeed researchers have developed systems using a variety of connection methods including mechanical latches [2, 36–38], magnetics [14, 39, 40], electrostatics [41], and solder [42]. To allow the designer to study arbitrary connection methods, we define connections between modules using a generalized 6 DOF spring [43, 44].

Consider two elastically coupled bodies $A$ and $B$ as shown in Fig. 9a. Frame $a$ attached to body $A$ and frame $b$ attached to $B$ are coincident. A $6 \times 6$ stiffness matrix maps the finite displacement of frame $b$ relative to $a$ defined by a twist $\Delta T_{ab}^a$ to the wrench $w_{ab}^b$ B applies to the elastic connection to $A$ given by:

$$w_{ab}^b \approx K \Delta T_{ab}^a \rightarrow \begin{bmatrix} f_{ba}^a \\ \tau_{ba}^a \end{bmatrix} \approx \begin{bmatrix} K_I & K_c \\ K_e & K_o \end{bmatrix} \begin{bmatrix} \Delta p_{ba}^a \\ \Delta \theta_{ba}^a \end{bmatrix}$$

(5)

where $K_I$, $K_o$, and $K_c$ represent the $3 \times 3$ translational, rotational, and coupling stiffness matrices respectively.

We choose frames $a$ and $b$ to be located at the center of stiffness because it is a unique point that maximally decouples $K$ [45]. The stiffness terms can be determined from elasticity theory or finite element analysis [46].

The simulator [47] written in MATLAB uses the nonlinear equation solver fsolve to find the equilibrium position of the modules under a static load condition. The nonlinear solver allows the designer to model nonlinear effects such as collision and rigid motion due to joint imprecision.

5.2 Example

The simulator allows the designer to study the effect of a relative change in stiffness between two or more connection methods.

As an example, consider the method for folding configurations of cubes from a sheet depicted in Fig. 8. Each set of six faces in Fig. 8a first folds to a cube (Fig. 8b) and subsequently the cube chain can fold to a configuration (Fig. 8c).

This form of programmable matter has three connection methods between square face tile modules: face: connections between modules sharing a face, (Fig. 9a), edge: connections at edges that meet when the sheet folds to a cube (Fig. 9b), and hinged: folded edges (Fig. 9c). Each connection method has an associated stiffness matrix. For this example, assume the connection method being modeled resulted in face and edge connections having isotropic stiffness. The stiffness matrices can be approximated as:

$$K_{face} = K_{edge} = \begin{bmatrix} 1E6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1E6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1E6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1E1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1E1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1E1 \end{bmatrix}$$

(6)

All elements of $K_{hinged}$ are the same as $K_{face}$ and $K_{edge}$ except...
Because the folded edges are relatively weak in resisting moments about \( e_3 \), regardless of connection method, we approximate \( K_{hinged} \) to be 1.

![Image](https://example.com/image.png)

**FIGURE 10:** A beam of 48 tile modules in the editor (a), folding (b) and deforming under static load (dark arrows) with fixed tiles (white triangles) (c).

We show a beam example handled using our toolchain. Figure 10a shows 48 tiles with connectivity described in Fig. 8a. Figure 10b shows the tiles in our folding simulator, reconfiguring into a 2\( \times \)4 cube beam. Figure 10c depicts the beam under transverse loading. White arrows indicate fixed modules and dark arrows indicate the applied load.

We can explore how the relative change in stiffness of one connection type with respect to another affects the apparent stiffness of the beam. Given the stiffness values from Eqn. 6 the beam in Fig. 10c deflects 3.2mm under a 40N load. Increasing the face stiffness by a factor of four reduces the displacement to 1.7mm while alternatively increasing the edge stiffness by a factor of four reduces the displacement to only 2.3mm. This may indicate that the design should focus on maximizing the stiffness of the face to face connection method.

### 6 CONCLUSION

We have created a toolchain to expedite the process of designing foldable programmable matter structures. At the concept stage, the user can draft the crease pattern layout or enter it using PMML. We have presented algorithms for motion planning and control for multi-vertex FPM; the toolchain supports folding intermediate shapes such as edge-connected chains of cubes. The motion plan is converted to CMML and the fold process can be verified by dynamic simulation. Once a folded structure is reached, the method of strength analysis can be used to determine under applied load.

Additional work remains to bridge the gap between simulation and physical devices. Material properties are currently lacking from the toolchain components, but can be readily incorporated as they are known. The idea is to characterize sensors and actuators embedded into the rigid sheets, and implement control based on the information stored in CMML. Sensor possibilities include, but are not limited to, strain gauges. As for actuators, existing prototypes rely on springs and actuator wire, using magnets to latch components together. Additionally, layout issues such as the placement of actuators and routing of traces, have yet to be incorporated into the toolchain.

Other difficulties lie ahead in the synthesis of mechanical materials for FPM construction. Small-scale actuation, especially in the presence of multiple overlapping layers of material, is a challenge. Material thickness issues are also exacerbated by overlapping layers; in practice, nesting multiple folds leads to alignment issues and failure to reach desired fold angles. However, much promising work is being done on the construction of such devices [1, 17, 18].

A simulation video of a sheet folding into a chain of cubes as in Figure 5a is available at http://kumar.grasp.upenn.edu/folding5box.m4v. An end-to-end video for an origami boat is also available at http://kumar.grasp.upenn.edu/boatend2end_title.mp4.

### ACKNOWLEDGMENT

We gratefully acknowledge support from DARPA grant W911NF-08-1-0228, DOE grant P200A060275, ARL grant W911NF-08-2-0004, and ONR grant N00014-09-1-1031.

### REFERENCES


sium on, 0, p. 432.


