Programmable Assembly With Universally Foldable Strings (Moteins)

Kenneth C. Cheung, Erik D. Demaine, Jonathan R. Bachrach, and Saul Griffith

***Abstract*—Understanding how linear strings fold into 2-D and 3-D shapes has been a long sought goal in many fields of both academia and industry. This paper presents a technique to design self-assembling and self-reconfigurable systems that are composed of strings of very simple robotic modules. We show that physical strings that are composed of a small set of discrete polygonal or polyhedral modules can be used to programmatically generate any continuous area or volumetric shape. These modules can have one or two degrees of freedom (DOFs) and simple actuators with only two or three states. We describe a subdivision algorithm to produce universal polygonal and polyhedral string folding schemas, and we prove the existence of a continuous motion to reach any such folding. This technique is validated with dynamics simulations as well as experiments with chains of modules that pack on a regular cubic lattice. We call robotic programmable universally foldable strings "moteins" as motorized proteins.**

***Index Terms*—Biologically inspired robots, cellular and modular robots, folding robots, kinematics, micro/nano robots.**

I. Introduction

THE EXACT method by which a 1-D code translates into a 3-D structure in biological protein folding is currently unknown (although science is making great progress). Still, the complexity and diversity of 3-D structures that are accessible by this 1-D to 3-D approach have long been appreciated [1].

This paper seeks to demonstrate the completeness and applicability of simple forms of the 1-D to 3-D strategy in designing new robotic systems that can take any shape. For clarity, we will primarily discuss Euclidean space-filling curves in 2-D and 3-D, with a brief analysis of more general space filling curves, within the context of the presented algorithms. Further, we show the ability of these systems to geometrically achieve the proposed results through continuous motion without self-intersection. While the examples that are provided address Euclidean orthogonal lattices in 2-D and 3-D, the concepts and algorithms are extensible to non-Euclidean lattices and space tilings (many of the experiments and simulations have been successfully repeated with space-filling right-angle-tetrahedron chains).

Powerful strategies already exist to design discretized robotic systems with units that pack onto a lattice [2]. Many examples have been built (Atron, Fracta, I-Cube, M-Tran, Molecube, Telecube, Superbot, Microunit, Crystalline, Robotic Molecule, Stochastic Modular Robots, etc.), utilizing various schemes for unit attachment, detachment, and self-manipulation [3]. In this study, we propose that introducing a connectivity constraint—that all units of a lattice robot are chained together as a string, configuring to a space-filling curve—greatly simplifies the mechanical design of lattice robots, while retaining the ability to universally reconfigure. The essential ability of units in a metamorphic system to travel across, attach to, and detach from other units [4] becomes unnecessary. As each unit is constrained by the previous unit (which is in turn constrained by its previous unit), transformational periodicity and symmetry—which are key aspects of universally reconfigurable lattice systems [2]—can be maintained with one degree of freedom (DOF) per unit.

There is also prior work on folding robots [5] and robotic origami [6], which shows the promise of serially performing simple and single-DOF operations to form complex shapes. Recent work has shown self-folding planar origami sheets [7]. The main distinction, here, is that our approach folds 1-D to 3-D, while classical and robotic origami folds 2-D to 3-D. We chose to work toward the simplest possible unit design, with low material loss in configuration. Planar folding (2-D to 3-D) results in a maximum of square loss of material to reach some shapes, whereas string folding (1-D to 3-D) results in a constant-factor loss for all shapes, as we explain next.

II. Folding Schema

Peano [8] and Hilbert [9] first constructed 2-D and 3-D space-filling curves. These recursive infinite curves define a connected linear mapping of 2-D or 3-D space. They have been of interest across many areas of mathematics and information science. It is known that a connected series of smaller and self-similar objects (polygons in 2-D and polyhedra in 3-D) connected at similar hinges can exist in a chained configuration that takes the form of any pixellated 2-D [10], [11] or voxellated 3-D [12] object. While, in 2-D, some of these chains are known to continuously fold without intersection into all pixellated 2-D shapes [13], no

Manuscript received February 21, 2010; revised September 11, 2010 and January 18, 2011; accepted March 16, 2011. Date of publication June 9, 2011; date of current version August 10, 2011. This paper was recommended for publication by Associate Editor A. Ijspeert and Editor J.-P. Laumond upon evaluation of the reviewers' comments. This work was supported by the Massachusetts Institute of Technology Center for Bits and Atoms and the U.S. Army Research Office under Grant W911NF-08-1-0254 (Programmable Matter).

K. C. Cheung is with Massachusetts Institute of Technology Center for Bits and Atoms, Cambridge, MA 02139 USA (e-mail: kenny.cheung@cba.mit.edu).

E. D. Demaine is with Massachusetts Institute of Technology Center for Bits and Atoms and the Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139 USA (e-mail: edemaine@mit.edu).

J. R. Bachrach and S. Griffith are with Massachusetts Institute of Technology Center for Bits and Atoms and Otherlab, San Francisco, CA 94107 USA (e-mail: jrb@csail.mit.edu; saul@otherlab.com).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org.

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

Digital Object Identifier 10.1109/TRO.2011.2132951

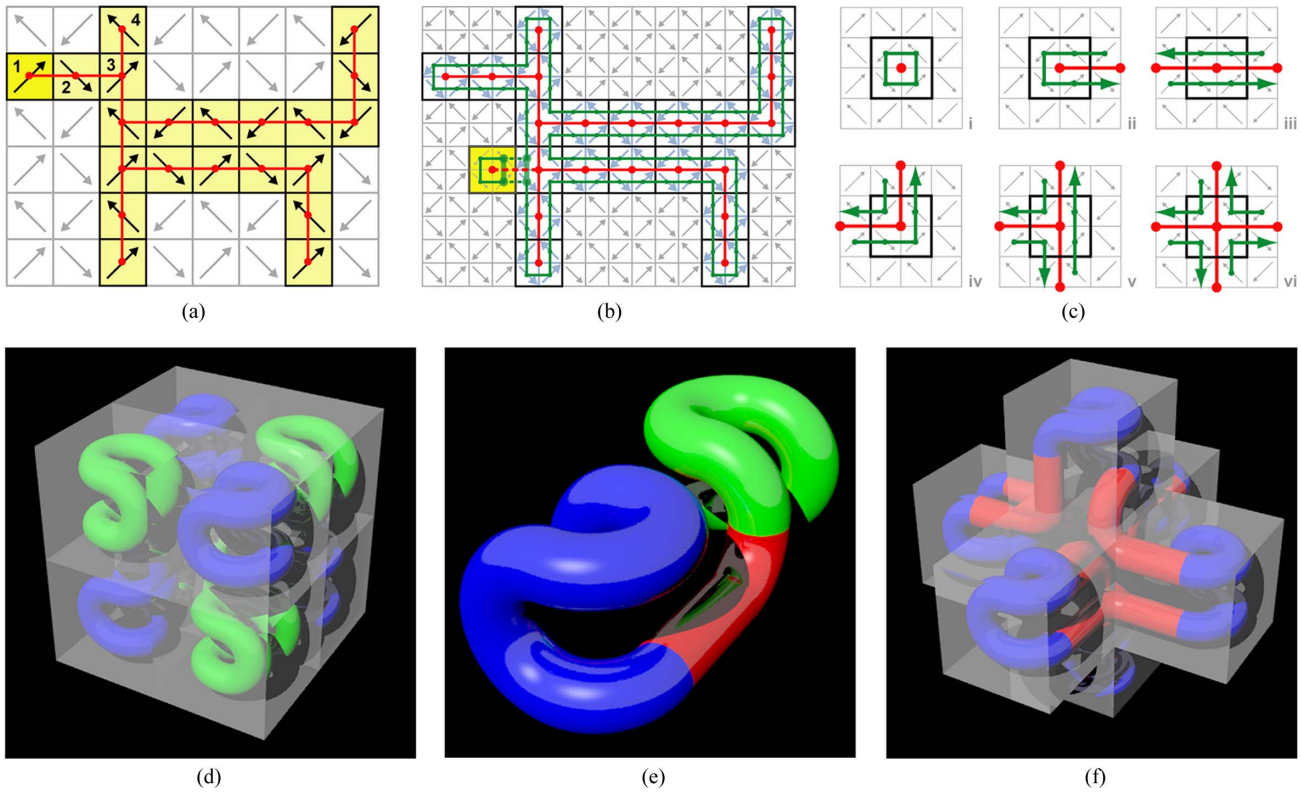

Fig. 1. (a) Spanning tree is shown with red lines that connect the nodes (red dots) at the center of each "pixel." (b) Subdivision of each pixel into four "subpixels," each group of which forms a Hamiltonian path, and any assembly of which contains a Hamiltonian path (the yellow tile shows the construction by the addition of new tiles). (c) Six possible "pixel" configurations and their "subpixels" demonstrating edge connectivity. (d) Eight cubic voxels, arranged in constructive lattice, each comprised of eight cubic subvoxels with Hamiltonian loops of connectivity shown in green and blue. (e) Constructive connection between two paths to make a circuit that includes all subvoxels of the original two. (f) Fully face connected voxel, connected to all six adjacent voxels, enabling a connected path to all surrounding voxels from any given voxel.

prior results attain continuous foldability with a method that is generalizable across 2-D and 3-D. This is a goal that we achieve here.

We start by viewing the collection of pixels as a graph. The nodes of the graph are the centers of the pixels, and the edges in the graph connect adjacent pixels. In order to construct the shape by folding, it must be possible to connect all of the nodes in the graph through a Hamiltonian (single and nonintersecting) path. Not all graphs have Hamiltonian paths; for instance, consider the yellow dog shape in Fig. 1(a), which does not contain a Hamiltonian path. It is well known that even determining whether a graph has one is an NP-complete problem [14].

We can get around the problem with an efficient subdivision algorithm [11], by replacing each pixel with a collection of subunits that contains a Hamiltonian circuit. For our example of square macrotiles, the simplest subdivision method that satisfies these requirements is to divide each square equally into four smaller squares, thus increasing the number of pixels by a factor of 4. To illustrate what we obtain, we consider the spanning tree of an original macro-pixellated figure, such as Fig. 1(a), which is a subgraph that contains all of the pixels and a subset of the edges. Enough edges must be included such that any two pixels of the original graph are still connected by a single path, which may go through any number of other edges and pixels. Every graph has at least one spanning tree (for the types of graphs we discuss, the upper bound on the number of spanning trees for a graph of n nodes is $2^{O(n)}$). If a Hamiltonian path does not exist, as shown in Fig. 1(a), then the spanning tree must be branched. If each of our original macropixels is replaced by four micropixels, a one-micropixel wide perimeter can be created around any original spanning tree, as shown in Fig. 1(b). This perimeter is always a Hamiltonian path (now explicitly a circuit) on the enlarged graph that takes the new subpixels as its nodes. It follows that this method exposes as many Hamiltonian circuits as there are spanning trees for the original graph.

This construction can also be viewed inductively, and it is this perspective that allows for a simple extension to 3-D. Instead of laying out the shape, finding a spanning tree, and subdividing all pixels to create the Hamiltonian path, this path can be constructed by repeatedly adding subdivided Hamiltonian circuits that contain macropixels, as sets of micropixels, until the desired shape is constructed. The Hamiltonian circuits of any two adjacent subdivided pixels may be merged to form a single circuit, which in turn may be merged with any other adjacent subdivided pixels or circuits formed in this fashion. With each addition, the path is extended to encompass the new subpixels (by replacing adjacent paths with new connecting paths, which are shown as

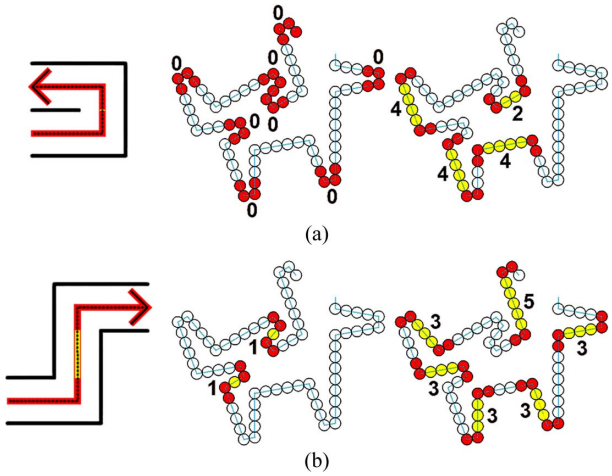

Fig. 2. Types of turning sequences that comprise a valid path, starting and finishing with red-colored units. The numbers at each sequence indicate the unit separation between the start and finish of the sequence. (a) Consecutive u-turns or even-number-of-units separated u-turns. (b) Odd-number-of-units separated chicanes (for clarity, not all are highlighted).

red in Fig. 1). By similar construction, any object can be built additively by combination of Hamiltonian circuits in this manner.

The minimal 2-D tile set to programmatically determine the folds according to the embedded sequence is a left-turning tile, a right-turning tile, and a tile with a straight final position. Further diagrammatic explanation of the resulting strings and folding schemes can be found in Figs. 2 and 3(a). In any configuration that results from this algorithm, the set of features in the path is comprised of straight lines and right-angle turns. Turns in the same direction (a right-hand or left-hand turn followed by another right-hand or left-hand turn, respectively) only occur consecutively or with an even number of units separating them in a straight line [as shown in Fig. 2(a)], and chicanes (a right-hand turn followed by a left-hand turn, or vice versa) only occur with an odd number of units separating them, in a straight line [as shown in Fig. 2(b)]. Because of this feature, only one bit is required to represent each unit, for a specific shape—turn or no-turn. The direction of a turn is simply based on the previous turn direction and the parity of the number of no-turn units that separate the two.

The simplest Euclidean 3-D case to consider is a cubic macrovoxel that is subdivided into eight cubic microvoxels, with Hamiltonian per-macrovoxel Hamiltonian circuits described by the green or blue paths shown in Fig. 1(d). One entry and exit subface per macroface is required for the Hamiltonian construction used thus far, as demonstrated in the four sides ($\pm$ in two axes) of the square pixels in the 2-D proof. The analogous volumetric pixel (voxel), therefore, has six faces—one for positive and one for negative translation in each axis, as shown in Fig. 1(f)—and requires at least 12 subfaces derived from the microvoxels that comprise it: two on each of the six faces analogous to the two micropixel faces on each macropixel face in the 2-D case. For clarity, the example of two macrovoxel circuits joined to form a single circuit is shown in Fig. 1(e). The cube satisfies these constraints for a subvoxel; it is space filling with

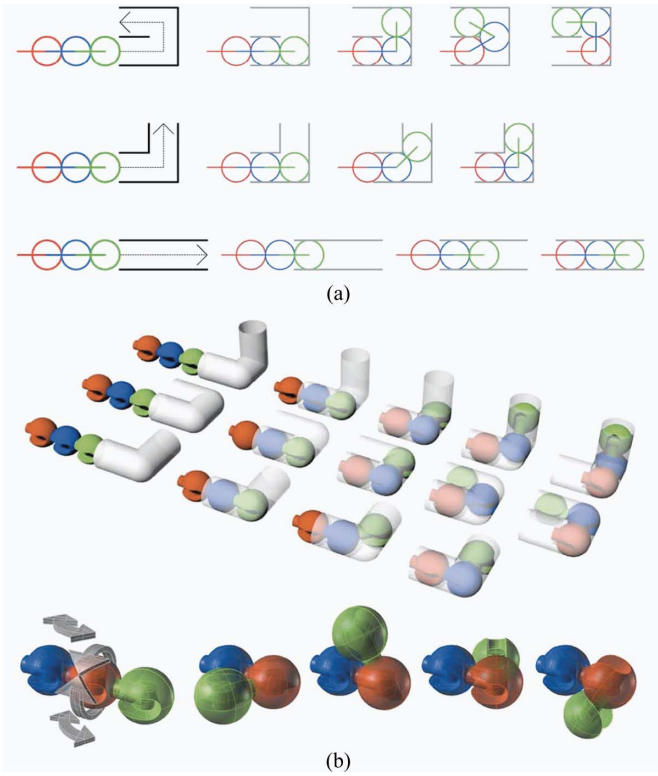

Fig. 3. (a) Types of turning conditions that all paths—constructed with our algorithm—are composed of (u-turn provides kinematic constraining condition for universal folding by continuous motion of disks on square lattice). (b) Analogous diagram of kinematics required for universal folding by continuous motion for spheres on cubic lattice.

six pairs of subfaces on six faces with normals to those faces on three axes.

To perform the Hamiltonian circuit construction in 3-D, a lattice constructed from alternating transformations of the Hamiltonian circuit for the volumetric pixel (voxel) allows for analogous circuit adjacencies between every voxel. These transformations are accordingly tiled in space, as shown in Fig. 1(d), and joined to form a final path, as shown in Fig. 1(d)–(f). This algorithm again exposes as many Hamiltonian circuits as there are spanning trees of the original macro-voxellated figure (at least one, and for this 3-D system, there is an upper bound of $3^{O(n)}$ spanning trees for a figure composed of n voxels). Selection of a spanning tree will be discussed later in this paper.

As in the 2-D construction, we can use an additive construction technique to show that these modules can fold from a string to fill any voxellated 3-D object. The fully face-connected case is presented in Fig. 1(f), where it is demonstrated that, indeed, the return paths to six additional cubes—one connected to each of the six faces of the original (central) cube—are possible. The two (green and blue) circuits shown here may be utilized in the same method, in rotation, and the direction of travel along the circuit may be right handed or left handed. This is an arbitrary choice; it is only required that the algorithm continues to follow the handedness that is initially decided upon.

As with the 2-D system, such a configuration in 3-D is implicitly required to have a Hamiltonian circuit with certain turning

motifs, if it is a configuration of the chain. In this configuration, the set of features in the path is again comprised of straight lines and right-angle turns. For aggregated turns within the same plane, the same rules as in the 2-D system apply (turns in the same direction only occur consecutively or with an even number of units separating them, in a straight line, and chicanes only occur with an odd number of units separating them, in a straight line).

For this example, the minimal 3-D unit set to programmatically determine the folds according to the embedded sequence is an x-axis turning tile, a y-axis turning tile, and a z-axis turning tile. As in the 2-D system, where each unit turns left or right relative to its own coordinate system and the preceding tile only, each unit in the 3-D system is defined to turn in a direction in 3-D space relative to the tile preceding it, according to its own local coordinate system.

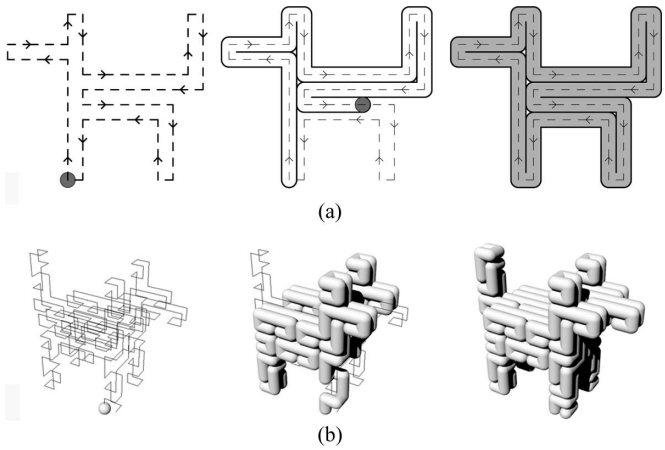

Fig. 4. (a) Minkowski sum of unit disk and Hamiltonian path. (b) Minkowski sum of unit sphere and Hamiltonian path.

III. Continuous Foldability

The spanning graphs that are produced by the aforementioned methods are non-self-intersecting paths, and since the resulting paths are circuits, the position of the beginning and end of a constructive string is arbitrary. If we take a virtual string, fold it into a path constructed with these methods, and then pull on the ends (regardless of their position on the string), we get a single loop with no knots. The folding of the string into, or out of its intended figure does not require passage through the spatial position of previously folded components. However, this does not address self-interference of units during the folding. There exists a subset of constructions for many (perhaps all) figures that produce non-self-interfering folding, when folded sequentially. Future work will explore whether this is also true for folding in parallel. However, neither are intrinsic requirements for these systems, given that it is geometrically possible for a string of particularly shaped units to achieve any configuration defined by our construction methods, including the intuitively most self-interfering configuration. One end of the string could be essentially threaded into the figure at a point on the border of the figure and fed through the path of the final configuration. Given the theorem that this continuous non-self-intersecting motion works between any two grid configurations of a string of zero thickness [15], it suffices to prove that every possible set of turning features in any final configuration can coexist on these strings, without collision, during the feeding motion.

When considering a physical string composed of a chain of discrete units, there are many possible shapes of the units as well as methods of attachment between each unit. For simplicity, we consider each unit to be a disk or ball and attach each unit to the center of the previous unit so that each maintains this distance while it is free to rotate about this point (the center of the previous unit).

For the 2-D case, consider a string of unit-diameter disks, connected together by hinges that pivot about the center of the previous unit. Each unit allows the following unit to rotate a fixed distance ($2\pi/3$ rad in either direction) about the point that is antipodal to its own hinge. Thus, the center points of any three units may subtend an angle no less than $\pi/3$ rad in either direction.

A valid grid configuration is a configuration of the string of units such that the centers of the units are on points of the unit square grid and such that neighboring units are tangent (at midpoints of grid edges). Note that such a configuration is implicitly required to have a Hamiltonian cycle with certain turning restrictions, in order to even be a configuration of the string, resulting from the previously described algorithm. One end of the string could be essentially threaded into the figure at a point on the border of the figure and fed through the path of the final configuration.

Every local situation that arises in our Hamiltonian path construction (straights, u-turns surrounded by straights, and turns surrounded by straights) can be navigated by three disks, while maintaining their connections to the previous and next disk. The rotational configuration spaces of units in these local situations, intersected along any valid assemblage of straights and turns, are therefore connected. As illustrated in Fig. 4(a), a continuous area that the units can fit into—and that does not self-intersect—can be trivially constructed as the Minkowski sum of the Hamiltonian circuit and the unit disk. Therefore, the linear grid configuration can be folded into any grid configuration, so by transitivity, the string can be folded between any two grid configurations, without self-intersection.

A simple extension of this proof shows that there is also a continuous non-self-intersecting motion between any two grid configurations of a string of units in 3-D, where the elements along the string are unit-diameter spheres (instead of disks), with final configurations centered on a 3-D cubic unit grid. Consider again that units along such a string are connected together by hinges that pivot about the center of the previous unit. Each unit allows the following unit to rotate a fixed distance of $2\pi/3$ rad in one direction about the point that is antipodal to its own hinge, and $\pi/2$ rad in either direction about the axis from its center to the center of the previous unit.

In this system, turns in orthogonal planes are also, conceptually, orthogonal in that they do not constrain their corresponding planar configuration spaces. Therefore, any rotational motions

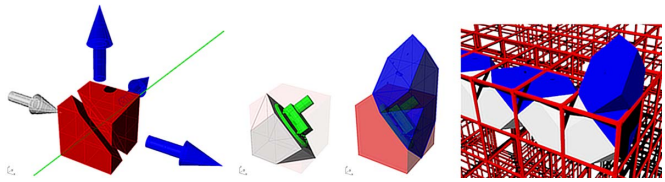

Fig. 5. Mechanical design of the C-motein. The green part in the center represents a servo motor with a bearing at the interface between adjacent modules; the red grid on the right shows how the modules pack into final configurations.

that are required to move through a sequence of turns in one plane will not affect the ability of the string to achieve rotational motions in an orthogonal plane. Given this, it suffices to prove that each of the three projections, for each axis, of all possible 3-D paths possesses the same characteristics as the 2-D paths discussed earlier. Since the three orthogonal projections of the 3-D path construction algorithm presented earlier follow the same basic turning aggregation interval rules as the 2-D system, the intersection of the rotational configuration spaces of any valid assemblage of turns still results in a continuous configuration space for each projection. Therefore, the linear grid configuration can be folded into any 3-D grid configuration, and therefore, the string can be folded from any 3-D grid configurations to any other 3-D grid configuration.

Furthermore, as illustrated in Fig. 5, there is always a continuous non-self-intersecting solid that the units can fit into, which can be constructed as the Minkowski sum of the Hamiltonian circuit and the unit sphere. Therefore, once again, the string can be folded between any two configurations. It is important to note here that the rounded corners of the path are necessary not only for our proof technique but for a physically realistic system as well.

IV. Folding Tool(s)

In summary, the aforementioned constructions prove that any space-filling structure can be built of a string of connected geometric primitives. These structures can be folded without self-intersection, and it is geometrically possible for any valid configuration to reach any other valid configuration through continuous motion. The length of strings (number of units) produced with these methods scales linearly with the number of discretized pixels or voxels in the desired shape (in the given examples, $4n$, where n is the number of pixels of the original figure in 2-D, and $8n$, where n is the number of pixels of the original figure in 3-D). Such favorable scaling, combined with the small number of required primitive components in these constructions, suggests that they are a promising direction for high-throughput fabrication methods, through mesoscale printing processes, microelectromechanical systems, or even chemical or biological systems [16].

Our workflow starts with an algorithmic representation of a shape. This is then evaluated over a lattice, to construct a Hamiltonian path, which is then processed into the code for the string (see Fig. 11). We have fabricated small-scale proofs of concept, and are experimenting with kinematics models that are a subset of that which is described in the proof in order to rapidly adopt known mechanisms (novel motors, bearings, or connectors are not required to implement this strategy). Large-scale fabrication processes are simulated (see supplementary movies 1–3), and a key aspect of the simulations is that each unit in the simulation solves for a local solution at each time step—the global solution is a product of the aggregated local results of the programs of each unit. This program can be as simple as a string of instructions for each single DOF revolute joint, such as "turn or go straight," in the case of 2-D, or "turn clockwise, counterclockwise, or go straight," in the case of 3-D.

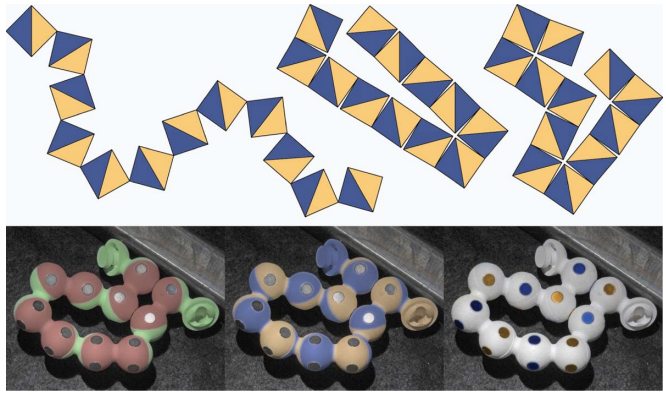

Fig. 6. Interconnection patterning: 1-D to 2-D (top) and 1-D to 3-D (bottom), a working model with magnets. The top example of a vertex-connected 1-D to 2-D string is shown for clarity. The direct 1-D to 2-D analog for our implementation of the 1-D to 3-D string would be shown but with alternating (left and right) faces permanently connected and the hinges occurring across the diagonals so that the device comes out of plane and into the third dimension during the folding process (similar to 2-D configuration of Rubik's Snake toy). The left bottom diagram shows the physically discrete units of our 3-D implementation as alternating red and green.

V. Experiment and Simulation

In order to validate the applicability of these techniques for engineered systems, we implemented examples of robotic programmable universally foldable strings (moteins) in simulation and built short robotic strings in order to verify the mechanics used in the simulations. Two scales of the cubic lattice-based motein (C-motein) will be discussed: here—a centi-C-motein (cC-motein) with about a 1 cm diameter unit size and a molecuboid C-motein (MC-motein) with about a 10 cm diameter unit size. These two examples possess the same fundamental kinematics (the difference between them is that the latter is about ten times larger than the former).

The larger physical MC-motein (see Fig. 8 and supplementary movie 5) was built to quickly test these ideas in a fully actuated test-bed. Each unit is constructed similarly to a molecube unit [17], with a single bearing and servo motor housed in a printed thermoplastic (Dimension Elite) chassis. The smaller physical cC-motein (see Fig. 6 and supplementary movie 9) shown here was built as a passive kinematic test-bed to study connectivity and reconfigurability characteristics. These were printed as complete strings in acrylic (3D Systems InVision si) with magnets pressed in afterward. We expect that a fully actuated physical cC-motein is realizable, given the simplicity of these units.

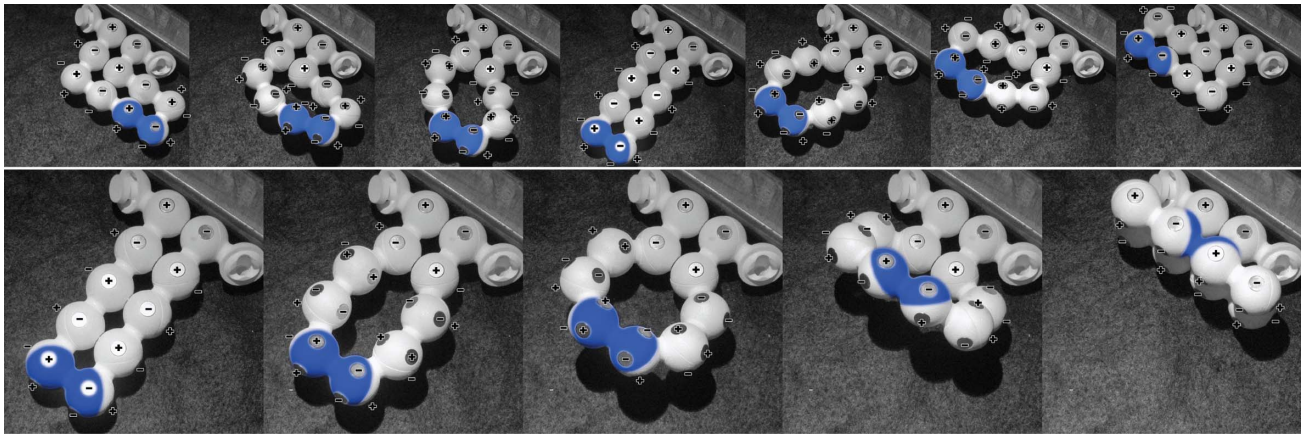

Fig. 7. Kinematic scaling with subloops, showing the ability to perform translation (top) and rotation (bottom) routines with closed kinematic loops of the universally foldable chain.

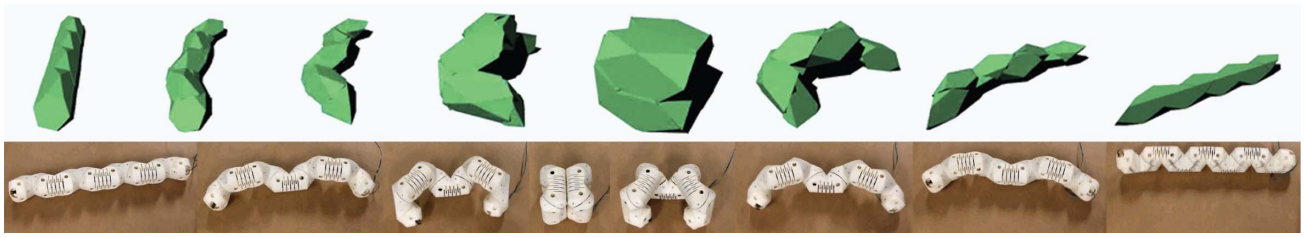

Fig. 8. ODE simulation (green) and actual movement of eight-module string, showing the ability of chains of the module geometry to perform parallel folding.

Both C-moteins presented here, as well as all simulations, employ a single revolute joint as the single DOF per unit. The MC-motein robot and all simulations employ a servo motor to actuate this joint. Observed and simulated mass density is about 0.4 gm/cm^3 for both C-moteins.

Our goal in this section is to address practical implications, such as motion-planning schemes (i.e., parallel versus serial folding) and their impact on the amount of time that it would take for this type of robot to perform (re)configuration.

A. Design

We chose the most basic kinematic design that provides continuous motion between all necessary final configuration states, for a string system that closely packs on a cubic lattice. Each module in this device would have to be able to position its following module at any one of three positions (shown by the blue arrows in Fig. 5) relative to its previous module (shown by the white arrow in Fig. 5, considering rotational symmetry). This is accomplished with a single rotational joint about a longest interior chord connecting two vertices of the cube (see the green line in Fig. 5).

An even bisection with a plane that is orthogonal to this axis produces the regular hexagonal bisection of the cube (the resulting cross-sectional face is a regular hexagon). When these are arranged as a string with rigid bonds between hemicubes arranged such that each is a mirror image of its neighbor (across the bonding plane shared with the neighbor cube), we obtain the module design shown on the right-hand side of Fig. 5.

While this design can form any shape (fits with the constructive proof of universal foldability), it meets only a subset of the requirements to navigate all folding motions with continuous motion. We are providing this example for simplicity and in order to suggest that the fundamental algorithms for folding schema presented here may be applied to chained versions of most existing reconfigurable robotic systems [17], [18] with useful results. The rest of this paper addresses this simplified model, in both simulation and hardware. This hexagonally bisected cube geometry was first realized in the field of reconfigurable robotics with the molecube system [17], with its cubic modules whose connections are reconfigurable, as opposed to being constrained as a chain. The example that we present here is also somewhat similar to Rubik's Snake toy (which can be approximated by other robots, such as Atron) but with modules that closely pack on a cubic lattice and with a corresponding

$$2(\mathrm{ArcTan}(1/(\sqrt{2}-(1/\sqrt{2})))) \approx 109.4713^\circ \qquad (0a)$$

dihedral angle between bearing faces (instead of the $\pi/2$ dihedral angle of Rubik's Snake module).

Since this architecture includes an integral backbone, data and power transfer can simply run through a continuous conductor bus (reliable electrical collectors/slip rings are trivial to integrate if the ability to perform large numbers of net twists is desired). The outer surfaces of the chains are free to be left to carry functions other than reconfiguration, such as carrying payloads.

These unnecessary but potential interconnections between spatially adjacent units that are far apart along the string may still be desired, for instance, to parallelize power and

data transfer. Corresponding connector plates require only twofold rotation symmetry, with a very simple layout. This is perhaps easiest to visualize in the 2-D example at the top of Fig. 6. If each unit in the square lattice has male (−) connector plates (blue in Fig. 6) on the down-string side and female (+) connector plates (orange in Fig. 6) on the up-string side, then the string will always pack with proper pairing of connector plates. This follows from the path construction algorithm, as a single-square macropixel mates properly on all four sides with another macropixel, and the connection patterning of the macropixel surface remains intact throughout the path construction operation (some faces are "replaced" with permanent connections in order to create the string and, therefore, become irrelevant to the interconnection scheme).

With our implementation in 3-D, each unit in the cubic lattice has four potential connector faces, since the up-string and down-string faces are permanently attached to other units. These four faces may be grouped into two groups of two, as divided by the hexagonal bisection line previously described. If we designate that the group that is connected down-string has male (−) connector plates and the group that is connected up-string has female (+) connector plates, for all units (as shown in Fig. 8), then all valid configurations will result in correctly paired connector plates. As with the 2-D example, a single 3-D macrovoxel mates properly on all six sides with other macrovoxels, and the connection patterning of the macrovoxel surface remains intact throughout the path construction operations.

Fig. 7 (and supplemental movie 9) shows examples of larger-than-unit scale translation and rotation routines that can be performed by closed subloops of these string robots. As such, one might imagine suites of locomotion and/or manipulation robots that can reconfigure between functions—with a key attribute that they are all composed of strings of simple and identical units.

B. Implementation

To summarize the design, we chose a string system that packs on a cubic lattice, with one rotational DOF per module. The single corresponding bearing/hinge and actuator per module has three states: $\{0, 2\pi/3, -2\pi/3\}$. Therefore, the only information that has to be sent to each unit is "stay straight," "turn counterclockwise," or "turn clockwise," for each module, together with an addressing scheme.

To obtain a better sense of the kinematic constraints of the design when a number of units are connected in a string, we built a number of prototypes and found the system to be quite flexible. For instance, the bearing gap seen in Fig. 5 may be significantly enlarged (along the rotation axis) to create a sparse structure, as the symmetry of the folding is such that in the final configuration the main body of the modules will occupy corresponding corners of the cubes on the packing lattice, such that incident faces of the modules still line up (for structural and/or latching purposes).

The final design (shown in Fig. 8 and supplemental movies 4 and 5) utilized the compact integration of the dynamixel AX-12 servo motor and ring gear of the molecube design [17]. Power and data were carried on the stock dynamixel three-wire bus. The packing lattice for this string has a 27-in^3 unit. Since this design desirably gears down the AX-12 units by a factor of 3, and the closed loop servo mode of the AX-12 unit does not allow full rotation, reconfiguration commands to each module were accomplished by first implementing a timed directional free run, followed by a switch to servo mode for precise positioning of the goal state. The geared-down drive systems provide less than one-tenth of a degree of resolution; therefore, we find it unnecessary to include an active connection interface to connect units that are far apart along the string but spatially adjacent in their final configuration.

Open dynamics engine (ODE) [19] was used for the simulations, with values for dimensions, mass, and motor torque matched to the compact robots in Fig. 8. Continuing work addresses smaller ($\sim$100 μm) and larger ($\sim$1 m) folding systems on various lattice geometries, but here, we will primarily discuss mass, force, and dimensional scales that are conventional in the field of reconfigurable robotics.

C. Folding Simulation

Our goal with the simulations is to broadly investigate the characteristics of these systems when they have large numbers of units. We describe initial experiments with folding using three motion-planning techniques here—naïve parallel folding, reverse explosion planning, and the probabilistic roadmap method (PRM).

1) *Parallel Folding:* Simple parallel folding, where the folding instruction is distributed to all modules and they are allowed to actuate simultaneously with even power distribution, can work well but is quite sensitive to the nature of the initially chosen folding path. Primarily due to inertial factors at these scales, the effect that we observe in the simulations is delayed folding in the middle or anchored end of the string, as the modules toward the end(s) complete their configurations first. These precompleted ends must fit together easily in order for this strategy to be efficient and not require refolding.
2) *Reverse Explosion:* In experimenting with various refolding heuristics, we observe that it may be easier to unfold than to fold. This reverse explosion method starts with the target configuration in simulation and applies repulsive forces to unfold the chain, while recording the joint angles at predefined time steps. The folding forces include outward forces to all modules in the direction originating from the center of mass of the entire ensemble, as well as a repelling force between nonadjacent modules during the unfolding. Therefore, the modules not only explode from the center but maintain distance from nearby modules as well. During this part of the process, the motors on the joints are turned OFF and the joints themselves merely enforce the passive between-module kinematic constraints. The resulting motion plan is to servo to the list of recorded angles in reverse chronological order. Each recorded set of angles acts as a target to which we servo until the largest angle error is less than some given threshold. Fig. 9 gives pseudocode for this reverse explosion algorithm.

```
record (chain, gain)
 anglez ← list()
 until is_unfolded()
  append(anglez, record_angles(chain))
  v ← radial_force(center_of_mass(chain))
  v ←- v + repelling_force(chain)
  apply_force(chain, v)
 return anglez

playback (chain, anglez, threshold)
 foreach angles in reverse(anglez)
  errs ← angle_errors(chain, angles)
  while max_over(errs) > threshold
   errs ← servo(chain, angles, gain)

reverse_explosion (chain, gain, threshold)
 anglez ← record(chain, gain)
 playback(chain, anglez, threshold)
```

Fig. 9. Pseudocode for reverse explosion algorithm.

We compute the radial force as a vector in the outward direction or a module's position minus the center of mass of the ensemble

$$e_i = \lambda \frac{p_i - c}{d} \tag{1a}$$

where λ is the radial force gain, p_i is the module position of the ith module, d is the diameter of the configuration, and c is the center of mass

$$c = \sum_{i=0}^{n} \frac{p_i}{n}. \tag{1b}$$

The radial force increases with distance from the center of mass, causing modules on the outside to explode faster than ones in the inside. This also lowers the chance of collisions by encouraging modules to be maximally distant from each other. The repelling force is computed as a $1/d^2$ force between nonadjacent modules

$$r_i = \sum_{j=0}^{n} \frac{\gamma}{(p_j - p_i)^2} \tag{1c}$$

where γ is the repulsion force gain. In order to speed up the simulation, a maximum unit distance for application of repulsion forces may be applied.

Results from this reverse explosion method are shown in Fig. 10 and movies 6 and 10 in the supplementary materials. Fig. 10 shows a 160-module chain and a configuration shape of a wrench. Relevant parameters for these simulations are $\lambda = 0.1$, $\gamma = 1$, and maximum squared error threshold of 0.15. The companion video shows the algorithm that runs on a number of canonical geometric shapes.

The naïve parallel folding could be viewed as a form of this reverse explosion method but with zero intermediary steps. Inversely, the reverse explosion method could be viewed as a composition of naïve folding steps. As such, the number of required intermediary steps for a successful reverse explosion method is also very sensitive to the nature of the initially chosen folding path.

D. Folding Analysis

Noting a large scope of serial, parallel, or other heuristic motion-planning methods that can be applied to our folding schema, we have analytically estimated lower and upper bounds on the amount of time that the folding process will take to complete, assuming significant inertial effects, and critical damping of the string's motion (see Fig. 11).

Our estimation for the upper bound on folding time assumes that the largest possible inertia term for the nth module will result from the 0th through $(n-1)$th modules that exist in a straight-chain configuration. For parallel folding, this upper bound on the time to fold is equal to the time to fold of the module at the middle or anchored end of the string (2c), shown below. For serial folding, this upper bound on the time to fold is the sum of all t_n, shown below in (2d). In the following equations, α represents the angular acceleration, τ represents the torque, which is assumed to be a fixed scalar, I represents the moment of inertia, and δ represents the one unit distance between modules.

(For a string of length n units)

$$\frac{2\pi}{3} = \frac{1}{2}\alpha_n t_n^2; \quad \alpha_n = \frac{\tau}{I_n} \tag{2a}$$

(time to fold as a function of distance, torque, and inertia)

$$I_n = \sum_{i=1}^{n} m(i\delta)^2; \quad |t_n| = \sqrt{\frac{4\pi \sum_{i=1}^{n} m(i\delta)^2}{3\tau}} \tag{2b}$$

(maximum inertial term; "least folded" chain configuration)

$$|t_n| = \frac{\sqrt{2\pi}}{3}\sqrt{\frac{nm(2n^2+3n+1)\delta^2}{\tau}}; \quad t_{\text{parallel}} = O(n^{3/2}) \tag{2c}$$

(maximum single fold time, for parallel folding)

$$t_{\text{serial}} \approx \int_0^n ki^{\frac{3}{2}}\,di = O(n^{\frac{5}{2}}) \tag{2d}$$

(sum of fold times for serial folding).

Our estimation for the lower bound on fold time assumes that the smallest possible inertia term for the nth module will result from the 0th through $(n-1)$th modules existing in a folded configuration whose center of mass is a distance of one unit from the center of mass of this nth module. Similar to the upper bound calculation, for parallel folding, this lower bound on the time to fold is equal to the time to fold of the module at the middle or anchored end of the string in (3b), shown below; for serial folding, this lower bound on the time to fold is the sum of all t_n, as in (3c), shown below.

(For a string of length n units)

$$I_n = (nm)\delta^2; \quad |t_n| = \sqrt{\frac{4\pi(nm)\delta^2}{3\tau}} \tag{3a}$$

(maximum inertial term; "most folded" chain configuration)

$$|t_n| = 2\sqrt{\frac{\pi}{3}}\sqrt{\frac{(nm)\delta^2}{\tau}}; \quad t_{\text{parallel}} = \Omega(n^{1/2}) \tag{3b}$$

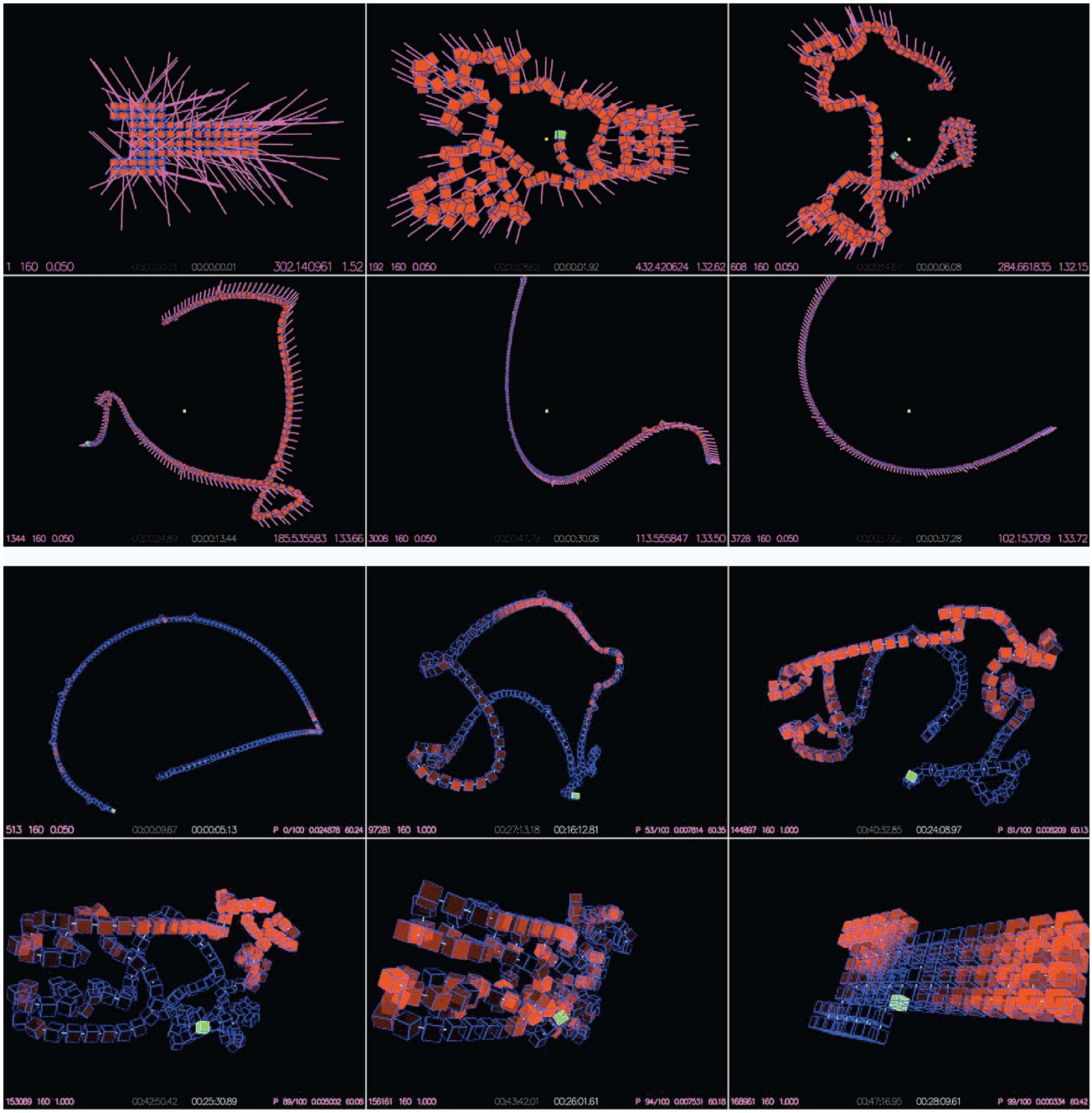

Fig. 10. Reverse explosion motion planning method. (Top) First, a prefolded target shape of 160 chained cubes is subjected to the radial unfolding forces applied from its center of mass outward. While the chain unfolds, multiple successive snapshots of all joint angles are recorded over time. (Bottom) From the unfolded shape, the joint actuators are servoed to the recorded angle snapshots in the reversed sequence. As a result, the chain folds into the target shape. The brightness of red color denotes normalized amount of joint error.

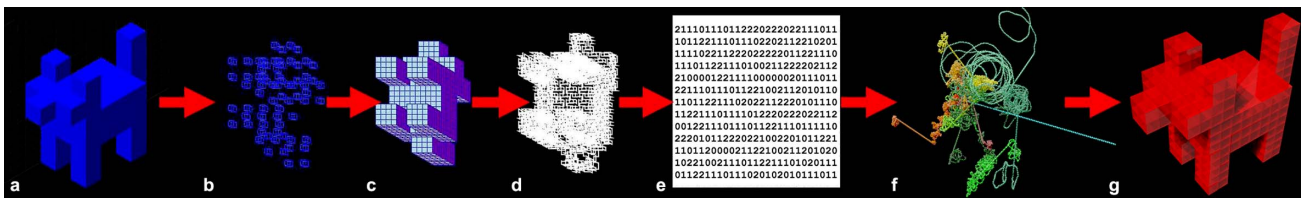

Fig. 11. Automated algorithmic planning for folding any shape. (a) Original stereolithography mesh. (b) Mesh vertices. (c) Lattice model of an object. (d) Folding code. (f) Folding simulation. (g) Final folded object.

(maximum single fold time, for parallel folding)

$$t_{\text{serial}} \approx \int_0^n ki^{1/2} di = \Omega(n^{3/2}) \tag{3c}$$

(sum of fold times, for serial folding).

These results show that for strings with large numbers of units, parallel folding schemes can be much quicker to fold than serial folding schemes, which is not surprising. The slowest n-unit long serial folding scheme will take a factor of n^2 more time to fold than the fastest parallel folding scheme. However, these results also suggest that the least time-efficient parallel folding schemes, in terms of time needed to fold, can be approximately matched by the most time-efficient serial folding schemes due to inertial effects. Therefore, we may consider serial folding strategies as viable even for long strings especially when other advantages are taken into account, such as managing the power requirements of the system or enabling the folded modules to be passive, while folded by a robot [5], [6], [20].

E. Reconfiguration Simulation

So far, we have implied methods of folding between two configurations—by unfolding completely from the first configuration before folding into the second configuration. We would prefer to investigate more efficient paths between any two configurations. In this section, we briefly show the applicability of existing robotic motion-planning strategies to folding digital chains, by demonstrating the use of a PRM. This method of finding motions between configurations, developed for the protein folding problem, has been shown to be reasonably efficient, and has already seen applications in a number of robotic systems [21].

The PRM initially samples configuration space for collision-free configurations. We employed simple random sampling in this example, but more sophisticated sampling algorithms are available. These initial configurations serve as the initial nodes in a roadmap graph. Nearby pairs of nodes are then chosen and approved to be collision free by running a simple local planner until a connected graph is formed. Finally, a path from a start and end configuration is found in the roadmap graph.

There are some particulars, which are unique to digital chains, that we need to address. First, a configuration for a chain is a vector of n joint angles. We could choose configurations with each joint's continuous range of motion ($-2\pi/3$ to $2\pi/3$), but we find that using discrete positions is preferable. All configurations are first filtered using a self-intersection algorithm—for continuous positions, this can employ a high-speed collision detector, such as the Software Library for Interference Detection (SOLID) [22]. However, when the initial configurations are restricted to discrete positions, collision detection can be performed very quickly, since each position can be represented efficiently, and examining the validity of a configuration simply requires stepping around a discrete lattice, instead of summing a long list of vectors. Further, simple patterns and algorithms can be used to prune invalid configurations (e.g., "if your directions say to make four consecutive turns in the same direction on a Manhattan grid, then you have bad directions"). This can be efficiently performed in an $O(n \times n \times n)$ matrix.

Neighbors in the configuration space are found by considering configurations in distance order, to see whether they are reachable using simple linear interpolation local path planning. We use the binary search approach to decide whether intermediate linearly interpolated configurations are self-intersecting. The distance metric between configurations can simply be the L^2 norm or the sum of the squared angle errors between configurations.

We use Dijkstra's algorithm for shortest path determination, which starts by marking the final (goal) configuration node as visited and with zero distance. Then, iteratively, neighbors of newly visited nodes are marked, and their distances to the end configuration are updated as a function of their local distance to their neighbors and their neighbors' best estimate to the end configuration. The smoothing algorithm to shorten paths found using Dijkstra's algorithm is very straightforward and previously reported [23]. Random configuration pairs from the shortest path are chosen, and intervening configurations are removed if there is a locally plan-able path between them.

In summary, with discrete configurations as way points in the PRM, the path can be described efficiently as discrete differences between configurations. In the binary case, the difference between configurations can be described as a Hamming distance [24]. Path fitness can be measured in terms of the hamming difference between the start and end target configurations and the sum of the hamming distances between configurations along the path.

The results are quite promising; see supplementary movie 11, for a simulation of reconfiguration of a 140-module digital chain between the words hello and world and a few other shapes.

VI. Discussion

We see three main topics for future work with this type of folding string: initial selection of a space-filling curve from the space of curves that are possible for a given shape, further work on motion planning, and applications.

The selection of a most suitable spanning tree for a given shape remains an open question. There are many strategies to develop this initial pre-subdivided tree from which the final 3-D space-filling curve is generated. We expect that specific answers will derive from functional (e.g., structural) requirements. Some strategies may derive from the ability to tune anisotropic bulk structural properties. It is relatively simple to achieve high differential strength between the permanent (with one rotational DOF) connections between units along the string and the other spatially adjacent connections (or lack of mechanical connection). Other strategies may relate to reconfiguration.

For example, it may be desirable to perform the least possible turning in the path or as much as possible. The former may be achieved by following the perimeter of a figure and spiraling in as necessary, only branching to fill areas of the figure that cannot be reached with a single spiral. This is suitable for serial folding schemes. One method to achieve relatively many turns involves performing a distance transform to the edge of a voxellated

figure, then constructively generating the spanning graph by starting with the voxels with the highest values (the most interior voxels), and always performing the constructive addition of the remaining voxels to neighbors with lower values (more exterior). Thus, the spanning graph has a medial axis backbone with many spines leading to the extremities so that the folding is most accordion like. This kind of structure has advantages for parallel folding schemes. Such consideration of actuation sequence—and the implication that one can use information about future desired configurations in order to plan earlier configurations—leads us back to the topic of motion planning.

In this paper, we have just touched on the topic of motion planning, in order to show the viability of the design. Prior work in motion planning has shown many techniques that could be applied to universally foldable string robots, which explore the space of folding strategies for reconfigurable systems. Many methods applied to the motion-planning problem for lattice robots, such as subdividing with similarity metrics [25] and simulated annealing [26], can be extended to apply to chained configurations. Perhaps, the most clearly applicable methods are derived from PRMs developed for the protein folding problem, and which already have been specifically applied to robotic folding systems [27]. It is worth noting that the applications of these types of motion planning methods toward closed kinematic loop mechanisms [28], [29] also point toward the wide range of potential functional applications of universally foldable string robots beyond shape making.

We know that geometry is sometimes regarded to be a cornerstone of many functional (i.e., biological) systems, and this may provide an avenue toward programming various types of mechanisms, through geometric arrangement of functional units. The folding system as described here could have functionality superimposed on different pixels or voxels, and sequencing would allow the positioning of those functional components at any desired location in the global 3-D structure. Furthermore, since the string can be folded between any two configurations, this directly implies a route to reconfigurable matter, where a single string with simple actuators could fold from any one configuration to any other in order to serve different and complex mechanical (i.e., locomotion) and/or computational [30] functions.

VII. Conclusion

We have shown a technique to design universally foldable string robots, with proof of existence of continuous motion for self-assembly and self-reconfiguration. These results may further the revolution from analog to digital materials and fabrication processes, through computational tools to employ biologically inspired assembly systems and by enabling low cost and reversible de novo systems. We know how to make communication and computation systems that scale well enough to operate as designed, with Avogadran numbers of units. This is largely achieved through error reduction and correction strategies that make good bets on the physics of the system. Biology shows that these goals can be satisfied in a system to fabricate things, or "programming matter," through the encoding of structural and functional information in 1-D, with a small and discrete set of parts. Furthermore, there is some evidence to indicate that complex biological structures can result from the aggregated behavior of large quantities of discrete components with ever simpler physical models [31].

Reconfigurable robotics has come a long way and has a long and interesting road ahead, that is, toward successful programmed assembly of very large and complex structures [32]; we hope that the techniques presented here will be useful as a method of programmatically making vast libraries of parts from any very basic set of mechanisms. In the shorter term, we hope that with these techniques and the simplifications afforded by having an integral backbone and very low DOF and states per unit, many existing reconfigurable robotics benchmarks might be surpassed—such as the number of active modules in a single system, actuated module size (smallness), and robustness of self-reconfiguration. A crux of many existing reconfigurable robotics systems is the reconfigurable communications and power connections (the ability for modules to attach and detach from each other)—these are difficult and expensive to build; our robots (moteins) are not reliant on such mechanisms.

Clearly, the most exciting and most open problem is that of applications. This technique of algorithmic generation of programs for self-folding matter presents a new method of working toward truly digital artificial fabrication systems. The old question that we strive to answer is how we can effectively and efficiently get from a description of an object to the functional object itself, with an eye toward material life cycles. This study suggests a manner to describe objects by their generative programs so that the description itself is also the very digital information needed to fabricate the object.

Ongoing work is aimed toward addressing folding strategies, including reconfiguration motion planning and the advantages of different geometric properties of the initial lattice used. 2-D and 3-D patchworks of polygons and polyhedra allow for a final result with tuned sparseness and correspondingly faster folding times (due to decreased string length). Other relevant ongoing work includes analysis of bulk properties of these kinds of assemblies [33], development of actuators specifically geared toward this application [34], and cellular computing based models for executing programs across these kinds of modules (such as to compute reconfiguration strategy) with extremely low per-unit cost [35], [36].

Open Source Code

Programs and open source code that execute the algorithms described in this paper are available for research and educational use from the authors.

Acknowledgment

The authors would like to thank the Center for Bits and Atoms, N. Gershenfeld, and J. Jacobson for support and critical discussions, V. Zykov for technical advice with device engineering, and the fab lab network for contextual grounding.

REFERENCES

[1] B. Lewin, *Genes IV*: Oxford: Oxford Univ. Press, 1990.

[2] N. Brener, F. B. Amar, and P. Bidaud, "Designing modular lattice systems with chiral space groups," *Int. J. Robot. Res.*, vol. 27, no. 3, pp. 279–297, 2008.

[3] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self reconfigurable robot systems [grand challenges of robotics]," *IEEE Robot. Autom. Mag.*, vol. 14, no. 1, pp. 43–52, Mar. 2007.

[4] G. S. Chirikjian, "Kinematics of a metamorphic robotic system," in *Proc. IEEE Int. Conf. Robot. Automat.*, San Diego, CA, May 1994, pp. 449–455.

[5] D. J. Balkcom and M. T. Mason, "Robotic origami folding," *Int. J. Robot. Res.*, vol. 27, no. 5, pp. 613–627, 2008.

[6] L. Lu and S. Akella, "Folding cartons with fixtures: A motion-planning approach," *IEEE Trans. Robot. Automat.*, vol. 16, no. 4, pp. 346–356, Aug. 2000.

[7] E. Hawkes, B. An, N. M. Benbernou, H. Tanaka, S. Kim, E. D. Demaine, D. Rus, and R. J. Wood, "Programmable matter by folding," *Proc. Nat. Acad. Sci.*, vol. 107, no. 28, pp. 12441–12445, 2010.

[8] G. Peano, "Sur une courbe, qui remplit toute une aire plane," *Math. Ann.*, vol. 36, no. 1, pp. 157–160, 1890.

[9] D. Hilbert, "Ueber die stetige Abbildung einer Line auf ein Flächenstück," *Math. Ann.*, vol. 38, pp. 459–460, 1891.

[10] E. D. Demaine, M. L. Demaine, D. Eppstein, G. N. Frederickson, and E. Friedman, "Hinged dissection of polyominoes and polyforms," *Comput. Geom.: Theory Appl.*, vol. 31, no. 3, pp. 237–262, 2005.

[11] S. Griffith, "Growing machines," Ph.D. dissertation, Mass. Inst. Tech., Cambridge, MA, 2004.

[12] E. D. Demaine, M. L. Demaine, J. F. Lindy, and D. L. Souvaine, "Hinged dissection of polypolyhedra," in *Proc. 9th Workshop Algorithms Data Structures*, 2005, vol. 3608, pp. 205–217.

[13] R. Connelly, E. D. Demaine, M. L. Demaine, A. Ribo, and G. Rote, "Locked and unlocked chains of planar shapes," in *Proc. 22nd Annu. ACM Symp. Computat. Geom.*, 2006, pp. 61–70.

[14] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter, "Hamilton paths in grid graphs," *SIAM J. Comput.*, vol. 11, pp. 676–686, 1982.

[15] S. H. Poon, "On Unfolding 3D Lattice Polygons and 2D Orthogonal Trees," in *Proc. 14th Annu. Int. Comput. Combinatorics Conf.*, 2008, pp. 374–384.

[16] J. H. Chen and N. C. Seeman, "Synthesis from DNA of a molecule with the connectivity of a cube," *Nature*, vol. 350, pp. 631–633, 1991.

[17] V. Zykov, E. Mytilinaios, B. Adams, and H. Lipson, "Robotics: Self reproducing machines," *Nature*, vol. 435, pp. 163–164, 2005.

[18] G. M. Whitesides and M. Boncheva, "Beyond molecules: Self-assembly of mesoscopic and macroscopic components," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 8, pp. 4769–4774, 2002.

[19] Open Dynamics Engine (ODE) (2004). [Online]. Available: http://www.ode.org/ode.html

[20] P. J. White, C. E. Thorne, and M. Yim, "Right angle tetrahedron chain externally-actuated testbed (RATChET): A shape-changing system," in *Proc. Int. Design Eng. Tech. Conf. Comput. Inform. in Eng. Conf. IDETC/CIE*, 2009, vol. 7, pp. 807–817.

[21] C. J. Chiang and G. S. Chirikjian, "Modular robot planning using similarity metrics," *Auton. Robots*, vol. 10, pp. 91–106, 2001.

[22] Software Library for Interference Detection (SOLID) (2004). [Online]. Available at http://www.win.tue.nl/∼gino/solid/

[23] N. M. Amato and G. Song, "Using motion planning to study protein folding pathways," *J. Comput. Biol.*, vol. 9, no. 2, pp. 149–168, Apr. 2002.

[24] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Techn. J.*, vol. 29, no. 2, pp. 147–160, 1950.

[25] A. Pamecha, I. Ebert-Uphoff, and G. S. Chirikjian, "Useful metrics for modular robot motion planning," *IEEE Trans. Robot. Automat.*, vol. 13, no. 4, pp. 531–545, Aug. 1997.

[26] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Sci. New Series*, vol. 220, no. 4598, pp. 671–680, 1983.

[27] G. Song and N. M. Amato, "A motion-planning approach to folding: From paper craft to protein folding," *IEEE Trans. Robot. Automat.*, vol. 20, no. 1, pp. 60–71, Feb. 2004.

[28] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[29] J. Cortés, T. Siméon, and J. P. Laumond, "A random loop generator for planning the motions of closed kinematic chains using PRM methods," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2002, pp. 2141–2146.

[30] M. Boncheva, D. H. Gracias, H. O. Jacobs, and G. M. Whitesides, "Biomimetic self-assembly of a functional asymmetrical electronic device," *Proc. Nat. Acad. Sci.*, vol. 99, no. 8, pp. 4937–4940, 2002.

[31] S. Sun, P. D. Thomas, and K. A. Dill, "A simple protein folding algorithm using a binary code and secondary structure constraints," *Protein Eng.*, vol. 8, no. 8, pp. 769–778, 1995.

[32] G. M. Whitesides and B. Grzybowski, "Self-assembly at all scales," *Science*, vol. 295, no. 5564, pp. 2418–2421, 2002.

[33] P. J. White, S. Revzen, C. E. Thorne, and M. Yim, "A general stiffness model for programmable matter and modular robotic structures," *Robotica*, vol. 29, pp. 103–121, 2011.

[34] A. Knaian, "Electropermanent magnetic connectors and actuators: Devices and their application in programmable matter," Ph.D. dissertation, Mass. Inst. Technol., Cambridge, MA, 2010.

[35] T. Fukuda, Y. Kawauchi, and F. Hara, "Dynamic distributed knowledge system in self-organizing robotic systems; CEBOT," in *Proc. IEEE Conf. Robot. Automat.*, 1991, pp. 1616–1621.

[36] N. Gershenfeld, D. Dalrymple, K. Chen, A. Knaian, R. Green, E. D. Demaine, S. Greenwald, and P. Schmidt-Nielsen, "Reconfigurable asynchronous logic automata," in *Proc. 37th Annu. ACM SIGACT-SIGPLAN Symp. Principles Programming Languages*, 2010, vol. 45, no. 1, pp. 1–6.

Authors' photographs and biographies not available at the time of publication.