# Folding Arbitrary 3D Shapes with Space Filling Chain Robots: Reverse Explosion Approach to Folding Sequence Design

J. Bachrach, V. Zykov, S. Griffith
Makani Power Inc
2175 Monarch Street, Alameda, CA, 94501 USA
jrb@pobox.com, victor@makanipower.com, saul@makanipower.com

## Abstract

*We propose a method for arbitrary 3D shape formation by folding chains of hinged polyhedra. This paper presents an algorithm that finds a folding trajectory from an unfolded chain to a target folded chain in a Hamiltonian path configuration. The algorithm is based on the observation that unfolding is generally easier than folding. We can find an unfolding trajectory by applying a radial force to the folded chain while recording the joint angles over time. The folding trajectory is then achieved by servoing to successive ensemble joint snapshots from the reverse explosion trajectory. We have applied the algorithm for cube and tetrahedral chains and show results for a few solids.*

## 1 Introduction

Automatic formation of irregular 3D objects is difficult in practice. In contrast, Biology achieves this routinely by folding sets of chained parts into geometric patterns dictated by mutual components' interactions [2], [10], [8], [5], [6], [1]. This turns out to be a powerful approach that allows for error correction and replication as well as reliable construction of complex physical 3D objects.

In this paper, we describe a method for target configuration design for folding chain robots into space-filling 3D solids. We use a particular chain robot design especially conducive to forming 3D solids through folding: our robots are composed from multiple space-filling polyhedra, such as cubes interconnected with universal joints and right angle tetrahedra interconnected with hinge joints.

Folding chains of hinged polyhedra into arbitrary 3D solids requires that we complete the following two tasks: (1) Take a description of a target solid, for example, in the form of a closed triangular surface mesh and produce a final configuration including target positions for monomers and joint angles for the joints so that the chain forms a Hamiltonian path through the shape; and (2) Plan for folding the monomers from an unfolded configuration into the final configuration. Specific procedures include rasterization of a target 3D shape, generation of a Hamiltonian path that visits all internal voxels of the target shape, along which the chain robot segments will be lined up in the folded configuration, and, finally, design of a folding trajectory for the chain robot.

This paper presents an algorithm and software tool involved in step (2) of fold trajectory planning. A companion paper [3] describes the tools and algorithms for step (1) of rasterization and Hamiltonian path formation.

We are currently focussing on two space packing parts: cubes and right-angled tetrahedra. We are using two universal joints to connect neighboring cube modules and two hinge joints to connect neighboring tetrahedral modules. Figure 1 shows a chain of cubes and a chain of tetrahedra.

We assume that we have rasterized the 3D model and generated a Hamiltonian path and target folded configuration of the chain. After producing a Hamiltonian path, we next place the modules along the path and calculate the target joint angles. In this section, we introduce our technique for planning folding trajectory by playing an explosion trajectory in reverse.

Once we have the positions of all modules and connections and angles of all joints, then we need to calculate a plan to go from an unfolded straight chain to this target configuration. The difficulty of this lies in the fact that it is easy for a long chain to get tangled and stuck. One possible remedy is to fold the monomers one by one, but that still is not guaranteed to avoid all tangles. Another difficulty is that the chains get heavier and heavier as you move from the chain ends to the center, and it requires more and more torque to fold. The ultimate goal is to find a folding strategy that allows for the most amount of parallelism in folding, avoids tangles, and scales to large lengths.

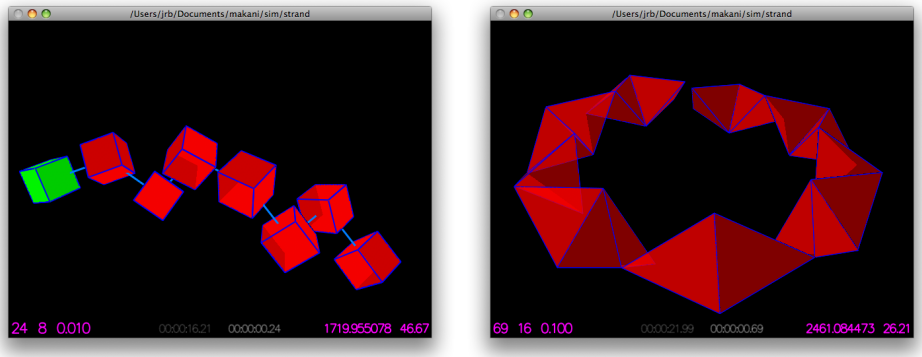


**Figure 1. Chain of cubes connected by universal joints on left and chain of tetrahedrals connected by hinge joints on right.**

## 2 Related Work

Modular robots are traditionally classified into lattice, chain/tree, and mobile architectures [18]. Lattice robots have traditionally been best known for shape versatility, whereas chain/tree architectures were considered a better fit for manipulation and locomotion tasks [7]. Formation of arbitrarily shaped solid objects is considered to be an important and difficult task in the area of modular robotics, because known systems generally require robot structure modification or reconfiguration to achieve an arbitrary shape [15]. Our approach shows how a chain robot with purposefully selected shape of modules and chain kinematics can be successfully utilized for arbitrary 3D shape formation, without the need of reconfiguration into a different non-chain architecture, by only using folding along the chain joints - thus combining important advantages of the two primary modular robot architectures.

Both centralized [12], and distributed [16] [11] [13] control strategies have been used in modular robotics control applications, including control of chain/tree robots. In most cases, however, chain/tree robot reconfiguration was shown in application to manipulation tasks, as opposed to shape formation tasks which we discuss here. Griffith [4] considered only sequential folding. Poon [9] considered motion of chains lying within a grid but did not apply this more generally to full path planning. Twigg and James [17] considered running physics in reverse in rigid body simulations in order to generate special effects.

## 3 Approach

One observation is that it may be easier to unfold than to fold. Our idea is to start from the target configuration, to explode the chain while recording the joint angles, and to servo to the angle snapshots in reverse. The particulars are as follows.

In simulation, we start with the chain folded into its final configuration. We then apply radial forces to modules emanating from the center of mass of the entire ensemble. The motors on the joints are turned off and the joints themselves merely enforce their module to module constraints. While moving outwards at regular intervals we record a snapshot of joint angles across the chain. We stop when the chain has reached a fully unfolded straight configuration. We now have a complete record of the unfolding as a sequence of angle snapshots.

This unfolding sequence can now be used to fold the chain by playing it back in reverse. This can be done again in simulation or for playback on a real robot. Each snapshot acts as a target to which we servo until the largest angle error is less than some given threshold. We need to servo with low enough stepsize and sufficient damping to have the actual folding trajectory mirror the unfolding trajectory so as to not cause tangling.

In order to lower the chance of tangling we introduce a repelling force between unconnected modules during the unfolding. Thus the modules not only explode from the center but also maintain as much distance from nearby modules. Figure 2 gives the pseudocode for the reverse explosion algorithm.

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

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

where $\lambda$ is the radial force gain, $p_i$ is the module position of the $i^{th}$ module, $d$ is the diameter of the chain, and $c$ is the

```
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)
```

**Figure 2. Pseudocode for reverse explosion algorithm.**

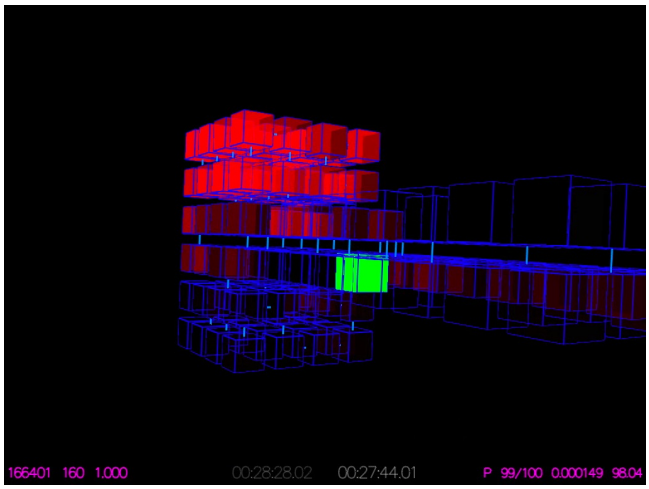


**Figure 4. Chain of 160 cubes connected by universal joints used in Figure 3 refolded into a mallet.**

center of the chain:

$$c = \sum_{i=0}^{n} p_i/n \tag{2}$$

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 $\frac{1}{d^2}$ force between non linked modules

$$r_i = \sum_{j=0}^{n} \frac{\gamma}{(p_j - p_i)^2} \tag{3}$$

where $\gamma$ is the repulsion force gain. In order to speed up the simulation, repulsion forces between only modules within a certain maximum distance are computed.

The complete set of reverse explosion parameters are $\gamma$, the repulsion force gain, $\lambda$, the radial force gain, and the maximum error threshold. ODE simulation parameters include the damping factor, the friction amount, the joint motor maximum force, and the step size.

## 4 Results

In this section we present a number of examples of the reverse explosion algorithm. We ran all our results using the Open Dynamics Engine (ODE) [14] using their shape geometry, collision detection, joints, and friction. The results of physics are rendered within OpenGL using a custom rendering engine. We display simulation properties and allow zooming and rotation, single stepping, part introspection and movie generation.

We ran the algorithm on a couple of shapes with the cubic geometry. Figure 3 shows the algorithm applied to a 160 module cubic chain and the wrench solid. It took approximately 160000 time steps (or 1600 seconds of simulated time) to achieve the wrench shape. Using the same 160 module chain we demonstrated reconfiguration by refolding it into a mallet as shown in Figure 4. It took approximately 150000 time steps (or 1500 seconds of simulated time) to achieve the mallet shape. We also successfully ran the algorithm on the tetrahedral geometry for simple shapes. We ran with $\lambda = 0.1$, $\gamma = 1$, and maximum squared error threshold of $0.15$. Finally, in ODE we ran with $1cm$ diameter objects, $2700kg/m^3$ density, damping set to $0.1$, friction set to $0.1$, and step size set to $0.01$.

The companion video shows the algorithm running on a 160 cube chain folding into a wrench and then refolding into mallet.

We then successfully folded lower and higher resolution wrench and mallet shapes. In particular, we successfully folded a 64 module mallet, a 94 module wrench, and a 192 module mallet. Simulation times grow with chain length but appear to largely reflect the increasing amounts of force required to fold larger subassemblies.

We also attempted to try a simpler approach of servoing from an unfolded chain directly to the the folded configuration. In neither of the wrench and mallet shapes with 160

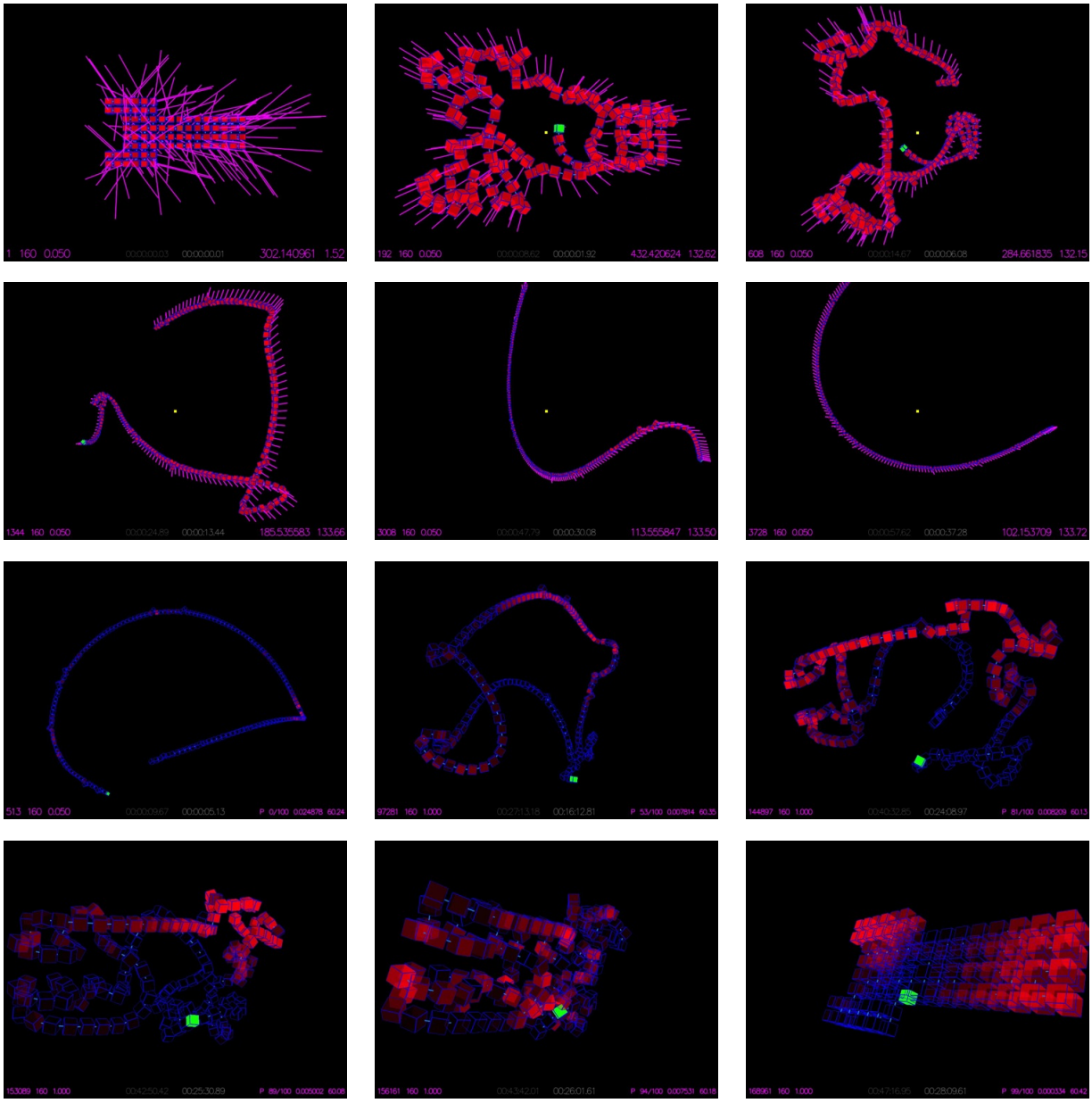

**Figure 3. Planning the folding trajectory: First, a pre-folded target shape of 160 chained cubes is subjected to the radial explosion forces applied from its center of mass outwards. While the chain unfolds, multiple successive snapshots of all joint angles are recorded over time. Next, the joint actuators are servoed to the recorded angle snapshots in the reversed sequence. As a result, the chain folds into the target shape. The top six pictures show the explosion and bottom six show the playback with brightness of red color denoting normalized amount of joint error.**

module resolution were we able to get this to work. In both cases, the chain got tangled and failed to fold. We also tried even lower resolution wrenches and mallets of 24, 64, and 80 modules, and again were unable to fold. Our experience is that chains greater than around twenty modules fail to fold with direct servoing approach, but of course this is very configuration path specific. The point is that folding is difficult and reverse explosion folding makes it possible for much larger chains.

## 5 Conclusion

In this paper, we presented an algorithm for finding a folding trajectory on a polyhedral chain using a reversion explosion. In [3] we show how we can find a target Hamiltonian path using incremental bottom up path merging. Future work will involve using the results of path planning to drive a real robotic chain robot to fold into desired shapes. More work is necessary to prove out the scalability and generality of the approach on more geometries, larger chains, and more shapes. Some form of hierarchy is necessary to scale the approach to bigger chains, although hierarchy in the target path might suffice.

## 6 Acknowledgements

We would like to thank DARPA for their funding under the Programmable Matter program. We would like to thank Erik Demaine, Neil Gershenfeld, Kenny Cheung, Jim McBride, Ara Knaian, and the rest of the MIT Media Lab Programmable Matter team for their helpful suggestions and guidance.

## References

[1] P. Alexander, Y. He, Y. Chen, J. Orban, and P. Bryan. Characterization of protein-folding pathways by reduced-space modeling. In *Proceedings of National Academy of Science USA*, volume 104, pages 11963–8, 2007.

[2] C. Anfinsen. The formation and stabilization of pretein structure. *Journal Biochem*, 128:737–49, 1972.

[3] J. Bachrach, V. Zykov, and S. Griffith. Incremental hamiltonian path creation within 3d solids. Under submission, 2009.

[4] S. Griffith. *Growing Machines*. PhD thesis, MIT, 2004.

[5] S. Kmiecik and A. Kolinski. Characterization of proteing-folding pathways by reduced-space modeling. In *Proceedings of National Academy of Science USA*, volume 104, pages 12330–35, 2007.

[6] S. Lee and F. Tsai. Molecular chaperones in protein quality control. *Journal Biochem Molecular Biology*, 38:259–65, 2005.

[7] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kakaji. M-tran: Self-reconfigurable modular robotic system. *IEEE/ASME Trans. Mech.*, 7(4):431–441, 2002.

[8] C. Pace, B. Shirley, M. McNutt, and K. Gajiwala. Forces contributing to the conformational stability of proteins. *Faseb Journal*, 10:75–83, 1996.

[9] S. Poon. On unfolding 3d lattice polygons and 2d orthoganl trees. In *Proceedings 14th Annual International Computing and Combinatorics Conference (COCOON)*, 1988.

[10] G. Rose, P. Fleming, J. Banavar, and A. Maritan. A backbone-based theory of protein folding. In *Proceedings of National Academy of Science USA*, volume 103, pages 16623–33, 2006.

[11] B. Salemi, P. Will, and W. Shen. Distributed task negotiation in modular robots. *Journal of the Robotics Society of Japan, Special Issue on Modular Robotics*, 21(8):32–39, November 2003.

[12] W. Shen, M. Krivokon, H. Chiu, J. Everist, M. Rubenstein, and J. Venkatesh. Mutimode locomotion for reconfigurable robots. *Autonomous Robots*, 20(2):165–177, 2006.

[13] W. Shen, B. Salemi, and P. Will. Homone-inspired adaptive communication and distributed contorl for conro self-reconfigurable robots. *IEEE Transactions on Robotics and Automation*, 18(5):700–712, 2002.

[14] R. Smith. Open dynamics engine. http://www.ode.org, 2000.

[15] K. Sty. How to construct dense objects with self-reconfigurable robots. In *Proceeding European Robotics Symposium (EUROS)*, pages 27–37, May 2006.

[16] K. Sty and R. Nagpal. Self-reconfiguration using directed growth. In *Proceeding 7th International Symposium Distributed Autonomous Robotic Systems*, pages 1–10, June 2004.

[17] C. Twigg and D. James. Backwards steps in rigid body simulations. In *ACM Transactions on Graphics SIGGRAPH*, 2008.

[18] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian. Modular self-reconfigurable robot systems: Challenges and opportunities for the future. *IEEE Robots and Automation Magazine*, pages 43–52, March 2007.