Scalable Shape Sculpting via Hole Motion: Motion Planning in Lattice-Constrained Modular Robots

Michael De Rosa
Seth Goldstein
Peter Lee
School of Computer Science
Carnegie Mellon University
[mderosa,seth,petel]@cs.cmu.edu

Jason Campbell
Padmanabhan Pillai
Intel Research Pittsburgh
[jason.campbell,padmanabhan.s.pillai]@intel.com

Scalable Shape Sculpting via Hole Motion: Motion Planning in Lattice-Constrained Modular Robots

Abstract— **We describe a novel shape formation algorithm for ensembles of 2-dimansional lattice-arrayed modular robots, based on the manipulation of regularly shaped voids within the lattice ("holes"). The algorithm is massively parallel and fully distributed. Constructing a goal shape requires time proportional only to the complexity of the desired target geometry. Construction of the shape by the modules requires no global communication nor broadcast floods after distribution of the target shape. Results in simulation show 97.3% shape compliance in ensembles of approximately 60,000 modules, and we believe that the algorithm will generalize to 3D and scale to handle millions of modules.**

This paper is submitted to Invited Session: New Trends in Modular Robotics.

I. Introduction

Modular robotics has been the subject of much interest in the research community [1]. Using large numbers of simple modules to replace one complicated, special-purpose device provides benefits in terms of flexibility, robustness, and manufacturing cost. The challenge in these systems lies in controlling large numbers of low-powered, unreliable modules. Motion planning and shape formation for these systems is a good example of such a difficult challenge. Traditional motion-planning algorithms typically require an exploration of the robot's potential state-space. This state-space grows exponentially with the number of degrees of freedom in the system, which in turn can be a linear function of the number of modules in the ensemble. While motion constraints and isomorphisms may dramatically reduce the size of this state-space [2], exploring it for an array of thousands of modules is an infeasible proposition.

Our algorithm uses the randomized motion of regular holes in a lattice as primitive operations for 2D shape formation. These holes can be considered as quanta of negative volume (Figure 1). The creation of a hole, via the enclosure of empty space at the perimeter, increases the contour of the object (Figure 2). Conversely, the capture and deletion of a hole at a site on the perimeter reduces the contour at that point (Figure 3). To ensure that repeated creations and deletions do not create an unsuitable edge, we employ a smoothing technique derived from [3] as a subprocess. Planning for a macroscale shape transformation is then just a matter of designating regions of the perimeter as regions that create or delete holes, a process which depends only on the geometry of the source and target shapes, and not on the number of modules involved.

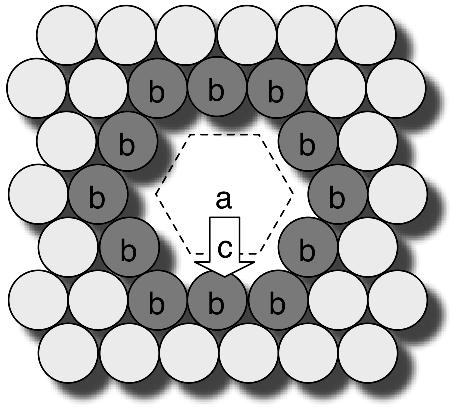

Fig. 1. Basic Layout: a) hole, b) shepherd group module, c) direction of motion

II. Related Work

The problem of reconfiguration in lattice-arrayed modular robots has received considerable attention in the literature. The intractably large state space of an array's potential motions has led to the development of many algorithms based on local decisions and control for reconfiguration. These algorithms may be broadly categorized into gradient-based [4]–[11] and graph-grammar [12]–[15] algorithms. Early gradient-based algorithms suffered from either a lack of expressivity [6], or tended to become stuck in state-space minima [4], [5]. Later work by Støy [7]–[9] allowed for the creation of arbitrary shapes, without the danger of getting "stuck" in minima. Additionally, gradient-based algorithms require broadcast flooding to disseminate gradient values, a bandwidth-intensive process involving roughly $\sqrt{n}$ hops for n modules. Graph-grammar based algorithms can be constructed without the need for broadcast flooding, but the number of states in the graph grammar is in many cases a linear function of the number of modules in the ensemble [14], [15]. In other grammar-based approaches the self-assembly process relies on the random addition of modules at the perimeter of the object [12], [13], an assumption that does not hold in ensembles with fixed numbers of modules.

Claytronics [16] provides inspiration for our work, postulating the existence and utility of massively scaled ensembles of modular robots for such applications as telepresence, remote manipulation, and programmable antennas. The work of Reshko [17] provides us with a means to establish and maintain a shared coordinate space between modules in the array, and allows us to assume that modules have current knowledge of their relative positions at any point in time.

III. Design

The ideal shape formation algorithm possesses several characteristics. The planning phase of the algorithm would be

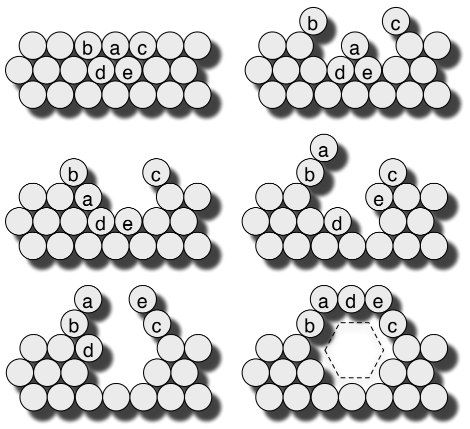

Fig. 2. Hole Creation (modules are labeled for illustrative purposes only)

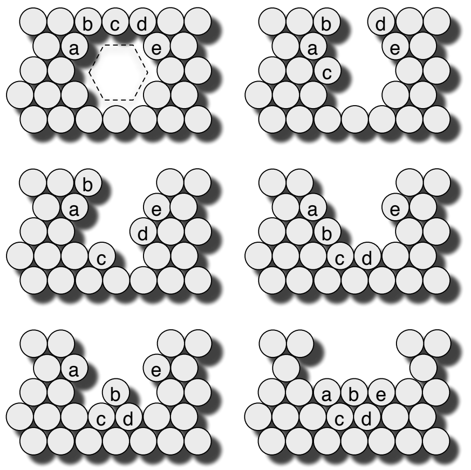

Fig. 3. Hole Deletion (modules are labeled for illustrative purposes only)

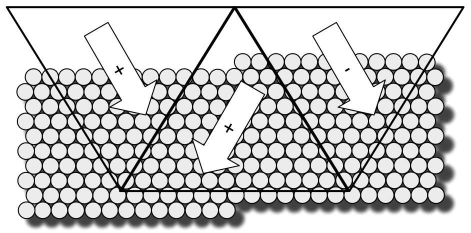

Fig. 4. Two growth tri-regions (left), one deletion tri-region, and their associated gravity directions

able to represent an arbitrary 2-dimensional shape, and the time required to generate the plan would not depend on the number of modules in the target ensemble. Once the algorithm began executing on the modules, it would not require global communication (either through a broadcast channel or hop-to-hop flooding), and would produce an exact rendering of the target geometry. Finally, the algorithm would not be susceptible to entrapment in local minima.

We note that in many modular robotic systems, motion constraints may prevent a single module from moving in desired directions. However, we note that when a sufficiently large void or empty space in a lattice is present, movement in the immediate neighborhood is always possible. Hence, by treating a canonical hole as first-class entity, and solving the problem of moving holes, we can avoid the motion constraints that plague single-module planning. We note that by inserting these holes into an ensemble, we increase total volume. Specifically, insertion of a hole at a surface causes the contour to expand at the point of insertion, as in Figure 2. Similarly, deletion or extraction of a hole at the surface causes the contour to lower, and the local density to increase, as in Figure 3. Randomly moving holes at all times ensures that the density of the entire array will converge to a uniform value. This will ensure a ready supply of holes for deletion. We conclude that the primitive operations of hole movement, creation, and deletion, provide the necessary capabilities for the formation and transformation of arbitrary shapes.

To implement this idea, there are several practical matter that must be addresses. First, we require a scalable planner for determining regions of growth and deletion. Second, we must ensure that conditions for hole creation and deletion are always present at the surface. In particular, hole deletion and creation increase surface roughness, which may preclude further operations on that surface. Techniques to maintain smoothness are needed. Finally, we will need to coordinate the overall growth or deletion to avoid topological changes (e.g. regions breaking free from the main mass). We have designed a set algorithms that allow arbitrary shape transformation while addressing these concerns.

IV. Algorithm Description

A. Layout and Definitions

Modules are arranged in a hexagonally packed array, with each module having 6 possible locations for neighbors. We denote the *neighbors* of a module k by $k[0] \cdots k[5]$. We define the set of modules containing k, its neighbors, and their neighbors as the *2-hop radius* of k.

We define a *hole* as the logical entity formed by the absence of a module h and its six neighbors $h[0] \cdots h[5]$. A hole is surrounded by a *shepherd group*, consisting of the 12 modules bordering the hole. The shepherd group modules share common state which identifies the presence of a hole, as well as its *direction of motion*, defined as one of the six cardinal directions in the hexagonal lattice. These concepts are illustrated in Figure 1.

We further define the *target geometry* as the desired final shape, represented in a scale-independent coordinate system. A *tri-region* is a triangular region in this same coordinate system, plus a *gravity direction* and a *growth/deletion flag* (Figure 4).

B. Basic Concepts

To plan a transformation from the source to the target geometry, the 2-dimensional coordinate space is tiled with equilateral triangles of a specific size. A triangle is the smallest unit of resolution that the planner can represent. Triangles that overlap regions in the source and target geometries where there is no change between the two representations are removed from consideration. Other triangles become growth or deletion tri-regions, based on whether the region they enclose is predominantly being filled or emptied during the transition from source to target geometry. These tri-regions are then transmitted to every module in the ensemble, and their location and state determine the roles that each module will assume.

In the "steady state" where no plan is being executed, holes inside the ensembles will move through the mass, colliding and reflecting off of each other and the surface of the ensemble, much like the molecules of an ideal gas. The presence of an active tri-region causes modules at the surface to begin producing or consuming holes. By producing a hole, the local contour of the surface is raised. Consuming a hole likewise lowers the local contour of the surface. These two operations alone are insufficient to effect macroscale shape change, as unrestricted deletion of holes can lead to a scenario where deletion sites that are closer to the center of mass preferentially bind to holes, starving deletion sites that are farther away (Figure 5). Additionally, creation and deletion create a rough surface, which must be smoothed to provide suitable locally flat sites for hole creation and deletion.

To solve the problem of holes preferentially being consumed near the center of mass (as opposed to the ends of contracting extremities), we introduce a technique inspired by simulated annealing [18]. Holes will not always bind to a deletion site, but rather will do so with a probability dependent on the distance between the site and the closest point on the target geometry's perimeter, modified by a decay value that increases binding probability with time. We refer to this technique as the *temperature test*, and it is described more fully in Section IV-E.

To solve the surface roughness issue, we introduce the notion of a gravity-driven collapse. In each tri-region, we define one of the lattice directions as "down", and all modules form virtual columns along that direction. Columns which are taller than their neighbors collapse, moving the top modules from a tall column to a shorter one nearby. This process, known as self-organized criticality [3], is analogous to the collapse of piles of sand or rice under gravity, and maintains a smooth surface for the ensemble.

C. Gravity and Tri-regions

We use the following algorithm to set the gravity direction for each tri-region:

- create a representation R of the source geometry
- create a worklist, add all tri-regions to it
- for each tri-region T in the worklist…
  - if no point in T is on the perimeter of R, place T on the end of the worklist and continue on to the next tri-region
  - for every point in T that is on the perimeter, find the directions with filled neighbors. Keep a running total for all points in T, and set T's gravity to the most common direction for filled neighbors (after 3-way moving average of the totals)
  - if T is a creation region, fill the region completely (in R)

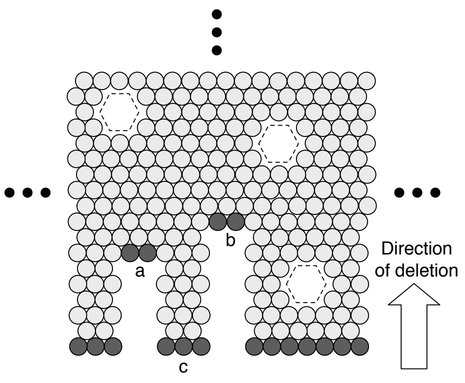

Fig. 5. Starvation occuring in the absence of smoothing. Regions a and b will prevent region c from activating.

  - else empty the region completely (in R)

Once gravity directions have been assigned to each tri-region, the tri-regions and the target geometry are distributed (either via broadcast or local flood) to each module in the ensemble. From this point on, each module acts independently.

At each timestep, the following actions occur:

- holes move
- creation/deletion regions activate
- smoothing occurs

D. Hole Movement

At each timestep, each hole attempts to shift the three modules at the leading edge of the shepherd group (designated by the direction of motion) from the leading to the trailing edge, thus shifting the entire hole one module in the direction of motion. If such a move would cause the hole to disrupt another hole's shepherd group, or to disrupt its own shepherd group (by moving outside the ensemble), then the move is not performed, and a new random direction is picked for the hole. Holes can ensure that a move is safe by reserving modules before movement, and respecting the reservation status established by other holes. As this motion requires the cooperation of a small number of nearby modules, it can be performed using local coordination,without any global synchronization. The same is true for growth, deletion, and smoothing.

E. Creation/deletion regions activate

If a module detects that it is on the perimeter (i.e. it has unfilled neighbors that are not part of a hole) and that it is inside a tri-region, it becomes a *growth or deletion node* (based on the growth/deletion flag of the relevant tri-region). If a module is a growth node, it checks to see if all of the following conditions hold:

- the node is inside the perimeter of the target geometry
- there are no holes in the nodes 2-hop radius
- there are exactly 11 other modules in the node's 2-hop radius

If all of these conditions are true, the growth node rearranges its 2-hop radius to create a shepherd group, and generates a

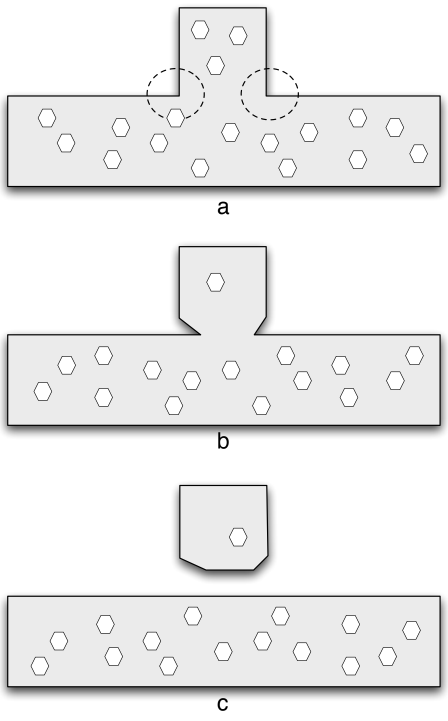

Fig. 6. Modules being abandoned during deletion of protruding element: a) initial configuration, with high binding probability sites circled, b) pinch-off beginning, c) region disconnected by pinch-off

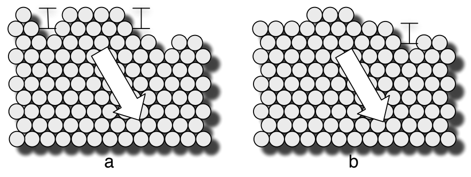

Fig. 7. Surface Smoothing: a) before smoothing b) after smoothing

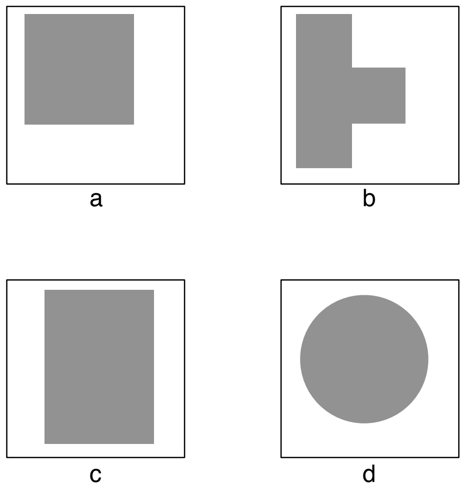

Fig. 8. Source and Target shapes: square,T,rectangle,and circle

new hole with a random direction of motion (Figure 2). The conditions for the activation of a deletion node are slightly more complicated, as we must counter the system's tendency to delete holes close to the center of the structure, potentially disconnecting small numbers of devices (Figure 6). A deletion node must satisfy the following conditions:

- the node is outside the perimeter of the target geometry
- exactly one hole is in the node's 2-hop radius
- the hole must contain at least one of the node's neighbors
- there are exactly 12 other modules in the 2-hop radius of that hole's center
- the node satisfies the *temperature test* (see below)
- the hole (if present) cannot move against the local tri-region's gravity

If all of these conditions are true, the deletion node destroys the hole, and lowers the contour of the object by using the portion of the hole's shepherd group closest to the perimeter to fill the hole (Figure 3).

If a hole is present, but the last condition was not met, then the hole's direction of motion is set to be against the local region's gravity. This forces holes to move along the surface, against gravity, which will encourage binding at the far extremities of deletion regions.

Temperature test: The probability of a hole being bound by a particular active region r is given by:

$$p = 1.0 + \log(\frac{d_{near}}{\max(d_{max} - c_{decay} \cdot t, 1)})$$

Where d_{near} is the distance (in diameters) between r and the closest point on the target geometry's perimeter, d_{max} is the maximum distance between the perimeters of the source and target geometries, c_{decay} is a time decay constant (typically $[15, 50]$), and t is the time elapsed (in unit timesteps). This provides a simple logarithmic decay function which favors hole binding at the extremities of deletion regions.

F. Smoothing

Every module which is a growth or deletion node also participates in the smoothing process (Figure 7), in order to maintain a smooth surface for additional growth or deletion. Each module on the perimeter is the head of a virtual column of modules that extends down into the ensemble along the direction of the local tri-region's gravity direction. Each *column-head* checks its neighbors at +120° and -60° from the gravity direction. If one of these neighbors is empty, the column-head begins a search downwards along the gravity direction, for up to 3 hops. If the number of hops from the column head to the head of a neighboring column exceeds 1, the column-head moves to become the head of the neighboring column instead (provided that this move will not disconnect any of its neighbors from the ensemble). This movement may create an imbalance in another column, requiring more smoothing. The process continues for a finite number of iterations, or until the system stabilizes.

V. Experimental Results

A. Experimental Setup

We implemented the algorithm in a Java-based event simulator, executing the algorithm via a single omnipotent controlling

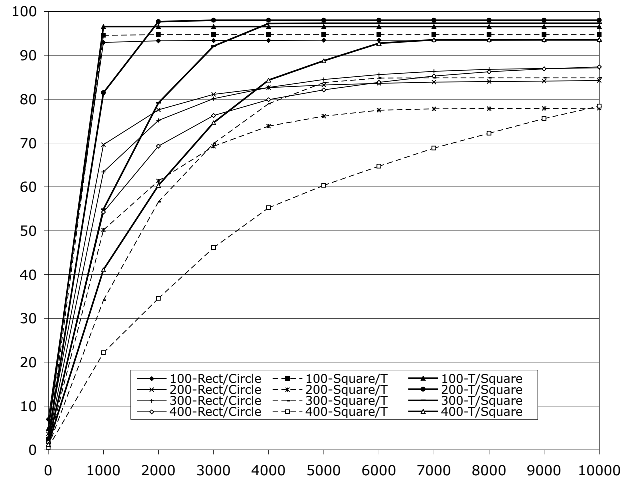

Fig. 9. Motion completeness of experiments over time

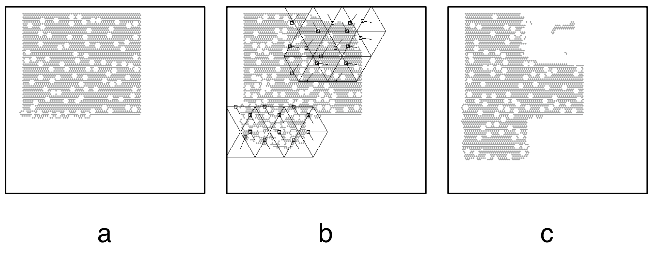

Fig. 10. Example frames from simulator

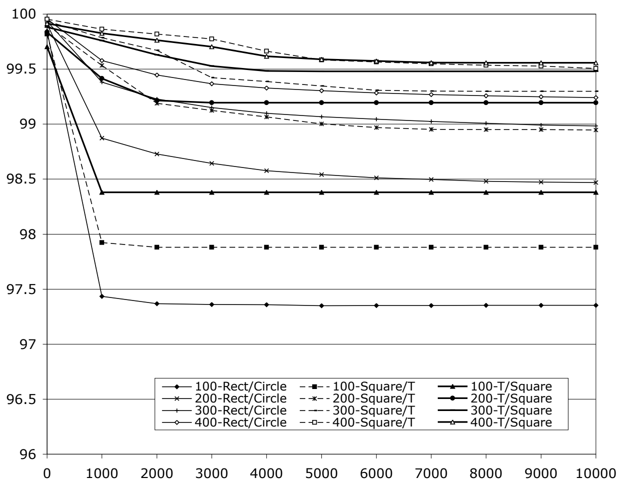

Fig. 11. Stability Maintenance of experiments over time

agency. The atomic unit of motion in the simulator was the transition of a module from one lattice point to an adjacent one. The simulator's inputs were the size of the array, the source and target geometries, and the resolution of the tri-regions. Note that no intermediate keyframes were generated; all motion planning was directly from the source to the target geometry. We selected four simple shapes (Figure 8) for the source and target geometries. From these shapes, we derived three plans: from a square to a T shape, from a T shape to a square, and from a rectangle to a circle. These three plans show respectively the creation of corners, the deletion of corners, and the creation of curvature. Each plan was executed on square arrays of 100, 200, 300, and 400 lattice points per side, yielding simulations with approximately 3,000-62,000 active modules (Table I). Each of these 12 experiments was repeated 10 times with random initial hole placement and start times, and the arithmetic mean of the results was reported. Standard deviations on all error measures were low (typically below 2.0). Simulations ran for 10,000 timesteps, where one timestep was sufficient time to perform hole creation or deletion, followed by smoothing.

During the simulation, three metrics were evaluated every timestep. The first metric, *motion completeness*, was the percentage of lattice points which need to be emptied or filled which had been successfully emptied or filled at that point in the simulation. The second metric, *stability maintenance*, was the fraction of lattice points designated to remain filled or empty that did so. The final metric, *shape compliance* was the fraction of lattice points that had the same status as the target geometry called for. In all cases, lattice points occupied by a hole were considered filled.

B. Completion Time & Correctness

Motion completeness over all 12 experiments showed several general trends (Figure 9). For a given shape transformation, experiments at larger scales approached their maximal motion completeness (m_{max}) more slowly than at smaller scales. This is expected behavior, as the number of modules which must be moved increases with the experiment's scale. Additionally, motion completeness for the square to T transformation was significantly lower than that of the other experiments at scales greater than 100x100. We attribute this to the increasing coarseness of the fixed number of tri-regions used in the experiments. As the scale increases, each tri-region controls a larger number of perimeter modules, leading to a decrease in the control of local curvature. This was especially apparent in the square to T case, as the decrease in control manifested itself as "islands" of modules that were abandoned during the removal of the square's upper-right corner (Figure 10).

Stability maintenance was consistently high, with values never dipping below 97.2% (Figure 11). As stability violations typically manifested themselves via one-module disturbances on the surface, larger scales (with larger stable volumes) exhibited higher levels of stability maintenance.

Finally, shape compliance in all experiments was uniformly high, with all configurations achieving at least 95% maximum compliance (c_{max}). As shape compliance is merely a weighted sum of motion completeness and stability maintenance, the factors which affected those metrics were visible to varying extents in the shape compliance results (Figure 12).

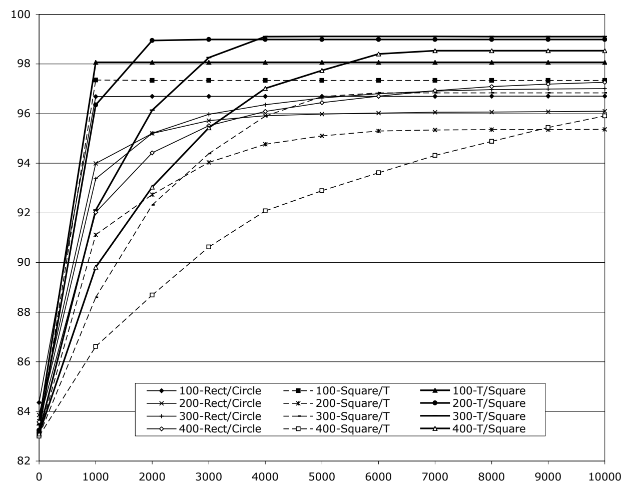

Fig. 12. Shape Compliance of experiments over time

TABLE I

SUMMARY OF EXPERIMENTAL RESULTS (MEAN OVER 10 RUNS)

Experiment	# modules	m_{max}	c_{max}
100x100 Square/T	3109.2	94.70	97.34
100x100 T/Square	3029.1	96.55	98.07
100x100 Rect/Circle	3916.2	93.45	96.70
200x200 Square/T	12331.1	77.92	95.36
200x200 T/Square	11929.5	98.0	98.99
200x200 Rect/Circle	15491.9	84.25	96.10
300x300 Square/T	27737.8	84.87	96.84
300x300 T/Square	26606.0	97.30	99.11
300x300 Rect/Circle	34805.5	87.14	97.00
400x400 Square/T	49169.7	78.41	95.91
400x400 T/Square	47194.4	93.57	98.54
400x400 Rect/Circle	61714.9	87.38	97.26

VI. FUTURE WORK

Several obvious extensions to this work present themselves. The first is the improvement of the planning algorithms to eliminate the occasional disconnection of modules during shape transformation. Additionally, there currently exists no provision for modifying the topological genus of the shape, therefore a method for creating and destroying macroscale holes in the ensemble is required. A fully distributed implementation of the algorithm would provide a more compelling case for the utility of the algorithm. Finally, the extension of the algorithm into 3-space would allow for the rendering of arbitrary volumes. This extension could use hexagonally close-packed modules, and holes that encompassed a module and its 12 kissing point neighbors.

VII. CONCLUSION

We have successfully designed and implemented a novel shape formation algorithm for ensembles of lattice-arrayed modular robots. The algorithm is massively parallel and fully distributed. Constructing a goal representation requires time proportional only to the complexity of the desired target geometry. Construction of the shape by the modules requires no global communication or broadcast floods after distribution of the target shape. Results in simulation show high shape compliance and stability maintenance. We believe that this algorithm represents an initial step in the creation of fully-scalable motion-planning algorithms for modular robotics.

REFERENCES

[1] D. Rus and G. Chirikjian, Eds., *Special Issue on Self-Reconfiguring Robots*. Autonomous Robotics, January 2001, vol. 10,1.

[2] I. Chen and J. Burdick, "Enumerating the nonisomorphic assembly configurations of modular robotic systems," in *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems '93, IROS '93*, 1993.

[3] V. Frette, K. Chistensen, A. Malthe-Srenssen, J. Feder, T. Jssang, and P. Meakin, "Avalanche dynamics in a pile of rice," *Nature*, vol. 379, pp. 49–52, January 1996.

[4] M. Yim, J. Lamping, E. Mao, and J. Chase, "Rhombic dodecahedron shape for self-assembling robots," Xerox PARC SPL, Tech. Rep. P9710777, 1997.

[5] M. Yim, Y. Zhang, J. Lamping, and E. Mao, "Distributed control for 3d metamorphosis," in *Autonomous Robots 10*, 2001.

[6] H. Bojinov, A. Casal, and T. Hogg, "Emergent structures in modular self-reconfigurable robots," in *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, 2000.

[7] K. Støy, "Emergent control of self-reconfigurable robots," Ph.D. dissertation, AdapTronics Group, The Maersk Mc-Kinney Moller Institute for Production Technology, University of Southern Denmark, 2003.

[8] ——, "Controlling self-reconfiguration using cellular automata and gradients," in *Proceedings of the 8th international conference on intelligent autonomous systems (IAS-8)*, March 2004, pp. 693–702.

[9] K. Støy and R. Nagpal, "Self-repair through scale independent self-reconfiguration," in *Proceedings of IEEE/RSJ International Conference on Robots and Systems, (IROS)*, 2004, pp. 2062–2067.

[10] R. Nagpal, "Programmable self-assembly: Constructing global shape using biologically-inspired local interactions and origami mathematics," Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, June 2001.

[11] ——, "Programmable pattern-formation and scale-independence," in *International Conference on Complex Systems (ICCS)*, June 2002.

[12] P. W. K. Rothemund and E. Winfree, "The program-size complexity of self-assembled squares," in *STOC*, 2000.

[13] D. Soloveichik and E. Winfree, "Complexity of self-assembled shapes," in *DNA Computers 10*, 2005.

[14] E. Klavins, "Directed self-assembly using graph grammars," in *Foundations of Nanoscience: Self Assembled Architectures and Devices*, Snowbird, UT, 2004.

[15] E. Klavins, R. Ghrist, and D. Lipsky, "Graph grammars for self-assembling robotic systems," in *Proceedings of the International Conference on Robotics and Automation*, 2004.

[16] S.Goldstein, J. Campbell, and T. Mowry, "Programmable matter," *IEEE Computer*, vol. 38, 6, pp. 99–101, May 2005.

[17] G. Reshko, "Synthetic reality: Communication and localization," Master's thesis, Carnegie Mellon University, August 2004.

[18] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science, Number 4598, 13 May 1983*, vol. 220, 4598, pp. 671–680, 1983.