

# Rhombic Dodecahedron Shape for Self-Assembling Robots

Mark Yim John Lamping Eric Mao J. Geoffrey Chase  
Xerox Palo Alto Research Center  
3333 Coyote Hill Rd, Palo Alto, CA 94304

## Abstract

*A self-assembling robotic system capable of approximating arbitrary three dimensional shapes utilizing repeated rhombic dodecahedron shaped modules is presented. This shape allows very simple rotational motions about its edges. Using this shape and the motion constraints it imposes, a distributed algorithm using localized information is created and simulated. The algorithm causes each module to move in an efficient manner without explicitly planning the entire motion of each module before hand. For most shapes the algorithm achieves full completion of the desired goal configuration. Example of a tea cup shape utilizing 441 modules with 100% completion is shown.*

## 1. Introduction

The basic idea of self assembling automata\* is to have a collection of connected robotic modules which act together to perform a given task. Employing many small modules enables the self-assembly of an approximation of arbitrary three dimensional shapes from an arbitrary starting shape.

Self assembling automata can be used in a wide variety of applications including: locomotion over a variety of terrains for payload delivery, or inspection, or exploration; dynamically forming structures e.g. bridges, walls, chairs; 3D visualization through forming arbitrary shapes; embossing through forming the print head of stamping machines and many others.

These robotic systems contain many of the properties of metamorphic robots as defined in Chirikjian [1996]. Specifically, the systems presented here obey the following rules:

- 1) The robot is made of one type of repeated module.
- 2) The modules are packed so they fill space with minimal gaps.

---

\* Self-assembling automata has had several names in the past including: metamorphosing robots, polymorphic robots, shape changing robots, utility fog [Hall, 1996], and morphing structures

- 3) The robots achieve structural change by modules maneuvering around each other.
- 4) Each module contains a mechanism allowing for communication and transfer of power between adjacent modules, defining a robot whole to be all the modules in one connected component.

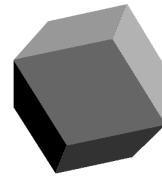


Figure 1: A rhombic dodecahedron

This paper explores two primary issues in realizing self-assembling robots; (1) the shape and motion of modules for assembling three dimensional shapes and (2) development of a decentralized motion planning algorithm. The rhombic dodecahedral shape is shown to simplify the motions of the modules while satisfying the defined system requirements. An ordered nearest goal method is demonstrated as an effective localized control mechanism in which inter-module communication is minimized.

Three different research groups in this field have developed modular reconfigurable robots. Chirikjian [1994, 1996] and Murata et. al [1994] have separately simulated and built planar hexagonal robots which “roll” around each other. Neither addressed the generalized shape planning issue necessary for the decentralized self-assembly of arbitrary three-dimensional structural shapes. Yim [1993, 1994, 1994a] developed a bi-unit-modular robot which studied locomotion and was designed to attach and detach. However, the robots did not exploit reconfiguration as a means of motion. Paredis and Khosla [1993], and Fukuda [1988] looked at modular reconfigurable systems however, those systems involved many types of modules.

The following section address the issues of module shape and the advantages of that shape. Subsequent sections address motion constraints that are imposed by the shape, planning using those motion constraints and finally

conclusions and future work including some hardware implementation details.

## 2. Rhombic Dodecahedron Kinematics

For three dimensional modular robots, the shape of the unit cell is a much more important design decision than might at first appear. For example, the obvious cubical shape turns out to have significant disadvantages. This work shows that a rhombic dodecahedral shaped (12 faces, each a rhombus as in Figure 1, modules work much better.

### 2.1 Motion

One of the crucial problems for a self-assembling robot is individual module motion. For small robots, rolling/rotational motions are preferable to sliding motions because they can have less friction, which becomes very important as systems scale down in size.

Using the cube as an example, rotational motion can also lead to significant problems. First, there are two different kinds of rotational motion for cubes, as illustrated in Figure 2. Both motions need to be supported, or the behavior of the system will be severely limited. However, designing a rotation mechanism that supports both of these motions is difficult. The large range of motion in the first rotation of Figure 2 is hard to achieve. The rotation on an axis shared with two other cubes in the second rotation of Figure 2 is also difficult. Having to support two different rotations, (90°, 180°) compounds these difficulties.



Figure 2: The two kinds of rolling motions for cubes

These problems are inherent in the way cubes are packed and occur with any group of simple-cubic packed modules, as illustrated in Figure 3. A review of packed structures is shown in Figure 4.



Figure 3: Simple cubic packed sphere rolling motions.

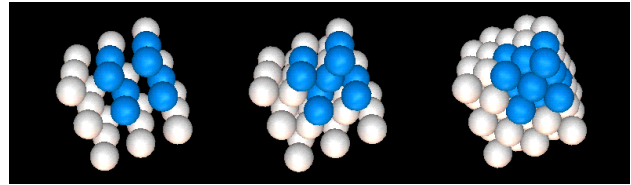


Figure 4: a) simple cubic packing, b) body centered cubic packing, c) face centered cubic packing

Finally, moving packed cube-shaped modules by rotation doesn't eliminate sliding, with the concomitant difficulties of overcoming friction and of designing sliding inter-module connections. This difficulty is illustrated in Figure 5, where the indicated rotation causes the moving cube to slide against the two cubes in back.

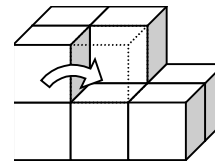


Figure 5: Sliding of edges during rotation

Rhombic dodecahedra (RD) are space filling isohedra that, in many ways, are the three-dimensional analog of hexagons. RD's eliminate the difficulties encountered with cubic modules and simple cubic packing.

Like hexagons, they require only a single simple rotational motion as illustrated in Figure 6. The rotations about any edge of an RD from one packing position to another is always exactly 120 degrees. Again, these properties actually stem from the way RD's pack: face-centered cubic (see Figure 4c).

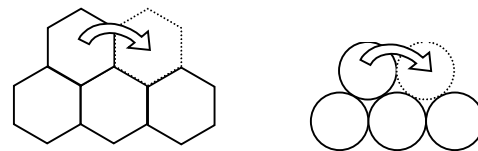


Figure 6: 2D projection of rhombic dodecahedron rotation (hexagon) and face-centered packed spheres rotation

The reason for this simplicity is that, when packed, every edge is formed by the adjacency of at most three RD's. When there are three RD's around one edge, no motion can occur about that edge - the 360 degree space is filled. When there is one RD, no motion can occur as there must be another RD to move around. Motion can only occur when there are exactly two RD's. This uniformity of the edge relationships is the key to the simplicity of the design of a mechanism that allows rotations about edges. In addition, rotations can be performed without requiring any

edges or faces to slide against each other, eliminating concerns with friction and designing sliding connectors.

## 2.2 Preserved Good Properties of Cubic Packing

While simplifying the mechanics of rotation, the face-centered cubic packing preserves many of the desirable features of simple cubic packing. First, when two faces of two identical RD's are aligned and pressed together (mated) as in Figure 7, all rotations about one of the 4 shared edges will result in another set of faces aligning. All subsequent rotations will result in the same. This property is also true for cubes and all regular polyhedra, however it is not true for all isohedra (polygons with identical sides).

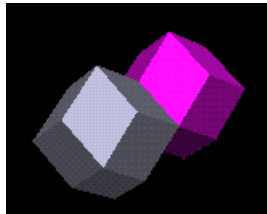


Figure 7: Two RD's with two faces mated.

This isohedral property also simplifies manufacture. One advantage to repeating modules is the ability to batch fabricate, minimizing the unit cost. In addition, each module can be made up of repeating parts. For example, each polyhedron is made up of faces, each face made of edges and each edge made of vertices. If the polyhedron is an isohedron with uniform edges like the rhombic dodecahedron, then batch fabrication can be applied at the face level and edge level as well.

## 2.3 Motion Constraints

There are three basic constraints for motion of a module:

1. There must be a parent RD with a shared edge to roll about.
2. The motion of the module must not collide with any other module.
3. The entire group must be contiguous during and after a motion.

The first two constraints are clear from basic physical laws. The third constraint is imposed on the system assuming one power source supplying power to all modules through neighboring modules.

The second constraint leads to "blocking constraints" as modules may block the motion of another module. At one extreme, if an RD is entirely surrounded by modules, that

is, there are 12 neighboring RD's, one on each face, then that module cannot move without colliding with one of its neighbors. At the other extreme, if there are only two modules as in Figure 7, then the motion of one rolling over the other is not blocked by any other module. The question which governs motion planning can be posed as: for any given edge of an RD, what neighboring modules will block the rotation of the RD about that edge? If a neighbor attached to a face blocks the motion about an edge, that face is called a blocking face for rotations about that edge.

Take the plane formed by the edge and the center of the RD and divide the RD into two. Five faces will lie entirely on one side, five on the other and two faces will be divided in half. The five faces can be the ones which appear in Figure 1. The two split faces would be perpendicular to the page on the left and right side of the RD in Figure 1, and are blocking faces about the splitting edge. In addition, depending on the sense of the rotation, the five faces on one side are also blocking faces. This result is called the 7-side blocking constraint.

Unfortunately, this constraint is too strict if RD's are used to assemble thick shapes. If six RD's are brought together at a common vertex, as shown in Figure 8, all RD's block the motion of all other RD's on all contacting edges. The group is immobile - no modules can move out. Since the blocking constraint is symmetric, the converse is also the case: modules cannot move into this configuration. This configuration is encompassed within many other possible configurations, most notably any solid configuration with a minimal thickness greater than 3 RD's. Therefore, this result greatly restricts the set of configurations that can be built by the pure rhombic dodecahedral shape.

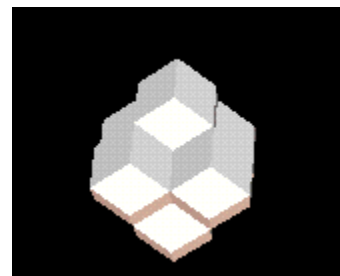


Figure 8: Six RD's sharing one vertex (three on top, three on bottom) in immobile state.

Modifying the modules' shape slightly so that the 7-side constraint can be reduced to a 5-side constraint would solve the immobility problem. Specifically, the two faces which are split by the bisecting plane would no longer be blocking faces. With the 5-side constraint, there are no configurations which are immobile.

This modification requires a vertex to move 14% closer to the center of the RD. The four edges connected to it must also move. One modification that will result in the 5-side constraint is to have the rhombic faces hinged about the short axes, and the face reduced slightly as shown in Figure 9a. When a potential collision arises, due to the 7-side vs. 5-side constraint difference, the four triangular halves of the four faces which form the interfering vertex move inward as shown in Figure 9b. Movable vertices allow the maintenance of rolling edges while removing the limiting constraints.

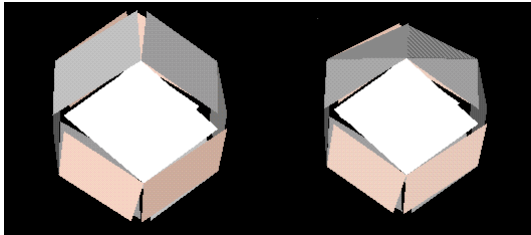


Figure 9: a) Modified RD b) top vertex actuated.

Each full  $120^\circ$  rotation about an edge is treated as a discrete motion. The motion constraints determine which discrete motions of a given configuration are possible. Thus, once these motions are defined, the next step is to find the sequence of these discrete motions that will result in the assembly of the goal configuration.

### 3. Planning

Another primary issue in making a viable self-assembling system is the distributed, yet coordinated, control of the many modules. One way to pose the self-assembly problem follows:

*Given  $n$  modules in an initial configuration, how can they move to form a new goal configuration?*

This problem is similar to the multi-robot path planning problem as presented by Latombe [1991], Mataric [1991], Caloud [1990], Li et al [1997]. As shown in these papers, optimal planning of the motion of  $n$  robots, or in this case modules, moving at the same time from one configuration to another is intractable, as the search space is exponential in  $n$ .

The self-assembly problem differs in two ways from the classical multi-robot path planning problem:

- 1) With self-assembly, the robots must not only avoid collisions with each other, they must also maintain contact with each other.
- 2) Since the modules are homogeneous, each module can be used at any arbitrary position in the goal

configuration. Thus, each robot has a large set of possible goal locations.

There are many approaches to solving the self-assembly problem. An approach that doesn't require global planning is preferable in order to utilize the limited, distributed computing resources in the modules. The assumption is that each module has enough computational resources to represent a complete configuration, but not enough to work out a complete plan. Hence, a decentralized approach is presented.

### 3.1 Example Approach

To illustrate some of the issues in planning for self-assembling robots, consider a very simple algorithm. Move all the modules at the same time toward one location within the goal configuration. Once that goal location is filled, the remaining modules move to the next goal location in a sequence that fills the goal configuration. One advantage of this approach is that with all the modules moving toward the goal, even if the closest RD is some distance away from the goal location, the other RD's will form a bridge toward the goal location. It is also very likely that a neighboring location to the goal will be filled. The problem is overcrowding. There may be too many neighboring locations filled, blocking the motion of an RD into the goal location. What results is a mass of RD's surrounding an empty location with each RD adjacent to the goal location blocking the motion of other adjacent modules.

This overcrowding illustrates an important constraint in describing the reachable space. In addition to having an RD to roll about, a location must also have at least five faces open, corresponding to the 5-side constraint (one face and the four faces sharing one if its edges.)

### 3.2 The Ordered Nearest Goal Method

In this method, each module moves toward the closest available goal while receiving information from adjacent modules about the status of that goal. Certain constraints are placed on the order in which goals are filled. These constraints can be determined by the individual modules as a function of the goal locations. Each module chooses a movement direction based on a Euclidean distance heuristic, but places higher preference on locations with two or more neighbors. The information communicated per time step is independent of the number of modules.

### 3.2.1 Information Storage and Propagation

Each module starts with a representation of all the goal locations and its current location. It adds to information about the current state of each goal, such as whether it knows the goal to be filled. Each module also dynamically stores its own state; the number and location of its current neighbors.

Information propagation is limited to information important to a module by limiting it to queries about a neighbor's state information. Examples include asking for information about a specific goal, or asking about a neighbor's neighbors.

### 3.2.2 Goal Ordering Constraint

To ensure that all goals are able to be filled, the algorithm constrains the order in which goals are filled. The basic idea is to sweep a plane across the target configuration, filling goal locations as the plane moves past; actually, some regions get somewhat ahead of others. Each module only needs to know the preferred direction of the plane.

First define a coordinate system on the modules. For face-centered cubic packing, the modules can be centered on a 3D checkerboard grid: defined to be every integer location on a 3 dimensional grid for which the sum of the coordinates  $x$ ,  $y$ , and  $z$  is even.

For any goal location  $G$ , define:

$$H(G) = y + z,$$

where  $y$  and  $z$  are the  $y$ - and  $z$ -coordinates of  $G$ . This function defines the direction of the plane.

Define two locations to be vertex-adjacent if they share at least one vertex. Then,  $G$  is goal-constrained if, for some vertex-adjacent goal  $N$  of  $G$ ,  $N$  is unfilled and  $H(N) < H(G)$ . In other words, goals near  $G$  with lower  $H$ -values must be filled first.  $G$  is admissible if it is not goal-constrained.

The reason vertex-adjacent is used here instead of just face-adjacent (sharing a face) is that it is possible that interference may occur between two RD's that are vertex-adjacent and not face adjacent.

Note that under this constraint, at most 7 neighbors of  $G$  are filled before  $G$  is filled. This result corresponds to the physical constraint, where a module at  $G$  can have up to 7 neighbors and still be free to move using the 5-side constraint discussed previously.

### 3.2.3 Reservation Constraint

A module at location  $L$  and moving toward goal  $G$  can reserve a goal if:

1.  $L$  is adjacent to  $G$  and
2.  $H(L) > H(G)$

If a module satisfies these two criteria, it records the time that it reserved the goal. The module then keeps this reservation for 100 turns, or until it communicates with a neighbor that reserved the same goal at an earlier time; after it moves adjacent, it queries it and communicates this state. If any of its neighbors is moving towards the same goal, the neighbor records that the goal has been reserved and is forced to choose a different goal. This reservation information is then propagated from these neighbors to other modules moving toward this goal. By forcing other modules to choose other goals, the module making the reservation should have more room to move and be able to fill its goal.

### 3.2.4 Assistance

A goal  $G$  will generally have a neighboring goal  $P$  such that  $H(P) < H(G)$ . Under the ordering constraint,  $P$  will be filled before  $G$ , so that when a module reserves  $G$ , the module uses  $P$  as its parent, rolling over  $P$  to reach  $G$ . In the case where such a parent  $P$  does not exist, one of two actions occurs. If  $G$  must be filled before all of its neighboring goals under the ordering constraint, then  $G$  cannot be reserved. Otherwise, when a module reserves  $G$ , it raises an assistance-required flag. If a neighboring module is also moving toward  $G$ , then instead of choosing another goal, it assists the reserving module by moving to a neighboring location  $L$  satisfying  $H(L) < H(G)$ . The reserving module can lower the assistance-required flag and roll over the assistant to reach its goal.

### 3.2.5 Decision Making

Based on what a module knows about the various goals, it can choose the closest unconstrained goal and begin moving toward it. When a module reaches its desired goal location, it stops moving. This ensures that the admissibility of goals is never overestimated, i.e. a module may consider a goal to be constrained when it is not, but the reverse never occurs. This also ensures that once a goal is admissible, it will never become inadmissible. Information about the filling of goals is also never overestimated.

Sometimes, a module will have no goals to move toward, because all the admissible goals are either filled or

reserved. It then moves towards a default goal  $D$ , which can be calculated from the goal configuration by:

$$D = M + (0,4,4)$$

where  $M$  is the goal location with maximum  $H(M)$  and the ordered triplet corresponds to the  $(x,y,z)$  coordinates.

Note that for all goals  $G$ ,  $H(D) \geq H(G) + 8$ . This helps to relieve overcrowding situations by allowing modules with nowhere to go to move away so that the goals can be filled by the modules which reserved them.

At each location, modules choose a direction to move toward the goal by querying about a neighbors' neighbors to determine all possible neighboring reachable locations that it can move to. It then ranks these locations by the following criterion:

1. a location is the desired goal; highest priority
2. a location has at least one filled goal as neighbor; medium priority
3. a location has at least two neighboring modules; medium priority
4. none of the above; lowest priority

Euclidean distance is used as a secondary criterion, with locations closest to the desired goal favored over other locations. Ties are broken randomly. The modules tend to stay together rather than spread out in long chains toward goals due to the second and third criteria. The lack of long chains results in a significant decrease in the number of local minima for the Euclidean distance heuristic. To escape from the few local minima that do occur, modules use simulated annealing, where they occasionally disregard the priority rules and choose a less favorable location.

### 3.3 Complexity Analysis

Information propagation is limited to only the information most important to the module. A module can only query its neighbors about the following:

1. Has my goal been filled already?
2. Has another module reserved my goal?
3. Do you need assistance? Are you assisting me?
4. Are you filling a goal that I did not know was filled?

If a module discovers any information from these queries, it records the discovery in its own set of state information. This process allows information to be gradually propagated to all the modules. If a module discovers that a particular goal  $G$  is being filled using query (4), it can also determine under the ordering constraints which other goals must have been filled before allowing  $G$  to be admissible.

The entire algorithm is processed in time steps. A time step is defined to be the time a module spends doing one or more of the following:

- Choosing among what goals it thinks are admissible
- Querying neighboring modules
- Rolling around a neighboring module.

The following complexity analysis uses this definition of time step. Since information is severely limited, and the movement heuristic is very low-cost, the time complexity, per module follows:

Information communication:  $O(1)$  per time step

Computation time:  $O(g)$  where  $g$  is the number of goals

Memory requirement:  $O(g)$

In the current implementation, the worst case computation time is due to periodically searching for the closest goal. With a precomputation scheme it is expected that this should reduce to  $O(\log(g))$  or  $O(1)$  with perhaps a modest increase in the information communication and/or the memory requirement.

These calculations assume a hardware solution for determining whether the continuity motion constraint will be broken, for which no efficient software solution has yet been found.

### 3.4 Algorithm Verification

The ordered nearest goal planner is simulated for several test cases. These problems start in initially square configurations of varying size, with goal configurations of varying complexity, including: a planar "X" with 54 RD's, a tea cup with 441 RD's and a hollow sphere with 625 RD's.

Figure 10c shows three steps in the assembly from a square planar shape (10a) to a teacup shape (10c). This process took 1808 time-steps to complete inside a 64 x 64 x 64 workspace grid. In this figure, RD's which have reached a goal location are shown in solid colors, others are shown in wire frame.

For some shapes 100% completion was not reached. The hollow sphere would sometimes end with RD's trapped on the inside unable to reach locations on the outside. This situation varied depending on the thickness of the sphere wall. Successful completion of a hollow sphere with 625 RD's takes 3442 time-steps.

## 4. Conclusions and Future Work

The rhombic dodecahedron shape is shown to be a viable shape for self-assembling robotic systems capable of approximating arbitrary three dimensional shapes. A parallelized algorithm to control the motion of these modules efficiently using the limited computing on each module is presented. A tea cup shape with 441 modules and 100% completion is shown.

At the time of this writing, hardware prototypes are being constructed. Each module will have 48 shape memory alloy actuators, 24 for the edge motion rotations, and 24 in 6 groups of 4 for the dynamic shape modification to obtain the 5-side blocking constraint.

**Acknowledgments:** This work was funded in part by the Defense Advanced Research Projects Agency (DARPA). Contract #DABT630095C-025.

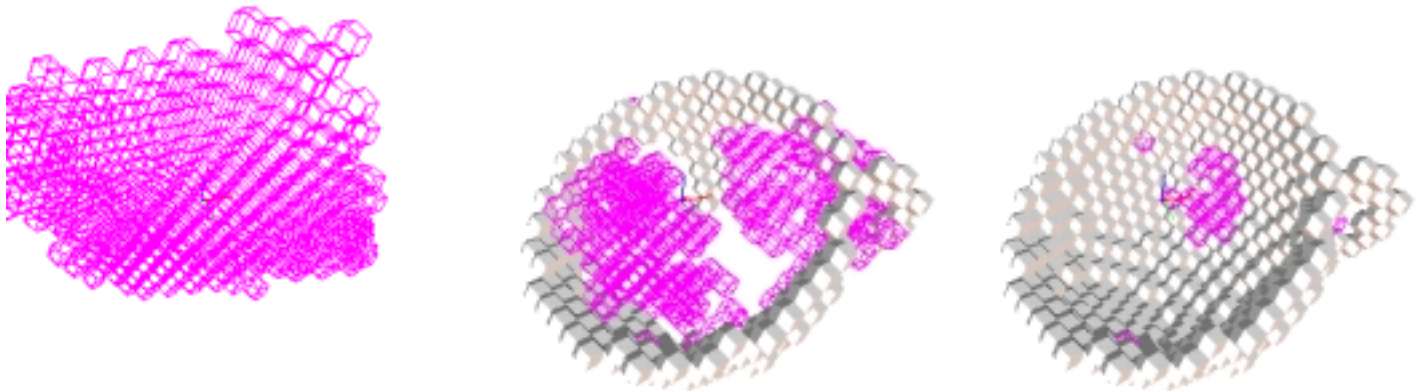


Figure 10a,b,c: Self assembly - transforming from a square plate to a teacup shape

## 5. Reference

Caloud, P. et al.[1990] "Indoor Automation with Many Mobile Robots" *Proceedings of IEEE, Intl. Workshop on Intelligent Robots and Systems '90 (IROS)* pp. 67-72.

Chen, I.-M., Burdick, J.W., [1993] "Enumerating Non-Isomorphic Assembly Configurations of a Modular Robotic System." *Proc. of IEEE/RSJ Intl. Conf. On Intelligent Robots and Systems (IROS)*, Yokohama, Japan pp 1985-1992.

Chirikjian, G.S, [1994] "Kinematics of a Metamorphic Robotic System," *Proc. of the 1994 IEEE Intl. Conf. on Robotics and Automation*, San Diego, CA, pp. 449-455.

Chirikjian, G.S., [1996] "Evaluating Efficiency of Self-Reconfiguration in a Class of Modular Robots," *Journal of Robotic Systems*, vol.13, no.5, pp. 317-38.

Fukuda, T., and Nakagawa, S., [1988] "Dynamically Reconfigurable Robotics System," *Proc. of Intl. Conf. on Robotics and Automation*, pp. 1581-1586.

Hall, J. Storrs, [1996] "Utility Fog: The stuff that Dreams are Made of", *Nanotechnology, Molecular Speculations on Global Abundance*, Ed. Crandall, B.C., MIT Press, pp.161-184.

Latombe, J.-C [1991], *Robot Motion Planning*,  
Dordrecht, Netherlands, Kluwer, 1991.

Li, T.-Y., et. al. [1997] "On-line Manipulation Planning  
for Two Robot Arms in a Dynamic Environment" *Intl. J.  
of Robotics Research*, Vol. 16, no.2 pp. 144-67.

Murata, S., Kurokawa, H., Kokaji, S., [1994], "Self-  
Assembling Machine," *Proc. of the 1994 IEEE Intl.  
Conf. on Robotics and Automation*, San Diego, CA,  
pp.441-448.

Mataric, M.J., [1991] "Parallel, Decentralized Spatial  
Mapping for Robot Navigation and Path Planning"  
Dortmund, West Germany , *Parallel Problem Solving  
from Nature. 1<sup>st</sup> Workshop*, Proceedings, pp. 381-6.

Paredis, C.J.J., Khosla, P.K., [1993] "An Approach for  
Mapping Kinematic Task Specifications into a  
Manipulator Design," in *Proc. of 5<sup>th</sup> Intl. Conf. on  
Advanced Robotics*, Pisa, Italy Vol 1, pp.556-561.

Yim, M.,[1993], "A Reconfigurable Modular Robot with  
Many Modes of Locomotion," *Proc. of JSME Intl. Conf.  
on Advanced Mechatronics*, pp. 283-288.

Yim, M.,[1994] "New Locomotion Gaits," *Proc. of the  
1994 IEEE Intl. Conf. on Robotics and Automation*, San  
Diego, CA, pp. 2508-2515

Yim, M., [1994b] "Locomotion with a Unit-Modular  
Reconfigurable Robot," PhD. Thesis, Dept. of  
Mechanical Engineering, Stanford University.