Algorithms and Mechanisms for Locomotion and Reconfiguration of Self-Reconfigurable Robots

Technical Report UT-CS-11-680

Kristy S. Van Hornweder*

Department of Electrical Engineering & Computer Science
University of Tennessee, Knoxville
kvanhorn@utk.edu

September 27, 2011

Abstract

There are numerous algorithms and methods for controlling the locomotion, assembly, and reconfiguration of modular self-reconfigurable robots. A brief overview of several of these algorithms is presented in this report. The general idea behind each algorithm is discussed, and these discussions are accompanied by images of example robot structures and configurations. The presentation also includes a discussion of results of simulation experiments, as well as issues and challenges with implementing these algorithms, and possible practical applications of the various research projects.

1 Introduction

Locomotion, assembly, and reconfiguration are crucial behaviors for self-reconfigurable robots in performing their tasks in their given environments. Several algorithms for implementing these behaviors in modular self-reconfigurable robots have been developed and tested, and several of these algorithms are presented in this report. The basic mechanisms and features of the algorithms are discussed, and several examples of robot structures and tasks are illustrated. The analysis of the algorithms also addresses issues such as scalability and future work involved in further developing these algorithms to be more amenable to real-life situations and tasks.

2 Algorithms for Locomotion and Reconfiguration of Self-Reconfigurable Robots

A number of algorithms and mechanisms for self-reconfigurable robot locomotion and reconfiguration have been reported in the literature. The following is a presentation of several of these algorithms.

2.1 Compressible Unit Modules and the Melt-Grow Algorithm

One type of mechanism for robot module relocation is an inchworm-like propagation [15] similar to the movement of muscles and amoeba. The modules can expand or contract as needed to achieve the desired configuration. This method has been applied to simulations of the Crystalline robot modules [16]. Figure 1 shows an example of this type of movement. The two center modules are compressed and then expanded in the direction of the goal position, which results in the entire structure moving forward one unit. Another example is shown in Figure 2. The module on the surface of the side face of the cube is pulled into the structure when two internal modules compress, and another module is popped out on the top face of the entire cube. In this way, a module can be repositioned on any convex structure in constant time. For concave portions of module arrangements, this method can be repeated for each turn (i.e., around each corner) in the structure.

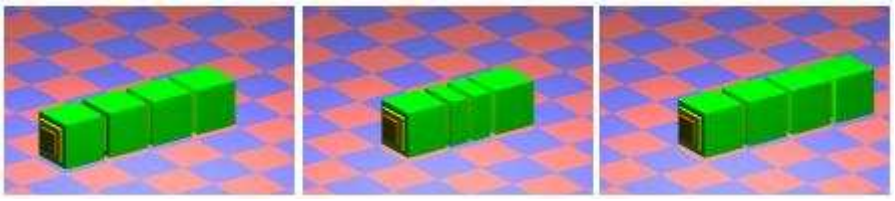

Figure 1. Inchworm locomotion of Crystalline modules [15].

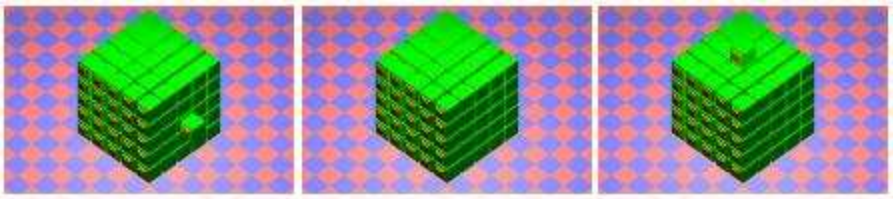

Figure 2. Moving a module along a cube surface [15].

The Melt-Grow planner [15] is used to establish shape metamorphosis of Crystalline modules. The algorithm is comprised of two basic steps:

1. Melt starting state S into intermediate state I
2. Grow goal state G from I.

The idea is to find a mobile grain (i.e., cube of modules) in S and move it to a place in I, and repeat this process until all grains are in I. A grain is mobile if it can be moved without disconnecting the entire structure. Then for each mobile grain in I, the grain is moved to a place in G until all grains are in G. A reconfiguration example using the Melt-Grow algorithm includes converting a table into a chair [15]. Another simulation of 3D Crystalline robots involves transforming a dog-shaped object into a couch-shaped object [15, 16].

2.2 Distributed Control for 3D Metamorphosis

The distributed control algorithms for reconfiguration presented in [20] were applied to simulations of Proteo modules. Proteo is a type of metamorphic robot comprised of homogenous modules (e.g., rhombic dodecahedron shape as shown in Figure 3) that occupy cells of a lattice structure [20]. Modules move in discrete steps by rolling over one another into an open neighboring space. A module determines a move based on the current cell, the current state of the module, and the states of the neighboring modules obtained from local communication between modules. The algorithm is based on a goal-ordering mechanism. A partial order of goal cells in the final configuration is computed, and this determines the order in which the goal spaces are filled. This allows a module to stay in place once it reaches a goal space, and it minimizes communication by enabling local reasoning about the global structure. Issues with these distributed control algorithms include stability, local minima, and overcrowding of modules.

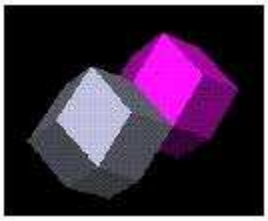

Figure 3. Rhombic dodecahedron shape of Proteo module [20].

There are three types of distributed control discussed in the research of [20]: distance-based method, heat-based method, and the combined method. In the distance-based approach, open

goal sites are ordered based on the Euclidean distance to unconstrained (i.e., unblocked) unfilled goals. In the heat-based algorithm, unconstrained unfilled goals are treated as heat sources, while modules that have not reached a goal place are considered heat sinks. Each module propagates heat to its neighbors. Modules that are closer to unconstrained unfilled goals have a higher temperature than the farther away modules, thus the closer modules will fill the goal sites first. The combined method begins with using the distance method, and switches to the heat mechanism when the system appears to be stuck. Each of the three mechanisms were applied to configurations that resemble a flat disk, solid ball, hollow ball, and a cup structure, as shown in Figure 4. The simulations involved up to 450 modules. The convergence time is approximately linear in the number of modules.

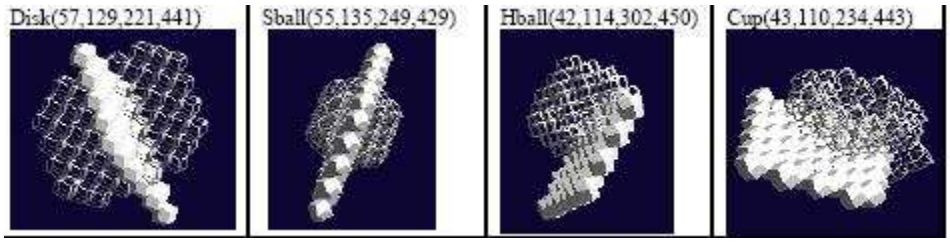

Figure 4. Simulations of flat disk, solid ball, hollow ball, and cup [20].

2.3 Emergent Structures in Modular Self-reconfigurable Robots

Control algorithms for producing emergent structures were applied to simulations of the Proteo metamorphic robot system [3]. The modules are homogenous and have a rhombic dodecahedron shape. Rather than achieving an exact goal configuration, it is only necessary to create a structure with the appropriate properties to accomplish the given task. This is more amenable to uncertain environments. In addition, finding an optimal solution in a large search space of exact configurations is very difficult. In this way, emergent structures are created, similar to techniques implemented in the field of Artificial Life. The algorithms are based on local rules, where modules only communicate with their immediate neighbors. The mechanisms in this work are useful for forming complex connected structures, object manipulation applications, and dynamic adaptation. The algorithms can easily be scaled up in the number of modules and down in the size of the modules. Features of these algorithms are growth (creating structures), seeds (for beginning growth), and scents, which serve as the communication between modules. There is a scent gradient throughout the structure, and these scents indicate the distance between a module and a scent-emitting module. The mode of a module is the current finite state machine (FSM) state, which can be one of SLEEP, SEED, SEARCH, FINAL, and NODE (for branching). The seeds function as local attractors and the scents serve as global gradients. By varying the number and combination of seeds and scents, different structures can be created. Several different structures are briefly presented below.

Chain: These linear structures are useful for climbing stairs or traversing holes or narrow spaces. The control rules for this system [3] are given below and Figure 5 illustrates modules forming a chain.

* If in SLEEP mode, if a scent is detected, go to SEARCH mode.
* If in SEARCH mode, propagate scent and move along scent gradient.
* If in SEED mode, emit scent, and if a module has appeared in the direction of growth, set that module to SEED mode, and go to FINAL mode.
* If in FINAL mode, propagate scent.

Figure 5. Formation of chain structure [3].

Branching: Can be used as an artificial hand, or for manipulation and locomotion tasks. Limbs are grown at several levels. The regular scent grows the structure and the node scent determines the distance to the nearest node. An example branching structure is shown in Figure 6.

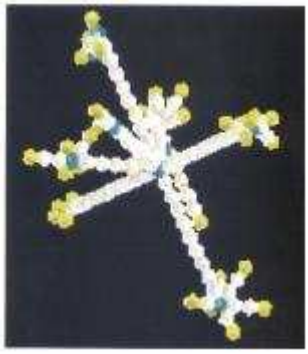

Figure 6. Branching structure [3].

Sponge: Can function as a supporting structure, squeeze through narrow spaces, or push walls apart. Figure 7 depicts an example of a sponge structure.

Figure 7. Sponge structure [3].

Regular lattice: Useful for structural support or scaffolding for disaster relief efforts. There are three possible directions for growth. The value of the node scent parameter can be varied to

create a more or less dense structure. An example lattice structure is given in Figure 8.

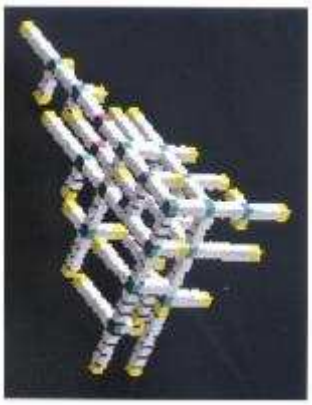

Figure 8. Lattice structure [3].

Grasping an object: The algorithm will function properly if the object has an unknown shape, size, or location. Two types of seeds are utilized: one for reaching the object, and the other for growing around the object (i.e., grasping). Figure 9 shows an example that resembles a hand grasping a ball-like structure.

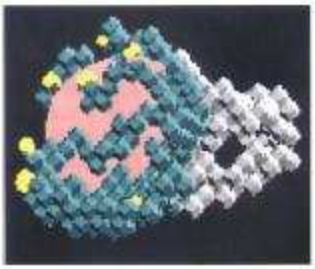

Figure 9. Hand-like object grasping a ball-like object [3].

Dynamically adapting to external forces: For example, a table that repositions its legs based on the location of the weight on the table surface. Figure 10 displays three examples of a table adjusting the placement of its legs according to the position of the weight indicated by the arrow.

Figure 10. Table adjusting its legs according to position of weight [3].

2.4 Complex Behaviors from Local Rules

In the work of [12], modular self-reconfigurable robots are guided by simple local rules similar to colonies of social insects such as ants and termites. Distributed control algorithms were developed and tested on a simulation of hundreds of homogeneous Telecube [17, 18] modules.

These algorithms scale well as the number of modules is increased and the size of modules is decreased. Structures perform reconfiguration, locomotion (including turning), and navigation (i.e., path planning) to a goal site in an environment containing obstacles. Global motion of the structure arises from local control rules and communication between neighboring modules. Similar to the work in [3], the control algorithms use modes, seeds, and scents. Figure 11 shows an example of a collection of robots navigating around obstacles to reach a goal flag. The modules configure into a snake-like structure to squeeze between the two obstacles. The spine of the structure is formed as a result of modules parallel to the spine merging into the spine structure, and modules perpendicular to the spine moving towards the spine structure. The locomotion of the entire structure is centipede-like, resulting from a movement scent propagated between modules. In order to achieve turning, the structure forms a new snake in the perpendicular direction of movement.

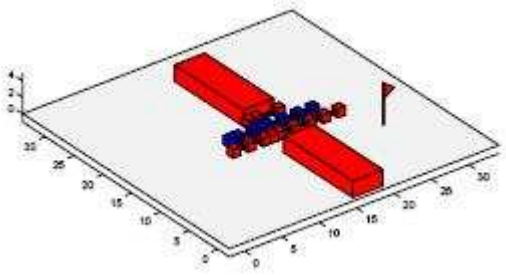

Figure 11. Snake-like structure of modules navigating through obstacles to a goal flag [12].

2.5 Reconfiguration for Heterogeneous Structures

Most research on self-reconfigurable robots has focused on structures containing homogeneous modules. However, Rus et. al [6] were the first to explore the possibility of using heterogeneous modules for self-reconfiguring robots. An advantage of heterogeneous modules is that modules can be specialized in that they contain a different set of sensors and actuators. A challenge for heterogeneous systems is that failed modules can no longer be replaced with any module. The researchers have found that reconfiguration of heterogeneous lattice-based systems does not asymptotically take any longer than the homogeneous counterparts, although an underlying assumption is that the modules operate in unbounded space, which may not always be the case in practical situations. Their modules apply the sliding-cube model, which is similar to the mechanism of Crystalline modules [16]. The algorithm used is the Melt-Sort-Grow algorithm, which is an extension of the Melt-Grow algorithm. The modules in the intermediate structure are sorted before they are placed into the final structure. This allows the goal structure to be easily grown from the intermediate structure. An assembly order of modules is computed and then the modules are physically sorted according to this computed ordering. Both centralized and decentralized versions of the algorithm were implemented. Decentralizing planning is preferred for a large number of modules. The decentralized algorithm uses a message passing protocol of communication, where modules are only allowed to communicate with their immediate neighbors. A simulation of the use of the Melt-Sort-Grow algorithm is shown in Figure 12, which involves configuring a chair into a table. There are two types of modules involved, one

for the legs, and the other for the rest of the structure.

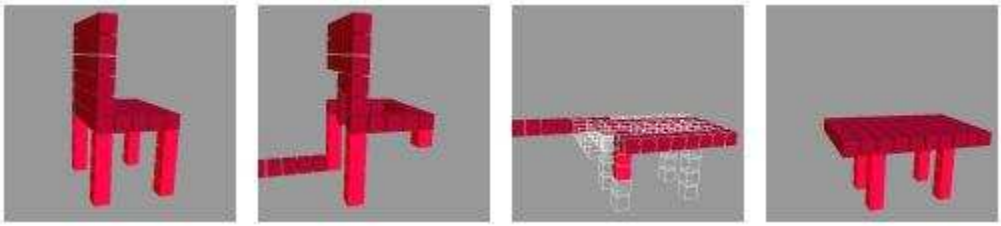

Figure 12. Transforming a chair shape into a table shape [6].

2.6 Distributed Assembly and Repair Algorithms

The work in [10] involves distributed control algorithms for synthesizing various shapes as well as repairing holes in structures. The algorithms are designed to be architecture-independent. The algorithms employ the use of rules that only require local information, similar to cellular automata. Restricting the scope of the algorithm to local information allows the algorithm to be easily distributed to systems containing a large number of modules. Each rule contains two portions – the precondition (left side) and the postcondition or result (right side). The basic modes of actuation are linear translation, concave transition, and convex transition, as described in the work for the Crystalline [16] and Molecule [11, 13] robots. Algorithms have been developed for locomotion, assembly, and repair. The algorithms involve locomotion with and without obstacles and include climbing tall obstacles and traversing tunnels. An example of assembling a cube structure from a sheet arrangement is depicted in Figure 13. Repair algorithms involve redistributing modules so as to retain a continuous structure of modules. An example of a hole-sealing application is illustrated in Figure 14. Practical applications of this technique include repairing space station walls upon colliding with foreign objects, and adaptive armor for military vehicles.

Figure 13. Building a cube structure from a flat sheet [10].

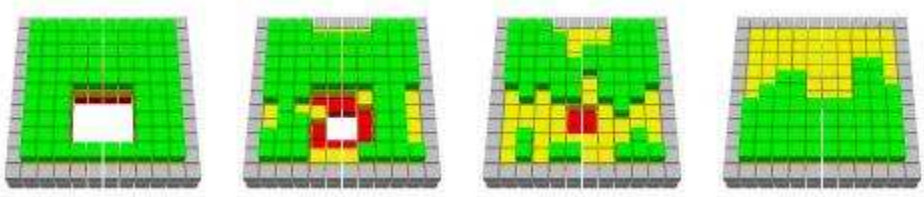

Figure 14. Repairing a hole [10].

2.7 Shape Sculpting via Hole Motion

The research of [4] is based on shape formation resulting from the random motion of regular holes in a lattice-based structure. Holes are created and deleted at the perimeter of the structure. Creating a hole leads to the enclosure of empty space which results in a bulge at that position. Deleting a hole will result in the structure caving in at that point. The creation and deletion processes are depicted in Figure 15. A smoothing technique is applied to prevent the development of areas where deletion cannot occur. Modules are packed into a hexagonal array and each module can have up to 6 possible neighbors. A hole is bordered by 12 modules. With a large enough hole, movement in the immediate neighborhood of a module will always be possible, thus many motion constraints can be avoided. The algorithm is massively parallel and fully distributed. Simulations involving simple structures have been conducted. These simulations include transforming square to T shape, T shape to square, and rectangle to circle. These transformations involve creating and deleting corners and creating curvature. The simulations have involved as many as 60,000 modules. The simulations are limited to 2D lattice structures, however, the researchers believe their algorithm can generalize to 3D, as well as scale to work with millions of modules. The Claytronics project [7-9] provides the inspiration for this work.

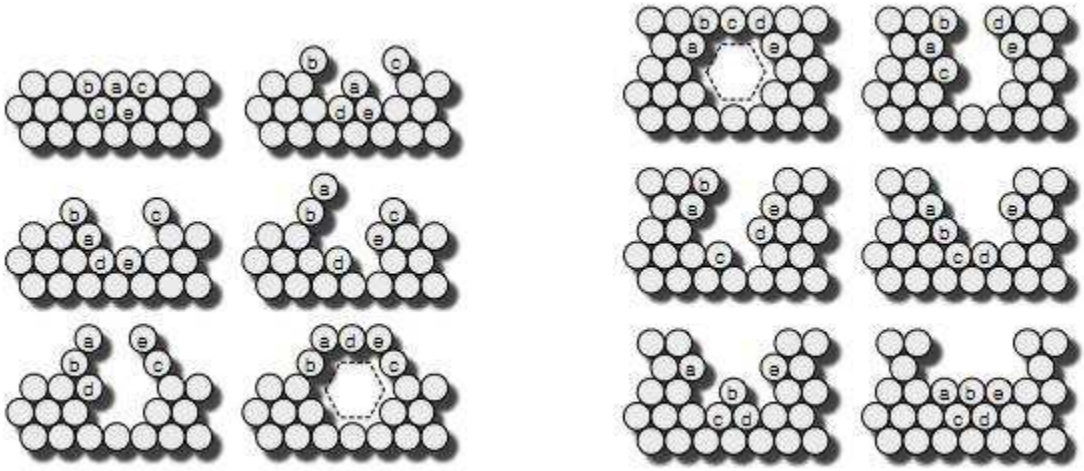

Figure 15. Hole creation (left) and deletion (right) [4].

2.8 Hierarchical Motion Planner

Researchers working on the Claytronics project [7-9] have developed a prototype hierarchical planning system [2] for their catoms (Claytronics atoms). It uses a divide and conquer approach where the algorithm calls a base planner and reuses precomputed plans at each level of the hierarchy. Modules are recursively grouped into metamodules, analogous to the self-similar structure of fractals. This technique minimizes the number of module moves. The highest level has the fewest meta-catoms, which allows traditional A* search algorithms to quickly compute a plan at this level. A metamodule contains seven metamodules (or modules at the lowest level) and a motion plan is determined for those seven metamodules. This scheme is depicted in Figure 16. The base planner uses a greedy nearest mismatched neighbor heuristic. This heuristic is

based on the distance between a module and its nearest neighbor in the goal state, and it also considers the desire to move towards free goal positions and not positions already occupied. Simulations using the algorithm have involved several thousands of modules. This metamodule scheme only works in the 2D case, but it is relatively straightforward to apply it in the 3D case.

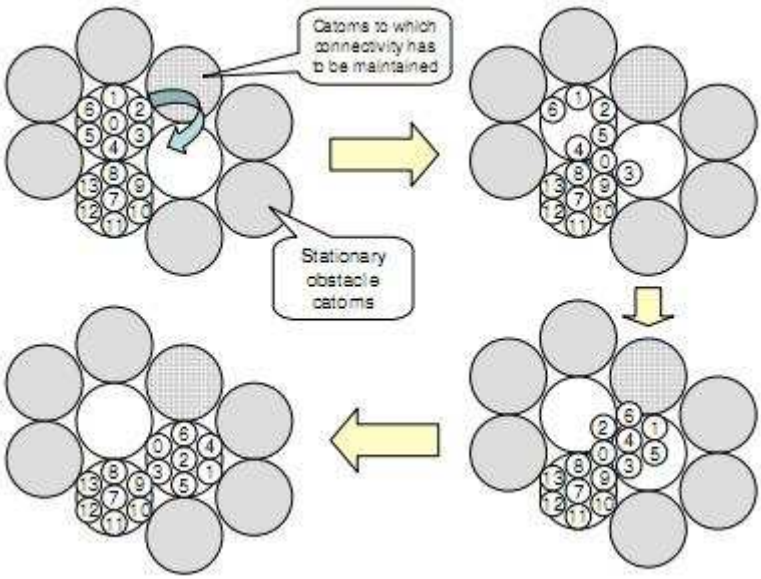

Figure 16. Metamodules containing modules [2].

2.9 Movement Primitives for Lattice-Based Modules

Movement primitives for closed lattice systems [19] have been devised based on Telecube [17, 18] and Crystalline [16] robots. A movement primitive is defined as an actuation sequence of one or more neighbors that can be applied to any subgroup of modules matching a target pattern [19]. This mechanism also allows for correcting misalignment and parallel actuation of modules. Seven different movement primitives are briefly outlined below:

1. **Mini-slide over, up, or across:** Moving a single module one position in the cases of slide-over and slide-up, and two positions in the case of slide across. Figure 17 illustrates this movement scheme.

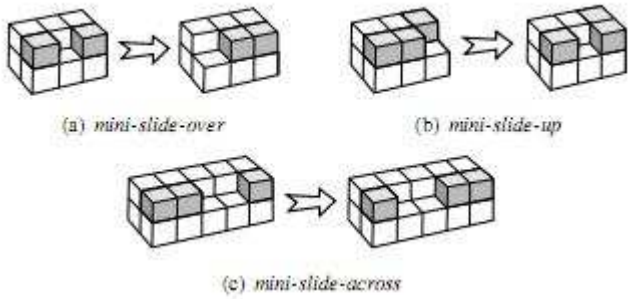

Figure 17. Mini-slide movement primitive [19].

2. **Slide:** Same as mini-slide except it involves moving a 2x2 group of modules, as in Figure 18.

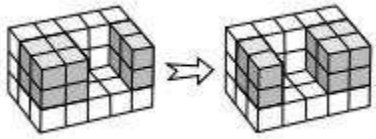

Figure 18. Slide movement primitive [19].

3. **Round:** A group of modules move around a convex corner, as seen in Figure 19.

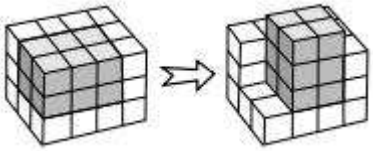

Figure 19. Round movement primitive [19].

4. **Blister:** A 2x2 piece is popped out from a two-module thick surface, as Figure 20 shows.

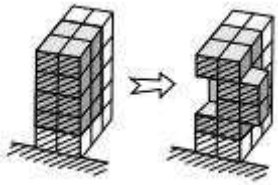

Figure 20. Blister movement primitive [19].

5. **Bubble:** An extension of blister that pushes out 8 modules and forms a 2x2x2 hole which can move around.

6. **Burst:** Pushes a bubble out of the interior, which leaves a depression in the surface.

7. **Spike:** Pushes one module out from the surface to function as a surface feature.

2.10 Shape Transformation for Multi-Robot Systems

Goldstein et al. [14] have developed an algorithm for multi-robot applications, which include sensor networks and formations. This algorithm is fully distributed with no need for global communication. The Hierarchical Median Decomposition (HMD) algorithm is based on the median consensus estimator. A collection of agents which have possibly different initial estimates of a global variable attempt to reach an agreement on the true value of the variable [14]. The basic idea of the algorithm is to establish a bijection from the initial configuration to the goal configuration. The algorithm consists of two steps; the first establishes the relative position of the agent, and the second establishes the final position in the goal configuration of the agent. The first step involves the agents achieving agreement on the median value of the x and y coordinates of their position. The algorithm iteratively builds a kd-tree, where the space is split into quadrants, left and right sides to determine the x coordinate, and up and down directions to determine the y coordinate, as illustrated in Figure 21. The second step of the algorithm is essentially the reverse of the first step.

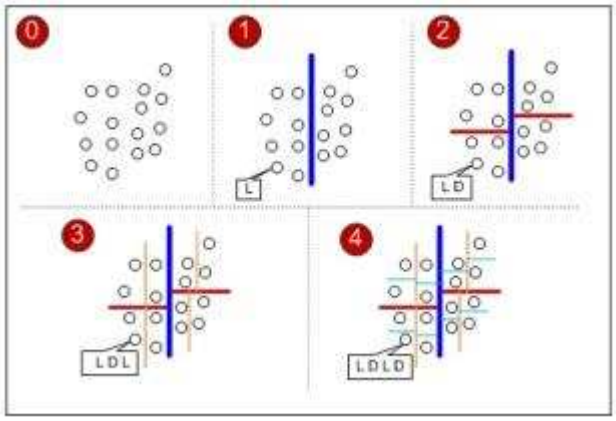

Figure 21. kd-tree scheme used in HMD algorithm to determine relative position [14].

2.11 Declarative Programming

Declarative programming approaches [1] have been implemented and applied to the shape planning problem. The execution of these languages is automatically distributed, but it appears that a single program is executing across the entire ensemble. States of a structure are modified by a rearrangement of labels representing them, where each label denotes a module or empty space in the lattice. The planner generates a sequence of rearrangements leading to the goal state. To ensure the structure remains globally connected, trees are grown from the connected sections, and only the leaves of the trees are allowed to be deleted. Two declarative programming languages for shape planning have been analyzed: Meld and LDP (Locally Distributed Predicates). Meld is a forward chaining system where a set of base facts is checked against a collection of production rules, and existing facts are combined to generate new facts. This process is iterated until all provable facts have been generated. In some cases, the generation of facts will result in side effects, such as actuation and locomotion. This system can also be applied to metamodules. The base facts include neighbor, position, vacant, and resources, while the facts with physical side effects are create, destroy, and give. LDP involves

predicates over the local neighborhood of a given node. Each predicate consists of a named node list, an expression, and any actions. A pattern matcher object in the form of an expression tree is used to detect predicates, and a match will trigger any actions associated with the predicate.

2.12 Self-Organization of Environmentally-Adaptive Shapes

Nagpal et. al [21, 22] have constructed a robot that functions as a self-balancing table by automatically adapting to changes in its environment. The structure is a chain-style robot that is formed from a flexible sheet with supporting legs. Applications of this type of robot include self-balancing furniture, terrain-adaptive bridges, and dynamic rendering for 3D media, as depicted in Figure 22. Modules contain tilt sensors, follow simple local rules, and communicate with their immediate neighbors. Structures perform shape change using a deformation mechanism in which modules compress or expand. Rather than specifying an absolute configuration, a shape is represented by a set of local sensor constraints between neighboring modules. This technique enables easier adaptation to environmental changes. Thus, the algorithm implements constraint maintenance in the form of distributed consensus, similar to the mechanism of biological activities such as flocking and firefly synchronization. A table-like structure is composed of surface (table top) and support (legs) groups of modules. Pivot modules are located between surface and support modules. The motion of the modules is determined by a distributed feedback control algorithm that iterates between two steps. In step 1, modules sense the tilt angle of each surface group. Each pivot collects the tilt data and computes the aggregate feedback. In step 2, each support group receives the tilt feedback and actuates accordingly by compressing or decompressing in an accordion-like fashion. The researchers have both developed a prototype and physics-based simulations. Experiments include simulations of six different shapes: statue, teapot, knot, bunny, donut, and face. The algorithms are effective for as many as 16,000 modules in simulation.

Figure 22. Environmentally adaptive structures. Self-balancing table, terrain-adaptive bridge, and dynamic rendering of 3D face [21].

2.13 Generalized Metamodules

Two reasons in favor of developing metamodules are for reducing the motion constraints of modules due to blocking, and to reduce the planning complexity. The metamodule itself is used as the basic unit of operation. The research of [5] involves developing generalized metamodules (and the associated generalized planner) that can be used in a variety of robots including robot arms, expanding cubes, and hex-packed spheres. A metamodule can be created or destroyed at any point in the lattice, as long as the creation or deletion occurs adjacent to an existing

metamodule. Thus, the system behaves similar to a compressible fluid that can expand or contract freely, as opposed to a set of discrete particles [5]. Metamodules are able to contain a variable number of modules, while maintaining physical stability and connectivity of the global structure. Metamodules move by exchanging modules between other metamodules and absorbing modules from a neighbor and moving them to another place. This mechanism is depicted in Figure 23. Experiments with these metamodules have indicated that the time to completion scales linearly with the diameter of the ensemble. A 400 metamodule simulation of transforming a flat rectangle to a standing rectangle is given in Figure 24.

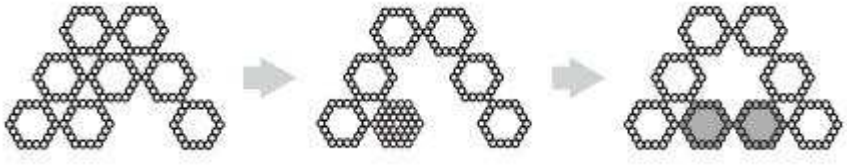

Figure 23. Motion of metamodules [5].

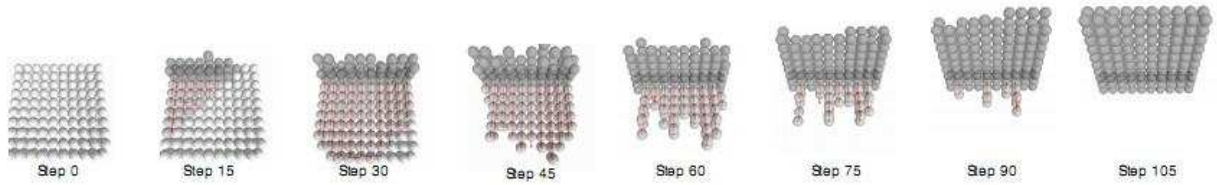

Figure 24. Converting a flat sheet into a standing wall using metamodules [5].

3 Conclusions

This report has presented several different algorithms for self-reconfigurable robot locomotion, assembly, reconfiguration, navigation, and repair. Most of the experiments with these algorithms have involved simulations of hundreds or thousands of modules. Many of the experimenters believe that their algorithms will scale well with increasing the number of modules, decreasing the size of modules, and operating in 3D as opposed to 2D. There have been promising results using heterogeneous modules as well as metamodules. Many of the algorithms analyzed in this report are distributed control algorithms based on local communication among neighboring modules. Several of the mechanisms for robot behavior are inspired by biological phenomena, such as amoeba inchworm movement, flocking, and scent gradients among social insect colonies. It would be interesting to learn how well these algorithms perform when applied to physical robotic modules and real-life tasks and environments. Also of interest would be to apply these algorithms to more complex structures, rather than the simple structures that most of these projects have worked with thus far. Results of such experiments would provide insight into the usefulness of these algorithms for real-world situations.

References

[1] Ashley-Rollman, M. P., De Rosa, M., Srinivasa, S. S., Pillai, P., Goldstein, S. C., Campbell, J. Declarative Programming for Modular Robots, In *Proceedings of the IROS*, 2007.

[2] Bhat, P., Kuffner, J., Goldstein, S., Srinivasa, S. Hierarchical Motion Planning for Self-Reconfigurable Modular Robots, In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, 2006.

[3] Bojinov, H., Casal, A., Hogg, T. Emergent Structures in Modular Self-Reconfigurable Robots, In *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, San Francisco, CA, USA, pp. 1734-1741, 2000.

[4] De Rosa, M., Goldstein, S., Lee, P., Campbell, J., Pillai, P. Scalable Shape Sculpting via Hole Motion: Motion Planning in Lattice-Constrained Modular Robots, In *Proceedings of the IEEE International Conference on Robotics & Automation*, 2006.

[5] Dewey, D. J., Ashley-Rollman, M. P., De Rosa, M., Goldstein, S. C., Mowry, T. C., Srinivasa, S. S., Pillai, P., Campbell, J. Generalizing Metamodules to Simplify Planning in Modular Robotic Systems, In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, Nice, France, 2008.

[6] Fitch, R., Butler, Z., Rus, D. Reconfiguration Planning for Heterogeneous Self-Reconfiguring Robots, In *Proceedings of IROS*, 2003.

[7] Goldstein, S. C., Mowry, T. Claytronics: A Scalable Basis for Future Robots (extended abstract), In *Robosphere*, 2004.

[8] Goldstein, S. C., Mowry, T. Claytronics: An Instance of Programmable Matter, In *Wild and Crazy Ideas Session of ASPLOS*, 2004.

[9] Goldstein, S. C., Campbell, J. D., Mowry, T. C., Programmable Matter, *Computer*, Vol. 38, No. 6, pp. 99-101, June 2005.

[10] Kotay, K., Rus, D. Generic Distributed Assembly and Repair Algorithms for Self-Reconfigurable Robots, In *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004.

[11] Kotay, K., Rus, D., Vona, M., McGray, C. The Self-Reconfiguring Robotic Molecule: Design and Control Algorithms, In *the 1998 Workshop on Algorithmic Foundations of Robotics*, 1998.

[12] Kubica, J., Casal, A., Hogg, T. Complex Behaviors from Local Rules in Modular Self-Reconfigurable Robots, In *Proceedings of the 2001 IEEE International Conference on Robotics & Automation*, Seoul, Korea, pp. 360-367, 2001.

[13] McGray, C., Rus, D. Self-Reconfigurable Molecule Robots as 3D Metamorphic Robots, In *Proceedings of the International Conference on Intelligent Robots and Systems*, 1998.

[14] Ravichandran, R., Gordon, G., Goldstein, S. C. A Scalable Distributed Algorithm for Shape Transformation in Multi-Robot Systems, In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems IROS*, 2007.

[15] Rus, D., Vona, M. Self-Reconfiguration Planning with Compressible Unit Modules, In *Proceedings of the IEEE International Conference of Robotics & Automation*, Detroit, MI, USA, Vol. 4, pp. 2513-2520, 1999.

[16] Rus, D., Vona, M. A Physical Implementation of the Self-Reconfiguring Crystalline Robot, In *Proceedings of the IEEE International Conference on Robotics & Automation*, pp. 1726-1733, 2000.

[17] Suh, J. W., Homans, S. B., Yim, M. Telecubes: Mechanical Design of a Module for Self-Reconfigurable Robotics, In *Proceedings of the IEEE International Conference on Robotics & Automation*, Washington DC, USA, pp. 4095-4101, 2002.

[18] Vassilvitskii, S., Kubica, J., Rieffel, E., Suh, J., Yim, M. On the General Reconfiguration Problem for Expanding Cube Style Modular Robots, In *Proceedings of the IEEE International Conference on Robotics & Automation*, Washington DC, USA, 2002.

[19] Weller, M. P., Karagozler, M. E., Kirby, B., Campbell, J., Goldstein, S. C. Movement-Primitives for an Orthogonal Prismatic Closed-Lattice-Constrained Self-Reconfiguring Module, In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems IROS*, 2007.

[20] Yim, M., Zhang, Y., Lamping, J., Mao, E. Distributed Control for 3D Metamorphosis, *Autonomous Robots*, Vol. 10, No. 1, pp. 41-56, 2001.

[21] Yu, C., Willems, F., Ingber, D., Nagpal, R. Self-Organization of Environmentally-Adaptive Shapes on a Modular Robot, In *Proceedings of IROS*, 2007.

[22] Yu, C., Haller, K., Ingber, D., Nagpal, R. Morpho: A Self-Deformable Modular Robot Inspired by Cellular Structure, In *Proceedings of IROS*, 2008.