

Agent-Based Control for Object Manipulation with Modular Self-reconfigurable Robots

Jeremy Kubica and Arancha Casal and Tad Hogg

Xerox Palo Alto Research Center

Palo Alto, CA 94304

Abstract

We demonstrate multiagent control of modular self-reconfigurable (MSR) robots for object manipulation tasks and show how it provides a useful programming abstraction. Such robots consist of many modules that can move relative to each other and change their connectivity, thereby changing the robot's overall shape to suit different tasks. We illustrate this approach through simulation experiments of the TeleCube MSR robot system.

1 Introduction

Modular self-reconfigurable (MSR) robots [37, 31, 38, 32, 25, 29, 30, 9] consist of many identical modules that can attach and detach from one another to change their overall connectivity. Each module has processing, sensing and actuation capabilities. The main characteristic of MSR robots is their ability to change shape to suit their task. While such shapes are unlikely to be as effective at particular tasks as special purpose tools (e.g., the modules are unlikely to have the same strength as, say, a hammer), they provide adaptability for a wide range of tasks in unpredictable environments. For instance, changing shape can provide a variety of gaits for locomotion, e.g., wheels for flat surfaces, snakes for small tunnels, etc. Robot shapes matching object contours can also help provide precise manipulation.

Beyond the challenge of building many such modules, their tight physical interactions and many degrees of freedom pose a difficult control problem. Moreover, modules constructed of micromachines (MEMS) [7] or even smaller devices [14, 11, 10] are likely to have limited capabilities and be prone to some failures or incorrect assembly. Thus the robust functioning of the robot requires a control architecture that tolerates such limitations, as has also been proposed for computing with molecular-scale devices [18].

One control approach would be to precisely specify the desired locations of all the modules, and perform a combinatorial search to find appropriate module motions according to some criterion, such as minimizing moves or power consumption. With many modules, such searches are generally intractable, but can be solved approximately using heuristics [31, 38, 29, 25, 32, 9, 20]. Even so, a high-level controller often lacks enough information on the modules' status or the

task environment to define a precise shape, e.g., when grasping an object whose detailed size or shape is unknown.

Instead, a structure with the general properties required for the task is often sufficient. When the properties are determined mainly by the local environment for each module, agent-based local control often finds a suitable shape without any need to precisely specify each module's position [4]. A multiagent approach to general behaviors, such as locomotion and object manipulation, simplifies the design of higher-level control by presenting a useful programming abstraction to a centralized planner or within levels of a sequence of increasingly complex behaviors [6]. That is, higher level control programs can determine the choice of local rules for the agents without detailed concern of the particular motion primitives of the modules or situation of each agent individually. For example, by switching among a set of available behaviors or modifying parameters used by the agents' rules.

Agent-based architectures readily match control to physical phenomena dominant at the different scales relevant to MSR robots with many modules. For example, micromachined robots [12] are dominated by friction and other surface forces rather than gravity. Even smaller structures [21] are subject to randomly fluctuating forces, i.e., Brownian motion. Biological structures provide numerous other examples [36]. In such cases, different agent types could be responsible for individual modules, small groups of modules and so on, forming a hierarchical or multihierarchical correspondence between the robot's physical structures and its environment, task and the controlling software [19].

In related work, multiagent control helps teams of robots cooperating to achieve a common task [35, 8, 33, 17, 24]. These teams usually consist of independently mobile robots that are not physically connected and have little or infrequent physical contact. In most current modular robot systems the modules remain attached to one another forming a single connected whole, and giving rise to a number of tight physical motion constraints that do not apply to teams of separate robots. On the other hand, the physical contact between modules allows them to locate their neighbors without complex sensory processing as would be required, for example, to visually identify a physically disconnected member of a team. Hence the techniques for coordinating teams address somewhat different requirements than those of MSR robots.

The key issue of relating local rules to global behavior also

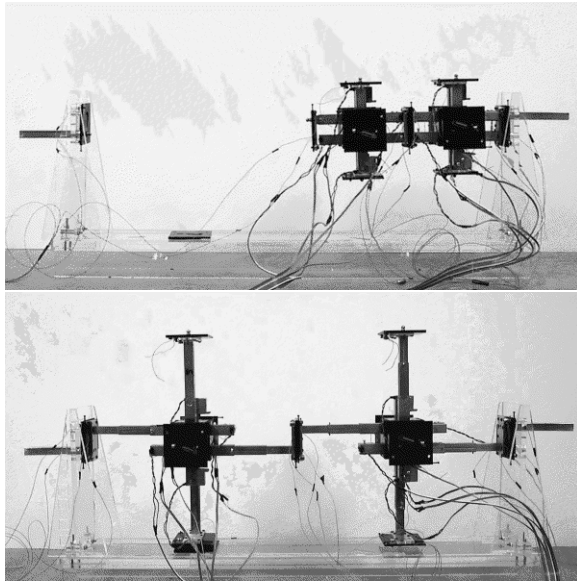


Figure 1: The TeleCube hardware: two modules shown with arms retracted and expanded. The current modules are about 15cm in diameter when fully contracted.

arises in artificial life [13] which concentrates on how complex natural organisms achieve sophisticated crowd behaviors or deal with abstract agents that currently can not be physically constructed [15, 16, 1]. Alternatively, biological and chemical techniques [5, 27] give complex shapes from local behaviors, but can not yet produce programmable robots.

Thus while multiagent systems have been applied to several related tasks, identifying appropriate agent behaviors for MSR robots remains an open problem. In the remainder of this paper, we describe the TeleCube modular robot and simple agent behaviors that provide basic object manipulation capabilities. These agent behaviors, in turn, abstract away the detailed nature of the module motion capabilities, providing a simpler programming model for higher level control.

2 The TeleCube Modular Robot

This section describes the modular robot we used, its simulator and the agent-based control technique.

2.1 Hardware

TeleCube is a new MSR robot where each module is a cube that can prismatically extend each of its six faces independently up to a factor of two times its fully retracted configuration (Fig. 1). A 2D analogue was previously developed at Dartmouth [32]. Each module can communicate with its immediate neighbors in all six directions. When a module extends an arm in contact with an object, it can exert a force on the object. This design allows modules bound inside a group to move, so TeleCube can perform tasks inside the aggregate, such as internal object manipulation and density changes.

2.2 Simulator

To examine behavior with more modules than currently feasible to construct, we simulated the TeleCube system. The simulator, written in Java, accounted for the motions of the modules in three dimensions and typically ran with hundreds of modules. The simulator performs a series of steps. For each step, a random ordering of the modules is selected, thereby simulating asynchronous module actions. Using this ordering, each module is given a “turn” in which to decide its behavior and perform any motion it selects.

For vertical motion, the simulator included the effects of gravity, as appropriate for relatively large scale objects such as the current TeleCube modules. For smaller scale modules, viscous and frictional forces will become more important and could be included instead of gravity.

The simulator makes several simplifying assumptions. First, the module arms have only two states: fully expanded or contracted. Second, the arms are infinitely rigid. Both of these assumptions simplify alignment of modules after a move. In practice, such alignment can be done with low-level feedback control or engineering the connectors on the module arms to guide slightly misaligned parts into alignment. Lastly, the simulator assumed enough friction between the module and the floor to eliminate any sliding. We use a motion primitive to ensure correct module movement by including required low-level actions such as disconnection from modules perpendicular to the motion direction, connection with any new neighbors and possible arm contraction while a neighbor extends its arm [26]. A global connectivity check insures disconnecting from a neighbor does not separate the ensemble. This check may be easily preformed in hardware, and was simulated with a depth-first search of the modules.

2.3 Agent-Based Control

Our approach views each module as an independent agent executing a simple program, consisting of a finite-state machine (FSM). This approach uses the distributed, homogenous and networked nature of MSR robots and scales well as the number of modules increases. There is no central control processor or designated leader module. Global motion results, or “emerges”, from purely local control rules executed at every module and through communication between neighboring modules alone. These behaviors can then be turned on and off by a higher-level system to give overall control for a sequence of tasks.

The FSM control consists of simple if-then rules which may act probabilistically. Modules switch among “states” that determine their behaviors and communicate by sending and receiving “messages”. Randomization helps prevent modules from becoming indefinitely stuck in unproductive configurations. It also prevents all modules from moving at once, potentially disconnecting the structure or leading to oscillations due to synchronous activity in which modules respond based on a system state that no longer holds if too many neighbors move simultaneously [22].

Messages are the means of global communication among modules. Messages are propagated through the system in a distributed breadth-first fashion and can be modified as they move from one module to another. A “scent” is a message

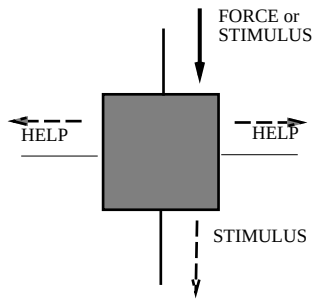


Figure 2: Message passing from a **column** module. Help messages are also sent in the other two directions: into and out of the page.

consisting of a numerical value that is slightly adjusted up or down as it passes through the modules [4]. Propagating scents form global gradients that can guide module motions. For instance, a scent indicating the distance to some location (e.g., a module under high stress) could consist of a single integer value a module sends to its neighbors. A module increments the minimum value of all such messages it receives and sends the new value to its neighbors. Using the *minimum* function to combine values from neighbors allows the scent to provide a useful gradient in spite of many cycles in the graph of links among the modules. These messages can have a maximum number of iterations to prevent them passing among modules indefinitely. Once these messages have passed through a region of modules, the difference in recorded values between one module and a neighbor is a local scent gradient.

Biological concepts, such as scents and hormones, have been applied to MSR robots for reconfiguration, grouping, synchronization and locomotion [4, 26, 34]. Extending these concepts to consider gradients with TeleCube’s ability to move modules within a structure allows object manipulation as described below.

We assume the modules are small compared to the size of objects in their environment, allowing many modules to act on them at once. Furthermore, we take module motions to be fast compared to the other object speeds, so numerous module updates take place with little change in the objects’ positions. Since modules move slower than they communicate, the response to messages does not introduce oscillating behavior [23] or irregular growth, and the intermediate configurations are well-balanced at all times.

3 Results

This section presents the rules and resulting behaviors for two tasks involved in object manipulation.

3.1 Growing toward an object

Our first task is a dynamic supporting structure in which modules under pressure signal others to add their support, reducing the weight pressure (i.e., force per supporting module) and providing more uniform support for the object. This application requires a pressure sensor in each module. Control is handled through four different states: **relaxed**, **column**, **expanding**, and **reset** and three different messages: *stimulus*, *help*, and *reset*.

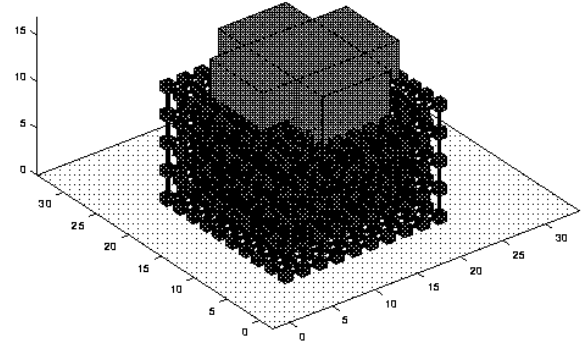
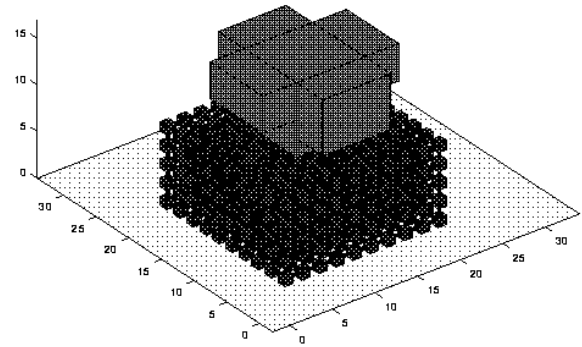


Figure 3: Growing up toward an arbitrary-shaped object. This example required 30 simulation steps.

All modules begin in a **relaxed** state. When a module receives an external force stimulus, it sends a *stimulus* message to the modules along the direction of the force (one of the module’s six faces). Upon receiving the message the module transitions to the **column** state and effectively forms a rigid support column. This column sends *help* messages to neighboring modules, which ask them to expand toward the object and provide support. The remainder of this section provides details of the behaviors.

Behavior for one turn in the **relaxed** state is:

- If a *stimulus* message is received, transition to **column** state and end turn.
- If a *help* message is received, record the direction from which the message came (the “help direction”), transition to **expanding** state, and end turn.
- If contact with an object is detected, transition to **column** state and end turn.
- Otherwise, when no message is received, adjust module density in all six directions as follows: For each direction, if the module is farther from or closer to a neighbor than one fully extended arm length, move towards or

away from the neighbor, respectively, with a probability of 50%. This motion tends to restore a uniform density.

Modules in the **expanding** state move in the direction where additional support is required (the “help direction”). However, to insure that the bottom level remains connected, the last module in each column (as determined by the direction of the original force stimulus) does not move or disconnect from its neighbors. Behavior of the **expanding** state is:

- If a *help* message is received, update help direction.
- If a *stimulus* message is received, transition to **column** state and end turn.
- If the force stimulus is removed, transition to **reset** state and end turn.
- If contact with an object is detected, transition to **column** state and end turn.
- Otherwise, send *help* messages in all six directions and, with 50% probability, move in the direction of required help, unless the opposite direction has no neighbor.

Modules in the **column** state remain rigid and stationary, providing support for an object. They emit *help* and *stimulus* scents to the appropriate neighbors, effectively calling for help and telling the others in the column to hold fast. Behavior of the **column** state is:

- If the force stimulus is removed, transition to **reset** state and end turn.
- Send *stimulus* messages to the modules above and below as determined by the direction of stimulus.
- Send *help* messages to modules perpendicular to the direction of stimulus, as shown in Fig. 2.

The **reset** state is used when the object is removed so modules return to their original configuration. A module in the **reset** state sends the *reset* message to all neighbors in the direction along which the force previously acted and transitions to the **relaxed** state.

3.2 Manipulating an object

The TeleCube architecture allows manipulating objects inside a group of modules. This can be accomplished by growing around a supported object or, more generally, opening gaps within the structure and pushing an object into them.

Internal object manipulation requires translational and rotational motions. In both cases, a module that is not in direct contact with an object moves based on the gradient of messages it receives from its neighbors, determined by comparing the relative numerical values of the “scent” messages. Namely, if possible, a module will move in the direction of the positive gradient. If the module does not sense a gradient, it will try to move so as to restore an “optimal” density of modules in its neighborhood. This means moving away from a neighbor that is too close or toward a distant neighbor.

Internal manipulation uses **shell** and **tissue** states, and one *stimulus* message propagated among neighbors to form a gradient scent. The process starts with a broadcast of the desired

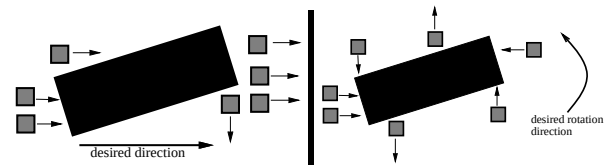


Figure 4: Module motions to move (left) and rotate (right) an object in 2D. Only a few of the modules surrounding the object are shown.

movement direction. Modules on the outside of the structure enter the **shell** state, and do not move under the influence of gradients or density adjustments. Thus, these modules form a rigid support structure off which inside modules can push, i.e., MSR robots can not only form tools shaped to their task, but can also provide dynamic support structures for using those tools. Interior modules enter the **tissue** state.

Translational 2D motion uses a simple stimulus-based response. Given the desired direction of motion, modules in contact with the object use the following rules:

- If a module is in front of the object (i.e., touching it with an arm extending from the module in a direction going against the desired direction of object movement), it releases a negative *stimulus* message and then tries to move away from the object. If possible, it moves in the direction opposite to the arm touching the object. If that direction is blocked by other modules, it tries to move instead perpendicular to that direction. Either motion tends to move the module out of the object’s way. Furthermore, the negative *stimulus* message creates a scent gradient among neighbors causing them to also move away and make room for the module to move.
- If a module is behind the object (i.e., touching it with an arm in a direction aligned with the desired direction of object movement), it releases a positive *stimulus* and tries to move toward and push the object. If the module has more than one arm in contact with the object, this rule applies to the arm most closely aligned with the desired direction of motion.

These behaviors are illustrated in Fig. 4(left) and result in modules pushing the object in the desired direction.

Rotation requires the approximate location of the object’s center to allow relating individual module forces to the overall torque imposed on the object. Estimating this location via a distributed local algorithm can be difficult if the modules are much smaller than the object and, on the scale of the modules, the object is not smooth. Instead, we assume the higher level controller broadcasts the approximate location of the object’s center. Another possibility is to use open-loop force fields to position the object in a known starting configuration, as demonstrated on a micromachined surface [3]. In either case, for the behavior described here we suppose knowledge of the approximate center. That is, any error in the center’s specified location is small compared to the object’s size, but not necessarily small compared to the size of the modules. The modules act in the same way as the translation modules, except they move away from or towards the object if they are in front of a corner that should be turned

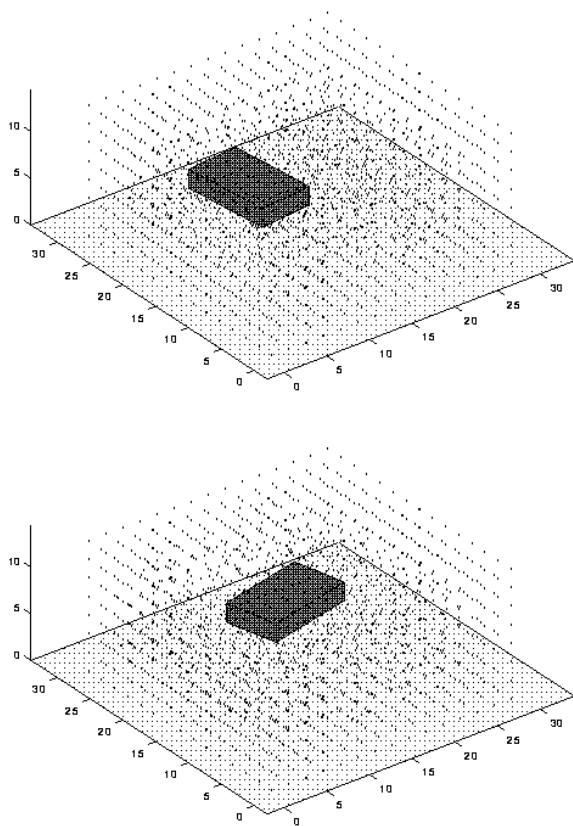


Figure 5: Internally manipulating an object. The modules form a $15 \times 15 \times 8$ structure, and are shown simply as points to allow seeing the internally manipulated object. The full sequence, consisting of 138 simulation steps, shows the object gradually shifted from its initial to final positions in response to instructions to move the object in an “L” shaped path. The figure shows steps 15 and 80.

towards or away from them, respectively, as determined by their relative location to the specified center and the desired direction of rotation. Again, they release positive and negative *stimulus* messages when moving towards and away from the object, respectively, and push the object when they move towards it. This behavior, shown in Fig. 4(right), gives 2D rotation in a plane.

Fig. 5 includes both types of motion. First the modules are told to move the object in one direction, then to rotate it, and finally move in the perpendicular direction. These instructions cause modules to switch between translation and rotation rules, giving desired motions without the higher-level control specifying individual module motions. We found the local rules could produce a variety of trajectories. The combination of translation and rotation rules allows achieving arbitrary planar position and orientation of the object.

Modules move the object by exerting forces on it. They could also exert compressive forces on the object by simply replacing the negative *stimulus* messages with positive ones so modules on all sides move toward the object. This behavior could be used to push parts together. Shear forces could

arise from tangential motion, provided friction between the arms of the modules and the object was sufficient. The modules then can be viewed as a material surrounding the object with forces under programmed control, changing from rigid support to fluid-like flow as needed. This behavior is a useful programming abstraction for higher levels of control, similar to programmable force fields in two dimensions [3]. By giving different instructions to different regions, a higher-level control could use the modules to move multiple objects along different paths or, through feedback based on the position of the objects (rather than details of the much larger number of modules), bring objects together.

4 Discussion

This paper described local rules for two object-handling behaviors for TeleCube robots, dynamic support and internal manipulation. Local rules can also reconfigure them for locomotion and navigation [26].

Our results extend prior distributed control [4] of Proteo [38], another MSR robot whose motions differ in two important ways from TeleCube. First, Proteo modules move independently. By contrast, a TeleCube module usually requires cooperative actions from neighbors to push or pull it, as shown in Fig. 1. Second, Proteo modules are rigid and can only move on the external surface created by other modules. Thus, Proteo can not internally manipulate objects unless the control creates holes inside the structure. TeleCube modules can deform and move inside a structure of other modules. The different hardware capabilities of Proteo and TeleCube affect the design and complexity of the local control rules that are appropriate to consider. Thus hardware designs should balance the control complexity and the manufacturing difficulty. Ideally, the hardware should support simple high-level programming abstractions while the control simplifies the required hardware capabilities. Schemes based on local (module-level) control help with this since they are simple and can be easily modified to apply to a wide range of module designs, including Proteo and prismatic robots, such as TeleCube and the Crystalline robot at Dartmouth [32].

Our approach could be combined with other methods, as part of an overall hierarchical control scheme. Moreover, testing a population of agents against variations in the desired task can evolve better behaviors [2]. For instance, with the primitives presented in this paper, genetic techniques could identify methods to modulate the rule parameters and switch among rules to achieve higher-level goals.

As robots with many modules are built, one important question is how module failures affect system behavior. Scaling is another important issue, i.e., increasing the number of modules or decreasing their size, or both. More modules allow greater variety of interactions and statistically more robust randomized behaviors. Smaller modules allow finer scale object manipulation. However, smaller modules take smaller steps resulting in slower manipulation unless they complete each step more rapidly. This observation highlights the importance of balancing the scaling of speed, strength and other module properties [11, 36] as, for example, with protein motors that carry weights much larger than their own [21, 28].

Multiagent control can balance competing goals to help such robots achieve robust behaviors.

References

- [1] Harold Abelson et al. Amorphous computing. Technical Report 1665, MIT Artificial Intelligence Lab, August 1999.
- [2] Forrest H. Bennett III and Eleanor G. Rieffel. Design of decentralized controllers for self-reconfigurable modular robots using genetic programming. In *Proc. of the 2nd NASA/DOD Workshop on Evolvable Hardware*, July 2000.
- [3] Karl F. Bohringer et al. Computational methods for design and control of MEMS micromanipulator arrays. *Computational Science and Engineering*, 4(1):17–29, January–March 1997.
- [4] Hristo Bojinov, Arancha Casal, and Tad Hogg. Multiagent control of modular self-reconfigurable robots. In *Proc. of Intl. Conf. on Multiagent Systems (ICMAS2000)*, 2000. An extended version is Los Alamos preprint cs.RO/0006030.
- [5] Ned Bowden, Andreas Terfort, Jeff Carbeck, and George M. Whitesides. Self-assembly of mesoscale objects into ordered two-dimensional arrays. *Science*, 276:233–235, 1997.
- [6] Rodney A. Brooks. New approaches to robotics. *Science*, 253:1227–1232, September 13 1991.
- [7] Janusz Bryzek, Kurt Petersen, and Wendell McCulley. Micro-machines on the march. *IEEE Spectrum*, pages 20–31, May 1994.
- [8] P. Caloud et al. Indoor automation with many mobile robots. In *Proc. of the Intl. Workshop on Intelligent Robots and Systems*. IEEE, 1990.
- [9] Arancha Casal and Mark Yim. Self-reconfiguration planning for a class of modular robots. In *SPIE Symposium on Intelligent Systems and Advanced Manufacturing: Sensor Fusion and Decentralized Control in Robotic Systems*, pages 246–257, 1999.
- [10] H. G. Craighead. Nanoelectromechanical systems. *Science*, 290:1532–1535, 2000.
- [11] K. Eric Drexler. *Nanosystems: Molecular Machinery, Manufacturing, and Computation*. John Wiley, NY, 1992.
- [12] Thorbjörn Ebefors et al. A walking silicon micro-robot. In *Proc. of the 10th Intl. Conf. on Solid-State Sensors and Actuators (TRANSDUCERS99)*, pages 1201–1205, 1999.
- [13] Joshua M. Epstein and Robert Axtell. *Growing Artificial Societies*. MIT Press, Cambridge, MA, 1996.
- [14] Robert A. Freitas Jr. *Nanomedicine*, volume 1. Landes Bioscience, 1999.
- [15] S. Hackwood and G. Beni. Self-organization of sensors for swarm intelligence. In *Proc. of the Conference on Robotics and Automation (ICRA92)*. IEEE, 1992.
- [16] J. Storrs Hall. Utility fog: The stuff that dreams are made of. In B. C. Crandall, editor, *Nanotechnology*, pages 161–184. MIT Press, Cambridge, MA, 1996.
- [17] Brosl Hasslacher and Mark W. Tilden. Living machines. In L. Steels, editor, *Robotics and Autonomous Systems: The Biology and Technology of Intelligent Autonomous Agents*. Elsevier, 1995.
- [18] James R. Heath, Philip J. Kuekes, Gregory S. Snider, and R. Stanley Williams. A defect-tolerant computer architecture: Opportunities for nanotechnology. *Science*, 280:1716–1721, 1998.
- [19] Tad Hogg and Bernardo A. Huberman. Controlling smart matter. *Smart Materials and Structures*, 7:R1–R14, 1998. Los Alamos preprint cond-mat/9611024.
- [20] K. Hosokawa et al. Self-organizing collective robots with morphogenesis in a vertical plane. In *Proc. of the Conference on Robotics and Automation (ICRA98)*. IEEE, 1998.
- [21] Joe Howard. Molecular motors: Structural adaptations to cellular functions. *Nature*, 389:561–567, 1997.
- [22] Bernardo A. Huberman and Natalie S. Glance. Evolutionary games and computer simulations. *Proceedings of the National Academy of Science USA*, 90:7716–7718, August 1993.
- [23] J. O. Kephart, T. Hogg, and B. A. Huberman. Dynamics of computational ecosystems. *Physical Review A*, 40:404–421, 1989.
- [24] Hiroaki Kitano, editor. *Robocup-97: Robot Soccer World Cup I*, volume 1395 of *Lecture Notes in Computer Science*. Springer, Berlin, 1998.
- [25] Keith Kotay, Daniela Rus, Marsette Vona, and Craig McGray. The self-reconfiguring robotic molecule: Design and control algorithms. *Algorithmic Foundations of Robotics*, 1998.
- [26] Jeremy Kubica, Arancha Casal, and Tad Hogg. Complex behaviors from local rules in modular self-reconfigurable robots. In *Proc. of ICRA2001*, 2001.
- [27] Ross J. Metzger and Mark A. Krasnow. Genetic control of branching morphogenesis. *Science*, 284:1635–1639, 1999.
- [28] Carlo Montemagno and George Bachand. Constructing nanomechanical devices powered by biomolecular motors. *Nanotechnology*, 10:225–231, 1999.
- [29] Satoshi Murata et al. A 3-D self-reconfigurable structure. In *Proc. of the Conference on Robotics and Automation (ICRA98)*, page 432. IEEE, 1998.
- [30] Satoshi Murata, Haruhisa Kurokawa, and Shigeru Kokaji. Self-assembling machine. In *Proc. of the Conference on Robotics and Automation (ICRA94)*, pages 441–448, Los Alamitos, CA, 1994. IEEE.
- [31] Amit Pamecha, Imme Ebert-Uphoff, and Gregory S. Chirikjian. Useful metrics for modular robot motion planning. *IEEE Transactions on Robotics and Automation*, 13:531–545, 1997.
- [32] D. Rus and M. Vona. Self-reconfiguration planning with compressible unit modules. In *Proc. of the Conference on Robotics and Automation (ICRA99)*. IEEE, 1999.
- [33] J. R. Rush, A. P. Fraser, and D. P. Barnes. Evolving cooperation in autonomous robotic systems. In *Proceedings of the IEEE International Conference on Control*, March 21–24 1994.
- [34] B. Salemi, W.-M. Shen, and P. Will. Hormone controlled metamorphic robots. In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA2001)*, 2001.
- [35] Luc Steels. Cooperation between distributed agents through self-organization. *Journal on Robotics and Autonomous Systems*, 1989.
- [36] Darcy Wentworth Thompson. *On Growth and Form*. Cambridge University Press, Cambridge, 1992.
- [37] Mark Yim. *Locomotion with a Unit-Modular Reconfigurable Robot*. PhD thesis, Stanford University, 1994.
- [38] Mark Yim, John Lamping, Eric Mao, and J. Geoffrey Chase. Rhombic dodecahedron shape for self-assembling robots. Technical Report P97-10777, Xerox PARC, 1997.